Support your visions

# User Manual

SIMULATION X®

Powered by ITI

**©Copyright ITI GmbH, 2009**
Der Inhalt dieses Handbuches sowie die Rechte an der zugehörigen SimulationX-Software sind Eigentum der ITI GmbH Dresden und urheberrechtlich geschützt. Die Vervielfältigung, Verarbeitung auf elektronischen Systemen und Verbreitung des Handbuches oder der Software beziehungsweise von Teilen davon bedürfen der Zustimmung der ITI GmbH.

Für die Nutzung von ITI-Software gelten ausschließlich die Vertragsbedingungen der ITI GmbH für die Überlassung von Software („AGB Software"). Beachten Sie die dort genannten Hinweise zur beschränkten Haftung. ITI ist eine eingetragene Marke der ITI GmbH. Alle anderen im Handbuch genannten Marken und Produktnamen sind Eigentum des jeweiligen Unternehmens.

*The contents of this manual and the rights to the SimulationX software described therein are the property of ITI GmbH Dresden, Germany, and are protected by copyright. Any reproduction, transmission, or electronic processing of the manual or the software or parts thereof in any manner whatsoever is prohibited.*

*Use of ITI software is governed exclusively by the terms of contract for software licensing ("GTC Software") of the ITI GmbH. Attention is called to the stipulations on limited liability contained therein. ITI is a registered trademark of ITI GmbH. Any other trademark and product names given in the manual are the property of their respective owners.*

**ITI GmbH**
Webergasse 1 · Haus C2
01067 Dresden · Germany
T + 49 (0) 351.260 50 - 0
F + 49 (0) 351.260 50 - 155
support@iti.de

**www.iti.de**

# Zu den Handbüchern

Die Handbücher sind Ihre Begleiter beim Arbeiten mit dem Programmpaket SimulationX. Sie sind für alle Mitarbeiter in Industrie und Forschung geschrieben, die sich mit der Simulation und Analyse technischer Systeme befassen.
Durch die Aufteilung der einzelnen Handbücher nach thematischen Gesichtspunkten und das Darstellen von Beispielen sind sie auch für den Einstieg in SimulationX geeignet. Nutzen Sie die Handbücher und die kontext-sensitive Online-Hilfe um das umfangreiche Leistungspaket schnell zu erschließen. Bei der Suche nach Informationen orientieren Sie sich an folgender inhaltlicher Aufteilung:

### Teil I Das User Manual

gibt eine Einführung in das Konzept von SimulationX**.** Alle zur Durchführung einer Simulation erforderlichen *Arbeitsabläufe und Werkzeuge* werden erklärt. Hier finden Sie Hinweise zur *Modellierung* und *Berechnung*, sowie alles über *die Arbeit mit Ergebnisgrößen* und verfügbare *Schnittstellen*.
Die Einführung in den TypeDesigner ermöglicht eigene Modellelemente zu erstellen, vorhandene zu erweitern (Vererbung) und Teilmodelle zu wieder verwendbaren Compounds zusammenzufassen.

### Teil II Das Library Manual

stellt die Inhalte der Bibliotheken im Einzelnen vor. Neben der Symbolik und der Beschreibung der einzelnen Modellelemente sind Angaben zu den Anschlüssen, den Parametern und den Ergebnisgrößen enthalten. Dieses Handbuch ist nur in **englischer** Fassung verfügbar.

- ***ITI Internal Libraries***
  SimulationX stellt eine Sammlung von Bibliotheken für verschiedene physikalische Domänen und Applikationsfelder bereit. Diese SimulationX-Bibliotheken werden von ITI entwickelt, gepflegt und sind validiert und getestet.

- ***ITI External Libraries***
  In diesen Bibliotheken werden Modellelemente für spezielle Anwendungen, z.B. Verbrennungsmotoren zur Verfügung gestellt, die vom Anwender durch entsprechende Modifizierung an das konkrete Aufgabengebiet angepasst werden können.

Legende:

Option, nicht im Basismodul von SimulationX enthalten

Hinweis auf Warnungen

Hinweis auf einen Tipp

Hinweis auf ein Beispiel oder eine ausführbare Handlung

*Kursiv* Befehle oder Menüeintrag aus dem Programm

`Serife` Variable, mathematische Ausdrücke

# Inhaltsverzeichnis

**Zu den Handbüchern** **I**

Inhaltsverzeichnis III

**1 Einführung** **1-1**

1.1 Anwendungsfelder 1-1

1.2 Lösungskonzept 1-1

1.3 Berechnungsumfang 1-2

1.4 Modelica 1-3

1.5 Softwareentwicklung 1-3

**2 Installation** **2-1**

2.1 Überblick und Grundlagen 2-1
2.1.1 Hard- und Softwarevoraussetzungen für SimulationX 2-1
2.1.2 Voraussetzungen für Code-Export und externe Solver 2-1
2.1.2.1 Kompilieren von x64 Code mit Microsoft Visual C++ 2010 Express 2-2

2.2 SimulationX – die Editionen 2-3
2.2.1 Alle Editionen im Überblick 2-3
2.2.2 TEST-Version (Evaluation) 2-4
2.2.3 Die STUDENT Edition 2-4
2.2.4 Die VIEWER Edition 2-5
2.2.5 Die ANALYZER Edition 2-5
2.2.6 Die PROFESSIONAL Edition 2-5
2.2.7 Edition ändern 2-5

2.3 Installation von SimulationX 2-6
2.3.1 Installation 2-6
2.3.2 Erneute Installation bzw. Aktualisierung 2-10

2.4 Erster Programmstart 2-12
2.4.1 SimulationX – die Testversion 2-12
2.4.2 Student- oder Viewer-Edition 2-15
2.4.3 Analyzer- oder Professional-Edition 2-16

2.5 Hinweise für Administratoren 2-16
2.5.1 Installation im Stapelbetrieb (ohne Nutzerinteraktion) 2-16
2.5.2 Installationen zum Programmschutz 2-17
2.5.2.1 Hardwareschutz (Key / Dongle) 2-17

2.5.2.2 Softwareschutz mit Lizenzdatei 2-19

2.6 SimulationX – die Lizenzierung 2-20
2.6.1 Einzelplatzlizenz 2-21
2.6.2 Netzwerklizenz 2-21
2.6.3 Mehrere unterschiedliche Lizenzen im Netzwerk 2-22

2.7 Auswahl eines Hardware-Key / Dongle 2-23
2.7.1 Änderung einer Lizenz 2-25

2.8 Auswahl einer Lizenzdatei 2-26

2.9 Besonderheiten für Netzwerklizenzen 2-29
2.9.1 Netzwerklizenzen mit Dongle 2-29
2.9.2 Netzwerklizenzen mit Lizenzdatei 2-30
2.9.2.1 Lizenzerneuerungsintervall 2-30
2.9.2.2 Verwaiste Lizenzeinträge in der Lizenzdatei 2-32

**3 Erste Schritte 3-1**

3.1 Arbeiten mit Beispielmodellen 3-1
3.1.1 Öffnen von Modellen 3-2
3.1.2 Ändern von Parametern 3-2
3.1.3 Simulation starten 3-3
3.1.4 Ergebnisfenster öffnen 3-4

3.2 Erstellen von eigenen Modellen 3-4
3.2.1 Auswahl der Elemente des Modells 3-4
3.2.2 Verbinden der Elemente 3-6
3.2.3 Parametereingabe 3-6
3.2.4 Simulation im Zeitbereich 3-7
3.2.5 Ergebnisauswertung 3-9
3.2.6 Stationäre Simulation im Frequenzbereich 3-11
3.2.6.1 Stationäre Simulation bei einem Antriebsstrang 3-11

3.3 SimulationX für die ganzheitliche Systemmodellierung und -analyse 3-16

**4 Bedienoberfläche 4-1**

4.1 Übersicht 4-1
4.1.1 Toolfenster 4-1
4.1.2 Dokumentenfenster 4-2

4.2 Bedienung 4-2
4.2.1 Menü- und Symbolleisten 4-2
4.2.2 Bibliotheksleiste 4-3
4.2.2.1 Allgemeine Hinweise zur Bibliotheksleiste 4-6

4.2.3 Struktur- , 3D-, Text- und Dokumentationsansicht 4-10
4.2.4 Modellexplorer 4-12
4.2.4.1 Bearbeiten von Komponenten-Eigenschaften 4-12
4.2.4.2 Hinzufügen von Parametern und Ergebnisgrößen 4-14
4.2.4.3 Löschen von Parametern und Ergebnisgrößen 4-14
4.2.5 Ausgabebereich 4-14
4.2.6 Aufgabenbereich 4-15
4.2.7 Ergebnisfenstermanager 4-15

**5 Modellierung 5-1**

5.1 Modellierungsmöglichkeiten in SimulationX 5-1
5.1.1 Physikalisch-objektorientierte Modellierung 5-1
5.1.2 Signalorientierte Modellierung 5-2
5.1.3 Modellierung mit Gleichungen und Algorithmen 5-3

5.2 Modellerstellung 5-4
5.2.1 Die SimulationX Programmoberfläche 5-4
5.2.2 Modellelemente und Bibliotheken 5-5
5.2.2.1 SimulationX Bibliotheken 5-5
5.2.2.2 Modelica Standard-Bibliothek 5-7
5.2.2.3 Nutzerdefinierte Element-Typen 5-8
5.2.3 Modellerstellung und Modifikation 5-9
5.2.3.1 Aufbau einer Modellstruktur 5-9
5.2.3.2 Editieren von Modellen 5-13
5.2.3.3 Benennung von Elementen und Verbindungen 5-15

5.3 Parametrierung und Ergebnisgrößen 5-17
5.3.1 Eigenschaftsfenster 5-17
5.3.2 Parameter 5-20
5.3.2.1 Parametereingabe 5-21
5.3.2.2 Physikalische Einheiten 5-23
5.3.2.3 Konstanten 5-23
5.3.3 Anfangswerte 5-23
5.3.4 Ergebnisgrößen 5-24
5.3.5 Globale Parameter 5-24

5.4 Weiterführende Themen 5-25
5.4.1 Beschreibung nichtlinearer Eigenschaften über Kraftkennlinie 5-25
5.4.1.1 Beschreibung über Formeln 5-25
5.4.2 Veränderliche Elementparameter 5-28
5.4.2.1 Parametrierung der Elemente mit veränderlichen Parametern - Parameter sind effektive Größen 5-28
5.4.2.2 Integration über veränderliche Parameter-Parameter sind als Ableitungen gegeben 5-29
5.4.3 Arbeiten mit Kennlinien und Kennfelder 5-34

5.4.3.1 Kennlinie 5-34
5.4.3.2 Kennliniensatz 5-45
5.4.3.3 Kurvenschar 5-45
5.4.3.4 2D-Kennfeld 5-46
5.4.3.5 3D-Kennfeld 5-51
5.4.4 3D-Ansicht 5-53
5.4.4.1 Elemente der 3D-Ansicht 5-53
5.4.4.2 Kameramanipulation 5-54
5.4.4.3 Einstellungen 5-55
5.4.4.4 Arbeiten mit der 3D-Ansicht 5-60
5.4.5 Spezielle Modellkomponenten 5-63
5.4.5.1 Einfügen spezieller Modellkomponenten 5-63

**6 Analyse 6-1**

6.1 Transiente Simulation im Zeitbereich 6-1
6.1.1 Simulationssteuerung 6-3
6.1.1.1 Allgemeine Einstellungen 6-3
6.1.1.2 Rücksetzpunkte 6-5
6.1.1.3 Initialisierungszustand speichern 6-6
6.1.1.4 Tracing-Einstellungen 6-7
6.1.1.5 Lösereinstellungen 6-9
6.1.2 Ablauf der Berechnung im Zeitbereich 6-10
6.1.2.1 Die Globale Symbolische Analyse 6-11
6.1.2.2 Berechnung konsistenter Anfangswerte 6-12
6.1.2.3 Einbettung (Homotopie) 6-16
6.1.2.4 Zeitschritte 6-25
6.1.2.5 Ereignisse 6-25
6.1.3 Verfügbare Löser 6-28
6.1.3.1 BDF- und MEBDF-Verfahren 6-29
6.1.3.2 Löser mit konstanter Schrittweite (externer Löser) 6-31
6.1.3.3 CVODE (externer Löser) 6-34
6.1.4 Performanceanalyse der Simulation 6-37
6.1.4.1 Ausgabe 6-37
6.1.4.2 Interpretation der Ergebnisse 6-37
6.1.4.3 Bedienung 6-37

6.2 Animation der Berechnungsergebnisse 6-38
6.2.1 Animation aufzeichnen 6-41
6.2.2 Animation wiedergeben 6-41
6.2.3 3D-Animation für externe Wiedergabe speichern 6-42

6.3 Stationäre Simulation im Frequenzbereich 6-42
6.3.1 Grundlagen der stationären Simulation 6-44
6.3.1.1 Stationär eingeschwungener Zustand eines Systems 6-44
6.3.1.2 Periodenvektor, Periodenvariable und Periodenermittlung 6-45

6.3.1.3 Harmonische Balance, Parametrische Analyse, Bezugsgröße und Ausgleichsparameter 6-46
6.3.2 Der Einstellungsdialog für die stationäre Simulation 6-48
6.3.2.1 Dialogseiten System und Verfahren 6-48
6.3.2.2 Start der stationären Simulation 6-48
6.3.3 System-Seite des Einstellungsdialoges 6-49
6.3.3.1 Festlegung der Bezugsgröße 6-49
6.3.3.2 Festlegung der Periodenvariable und der Periode 6-50
6.3.3.3 Festlegung der Ordnungen 6-51
6.3.3.4 Festlegung des Kompensationsparameters 6-51
6.3.4 Verfahren-Seite des Einstellungsdialoges 6-51
6.3.4.1 Relative Größen 6-52
6.3.4.2 Schrittweitenparameter 6-52
6.3.4.3 Relative Toleranz 6-53
6.3.4.4 Überabtastung 6-53
6.3.4.5 Verfahren 6-54
6.3.5 Ergebnisfenster für die stationäre Simulation 6-58
6.3.5.1 Aktivieren der Protokollattribute der Ergebnisgrößen 6-58
6.3.5.2 Spektren-Darstellung 6-60
6.3.5.3 Verlaufs-Darstellung 6-64

6.4 Gleichgewichtszustand (statisch/stationär) 6-65

6.5 Lineare Systemanalyse 6-67
6.5.1 Eigenfrequenzen und Schwingformen 6-67
6.5.1.1 Eigenwerte 6-68
6.5.1.2 Eigenvektoren 6-70
6.5.1.3 Auslenkung 6-70
6.5.1.4 Energieverteilung 6-72
6.5.1.5 Campbell-Diagramm 6-74
6.5.1.6 Animation 6-75
6.5.2 Übertragungsverhalten 6-76
6.5.2.1 Einstellungen 6-76
6.5.2.2 Frequenzanalyse 6-77
6.5.2.3 Pol- und Nullstellen 6-78
6.5.2.4 Export 6-80

6.6 Variantenrechnung 6-81
6.6.1 Anwendungsbereich 6-81
6.6.2 Vorbereitung 6-81
6.6.3 Der Variantenassistent 6-82
6.6.3.1 Startseite 6-82
6.6.3.2 Parameter 6-83
6.6.3.3 Varianten 6-83
6.6.3.4 Ergebnisse 6-84
6.6.3.5 Speichern 6-84

6.6.4 Beobachtung der Berechnungsergebnisse während der Variantenrechnung 6-88

6.7 Ordnungsanalyse 6-89
6.7.1 Arbeitsweise 6-89
6.7.1.1 Basissignal über Frequenz (Berechnung über der Zeit) 6-89
6.7.1.2 Basissignal über Ordnung (Berechnung über dem Integral des Basissignals) 6-90
6.7.1.3 Ordnung über Zeit 6-92
6.7.2 Bedienung 6-92
6.7.2.1 Import der Quelldaten 6-93
6.7.2.2 Einstellen der Analyseoptionen 6-96
6.7.2.3 Einstellung der Anzeigeoptionen 6-101
6.7.2.4 Speichern, Drucken, Export der Ergebnisse 6-103

6.8 Leistungsbilanz 6-104
6.8.1 Allgemeine Funktionsweise 6-104
6.8.2 Erstellung einer Leistungsbilanz in SimulationX 6-106
6.8.2.1 Allgemeine Vorgehensweise 6-106
6.8.2.2 Vorgehensweise bei Modellen mit Compounds 6-108
6.8.3 Leistungssensoren 6-109

6.9 Troubleshooting: Was tue ich, wenn...? 6-110
6.9.1 Typische Modellierungsprobleme 6-110
6.9.2 Es können keine konsistenten Anfangswerte berechnet werden 6-112
6.9.3 Das System ist singulär 6-113
6.9.4 Langsame Simulation 6-114
6.9.5 Allgemeine Hinweise 6-116

**7 Ergebnisauswertung, Dokumentation und allgemeine Einstellungen 7-1**

7.1 Übersicht 7-1

7.2 Momentananzeige 7-1
7.2.1 Momentananzeige öffnen 7-2
7.2.2 Verknüpfung mit Ergebnisgrößen 7-2
7.2.3 Weitere Einstellungen 7-3
7.2.3.1 Linien und Farben 7-3
7.2.3.2 Größe und Position 7-3
7.2.4 Anmerkungen 7-3

7.3 Ergebnisfenster 7-4
7.3.1 Grundlagen 7-4
7.3.2 Ergebnisfenster öffnen 7-5
7.3.2.1 Leere Ergebnisfenster 7-5
7.3.2.2 Kontextmenü (Strukturansicht) 7-5

7.3.2.3 Modell-Explorer 7-6
7.3.2.4 Eigenschaftsdialog (Modellelemente und Verbindungen) 7-6
7.3.2.5 Ergebnisleiste (Ergebnisfenster) 7-7
7.3.2.6 Ergebnisfenster-Manager 7-8
7.3.2.7 COM-Interface 7-8
7.3.3 Aufbau von Ergebnisfenstern 7-8
7.3.3.1 Begriffe 7-8
7.3.3.2 Fensterlayout 7-9
7.3.4 Verschieben und Kopieren 7-14
7.3.4.1 Verschieben und Kopieren von Verläufen 7-14
7.3.4.2 Verschieben und Kopieren von Panels 7-14
7.3.4.3 Verschieben und Kopieren von Seiten 7-16
7.3.5 Anpassen der Ansicht 7-16
7.3.5.1 Achsenanpassung und Skalierung 7-16
7.3.5.2 Hilfslinien 7-19
7.3.5.3 Ändern von Beschriftungen 7-20
7.3.6 Post-Processing 7-22
7.3.6.1 Transformationen 7-22
7.3.6.2 Verknüpfungen 7-22
7.3.6.3 Datenmodell und interne Struktur 7-23
7.3.6.4 Verkettungen von Transformationen und Operationen 7-26
7.3.6.5 Ausmessen von Verläufen 7-27
7.3.7 Einfrieren und Import von Ergebnisverläufen 7-30
7.3.8 Ergebnisse und Druck 7-31
7.3.8.1 Einstellungen - Die Eigenschaftsleiste 7-32
7.3.8.2 Laden und Speichern von Einstellungen 7-44
7.3.8.3 Festlegen der Standardeinstellung 7-45
7.3.9 Verwaltung von Ergebnisfenstern 7-45
7.3.9.1 Zusammenführen von Ergebnisfenstern 7-45
7.3.9.2 Der Ergebnisfenstermanager 7-45
7.3.9.3 Frequenzanalyse FFT 7-47
7.3.9.4 Komplexer Übertragungsfrequenzgang (FRF) 7-48

7.4 Drucken (PrintEngine) 7-49
7.4.1 Seitenansicht 7-49
7.4.1.1 Auswahlleiste 7-49
7.4.1.2 Toolbar 7-50
7.4.1.3 Seite einrichten 7-51
7.4.2 Benutzerdefinierter Druck 7-52

7.5 Schnittstellen zu MS-Office, Import/Export 7-55

7.6 Einstellungen zu SimulationX 7-55
7.6.1 Allgemeine Einstellungen 7-56
7.6.2 Bibliotheken und Erweiterungen 7-56
7.6.3 Verzeichnisse 7-57

7.6.4 Speichern 7-57
7.6.5 Modelica 7-59
7.6.6 Fluide 7-59
7.6.7 Sprachen 7-60
7.6.8 Lizenzierung 7-60
7.6.9 Programmanpassung 7-60

**8 Erweiterte Modellierung 8-1**

8.1 Verwaltung von Externen Typen in SimulationX 8-1

8.2 TypeDesigner 8-7
8.2.1 Aufruf und Oberfläche 8-8
8.2.2 Beschreibung der Funktionen des TypeDesigner 8-9
8.2.2.1 Allgemein 8-9
8.2.2.2 Anschlüsse 8-10
8.2.2.3 Basistypen 8-12
8.2.2.4 Komponenten 8-12
8.2.2.5 Lokale Typen 8-16
8.2.2.6 Import 8-18
8.2.2.7 Activity Groups 8-19
8.2.2.8 Verhalten 8-22
8.2.2.9 Modelica Code 8-24
8.2.2.10 Dokumentation 8-25
8.2.2.11 Sicherheit 8-26
8.2.3 Ableiten von Typen (Basistypen) 8-28
8.2.4 Zusammenfassen von Teilmodellen (Compound) 8-29
8.2.4.1 Innere Anschlüsse 8-30
8.2.4.2 Innere Komponenten 8-31
8.2.5 Anlegen von Bibliotheken oder Packages (SimulationX) 8-34
8.2.6 Funktionen 8-34
8.2.6.1 Interne/Externe Funktionen 8-34
8.2.6.2 Partielle Ableitungsfunktionen 8-34
8.2.6.3 Beispiel Externe Funktion 8-35
8.2.7 Simulationsspezifische Funktionen 8-42

8.3 FluidDesigner 8-47
8.3.1 FluidDesigner *Hydraulik* 8-47
8.3.1.1 Allgemein 8-49
8.3.1.2 Viskosität 8-50
8.3.1.3 Dichte 8-52
8.3.1.4 Kompressibilität 8-52
8.3.1.5 Wärmedehnung 8-52
8.3.1.6 Spezifische Wärmekapazität 8-52
8.3.1.7 Wärmeleitfähigkeit 8-53
8.3.1.8 Luftaufnahme 8-53

8.3.1.9 Dampfdruck 8-53
8.3.1.10 Grenzwerte 8-53
8.3.2 FluidDesigner Pneumatik – Gase 8-53
8.3.2.1 Allgemein 8-53
8.3.2.2 Viskosität 8-54
8.3.2.3 Kalorische Eigenschaften 8-54
8.3.2.4 Wärmeleitfähigkeit 8-54
8.3.2.5 Spezifische Enthalpie 8-54
8.3.2.6 Realgas 8-54
8.3.2.7 Grenzwerte 8-54
8.3.3 FluidDesigner *Mischungen* 8-54

8.4 Statechart Designer 8-55
8.4.1 Einführung in die Arbeit mit Statecharts 8-55
8.4.2 Grafische Notation von Zustandsmaschinen 8-56
8.4.2.1 Zustände / States 8-56
8.4.2.2 Transitionen / Transitions 8-57
8.4.2.3 Pseudozustände / Pseudostates 8-58
8.4.2.4 Trigger 8-60
8.4.2.5 Guards 8-61
8.4.2.6 Prioritäten 8-62
8.4.2.7 Self-Transitionen und interne Transitionen 8-62
8.4.2.8 Zusammengesetzte Zustände / Composite States 8-63
8.4.3 Statechart Designer in SimulationX 8-67
8.4.3.1 Aufruf und Oberfläche 8-67
8.4.3.2 Modellierung 8-68
8.4.3.3 Animierte Darstellung des Statechart in SimulationX 8-74
8.4.4 Semantik der Notationselemente in Modelica-Code 8-76
8.4.4.1 Zustände 8-76
8.4.4.2 Transitionen 8-78
8.4.4.3 Signal-Trigger 8-79
8.4.4.4 Change-Trigger 8-81
8.4.4.5 Time-Trigger 8-82
8.4.4.6 Guards 8-83
8.4.4.7 Prioritäten 8-86
8.4.4.8 Transition-Activities 8-87
8.4.4.9 Zusammengesetzte Zustände 8-88
8.4.4.10 Gesamtstruktur der Modelica-Codierung 8-95

**9 Schnittstellen 9-1**

9.1 Datenformate 9-1
9.1.1 Textformat 9-1
9.1.1.1 Datei speichern 9-2
9.1.1.2 Datei laden 9-2
9.1.2 IEEE-Binärformat 9-4

9.1.2.1 Datei speichern 9-4
9.1.2.2 Datei laden 9-5
9.1.3 ITI-Binärformat (*.rfs) und ITI-ASCII-Format (*.rfu) 9-7
9.1.4 XML-Format 9-7
9.1.5 DIAdem-Headerformat 9-7

9.2 COM Interface 9-8
9.2.1 Klassen 9-9
9.2.1.1 Application 9-9
9.2.1.2 Documents 9-9
9.2.1.3 SimObjects 9-10
9.2.1.4 Connections 9-11
9.2.1.5 Parameters 9-11
9.2.1.6 Curves 9-12
9.2.1.7 Results 9-12
9.2.1.8 Selection 9-14
9.2.1.9 ResultWindows 9-14
9.2.1.10 NaturalFrequencies 9-15
9.2.2 Beispiele 9-17
9.2.2.1 VBScript-Beispiel 9-18
9.2.2.2 JScript-Beispiel 9-18
9.2.2.3 ITI-Script-Beispiel 9-19
9.2.3 Code-Export über die COM - Schnittstelle 9-20

9.3 Co-Simulation 9-23
9.3.1 Co-Simulation mit SimulationX Koppelelement 9-23
9.3.1.1 Beschreibung des Kommunikations-Protokolls 9-23
9.3.1.2 Nutzung der Koppel-Elemente in SimulationX 9-26
9.3.2 Co-Simulation mit Simulink 9-27
9.3.3 Simulatorkopplung mit ADAMS 9-28
9.3.3.1 Einführung 9-28
9.3.3.2 Voraussetzungen und Installation 9-29
9.3.3.3 Einrichten der Kopplung in SimulationX und ADAMS 9-29
9.3.4 Co-Simulation mit CarSim, BikeSim, TruckSim 9-38
9.3.4.1 Einführung 9-38
9.3.4.2 Workflow 9-39
9.3.4.3 Beispielmodell 9-39
9.3.4.4 Vorbereitung des CarSim-Modells 9-40
9.3.4.5 Vorbereitung des SimulationX-Modells 9-42
9.3.4.6 Co-Simulation 9-46
9.3.4.7 Post processing in CarSim and SimulationX 9-48

9.4 Code-Export 9-49
9.4.1 Einführung 9-49
9.4.2 Code-Export Assistent 9-52
9.4.2.1 Dialogseite Projekt 9-52

9.4.2.2 Auswahl der Signaleingänge 9-53
9.4.2.3 Auswahl der Ausgänge 9-55
9.4.2.4 Auswahl der Parameter 9-55
9.4.2.5 Auswahl der Kennlinien (nur dSPACE DS1006) 9-56
9.4.2.6 Task Handling (nur dSPACE DS1006) 9-56
9.4.2.7 Zusätzliche Bibliotheken 9-56
9.4.2.8 Code-Generierung 9-57
9.4.2.9 Nachbereitung 9-58
9.4.3 Projekttypen 9-59
9.4.3.1 C-Code (ohne Solver) 9-59
9.4.3.2 Ausführbares Modell 9-59
9.4.3.3 Simulink S-Function 9-66
9.4.3.4 SIMPACK 9-70
9.4.3.5 SCALE-RT 9-73
9.4.3.6 Externes Modell 9-74
9.4.3.7 dSPACE DS1006 9-75
9.4.3.8 NI VeriStand 9-77
9.4.3.9 NI LabView Control Design & Simulation Module 9-78
9.4.3.10 Modelisar FMI für Model Exchange 9-81
9.4.3.11 Modelisar FMI für Co-Simulation 9-84
9.4.3.12 CarSim, BikeSim, TruckSim Component (VehicleSim) 9-87

9.5 Code Import 9-90
9.5.1 FMU Code Import 9-90
9.5.2 Code Import aus Simulink/RTW 9-92
9.5.2.1 Installation 9-93
9.5.2.2 Erzeugen und Verwenden der RTW-DLL 9-93

9.6 Schnittstellen zu anderen Tools 9-94
9.6.1 Optimierung mit OptiY 9-94
9.6.2 Optimierung mit Isight (optional) 9-95
9.6.2.1 Komponenteneditor 9-95
9.6.2.2 Mapping von Parametern und Ergebnisgrößen 9-96
9.6.3 Optimierung mit modeFRONTIER® 9-96
9.6.4 FEM-Import 9-97

9.7 Datenbankanbindung 9-97
9.7.1 Einfügen der Datenbankanbindung 9-98
9.7.2 Automatische Erzeugung einer Datenbank 9-99
9.7.3 Verbinden mit vorhandener Datenbank 9-101
9.7.3.1 Auswahl Datenbank 9-102
9.7.3.2 Auswahl Tabelle 9-104
9.7.3.3 Binden der Parameter 9-106
9.7.4 Parametrieren 9-107
9.7.5 Löschen von Bindungen 9-109
9.7.6 Binden von Kennlinien und Kennfeldern 9-109

9.7.6.1 Kennlinien 9-109
9.7.6.2 Kennfelder 9-111
9.7.7 Hierarchische Datenbankverbindungen 9-113

**10 Modelica® in SimulationX 10-1**

10.1 Allgemeine Hinweise 10-1

10.2 Modellierung mit der Modelica-Standard-Bibliothek 10-1
10.2.1 Installation 10-1

10.3 Kombinieren der SimulationX-Bibliotheken mit der Standard-Modelica-Bibliothek 10-4

10.4 Besonderheiten für SimulationX 10-5
10.4.1 Lexikalische Struktur und Syntax 10-5
10.4.2 Operatoren und Ausdrücke 10-6
10.4.2.1 Arithmetik-Operatoren 10-6
10.4.2.2 Vergleichs- und logische Operatoren 10-7
10.4.2.3 Einfache Mathematische Funktionen 10-7
10.4.3 Klassen, Typen und Deklarationen 10-8
10.4.3.1 Komponentendeklaration 10-8
10.4.3.2 Attribute vordefinierter Typen 10-8
10.4.4 Namensräume und Bezeichner 10-11
10.4.4.1 Referenzen auf lokale Parameter 10-11
10.4.5 Vererbung, Modifikation und Redeklaration 10-11
10.4.6 Gleichungen 10-11
10.4.6.1 Die for-Anweisung 10-12
10.4.6.2 If-Anweisung 10-14
10.4.6.3 Die when-Anweisung 10-14
10.4.7 Konnektoren und connect-Gleichungen 10-15
10.4.8 Arrays 10-15
10.4.9 Algorithmen 10-16
10.4.9.1 Zuweisungen 10-16
10.4.9.2 Die for-Anweisung 10-17
10.4.9.3 If-Anweisung 10-17
10.4.9.4 Die when-Anweisung 10-18
10.4.9.5 Die while-Anweisung 10-19
10.4.10 Funktionen 10-19
10.4.11 Bibliotheken (Packages) 10-19
10.4.12 Annotationen 10-19

**11 FAQ – Häufig gestellte Fragen 11-1**

11.1 Allgemein 11-1

11.2 Zur Installation/Lizenzierung 11-2

11.3 Zur Simulation/Berechnung 11-4
11.3.1 Interne Löser (BDF und MEBDF) 11-4
11.3.2 Externe Löser (Löser mit konstanter Schrittweite und CVODE) 11-27
11.3.2.1 Warnungen, die im Zusammenhang mit nichtlösbaren Blöcken auftreten 11-27

**12 Begriffsdefinitionen 12-1**

**13 Literaturverzeichnis 13-1**

**14 Index 14-1**

SIMULATION X®

Powered by ITI



# Introduction

# 1 Einführung

## 1.1 Anwendungsfelder

SimulationX ist ein Programmpaket für die *Modellierung, Berechnung, Simulation, Analyse, Optimierung und Zuverlässigkeitsanalyse* von Komponenten und Systemen in allen Bereichen technischer Entwicklungen. Die Berechnungsgrundlagen aller wichtigen physikalischen Domänen, wie *Mechanik, Fluidtechnik, Thermik, Regelungs- und Signaltechnik, Elektronik und Magnetik, Elektro- und Energietechnik*, sind in umfangreichen Bibliotheken verfügbar.

Mit dieser Software stehen Ihnen leistungsfähige Modellierungsmittel zur Verfügung, um alle physikalisch-technischen Zusammenhänge abzubilden. Das Tool ist somit universell einsetzbar und offen für neue Anwendungsgebiete. Darin widerspiegelt sich der Anspruch des ITI-Teams, Ihnen alle Voraussetzungen und Werkzeuge in die Hand zu geben, mit denen Sie Ihr spezielles Know-how in zugeschnittene Modelle für Ihre Aufgaben umsetzen können.

Zentrale *Anwendungsfelder*, auf denen mehr als zwanzigjährige Erfahrung in Modellierung und Simulation zur Verfügung steht, sind die Fahrzeugtechnik und die Antriebs- und Fluidtechnik in allen Industriebranchen. Das Programm ist im Maschinen- und Anlagenbau, im Schiffs- und Schienenfahrzeugbau und der Luft- und Raumfahrt zuhause.

## 1.2 Lösungskonzept

Das Programmpaket unterstützt den *durchgängigen Entwurfsprozess* sowohl horizontal als auch vertikal: Indem die Modellierung und Berechnung von Teilmodellen unterschiedlicher Art in einer einheitlichen Umgebung integriert ist, wird die umfassende Systemsimulation komplexer Komponenten, Maschinen, Fahrzeuge und Anlagen möglich. Universelle Datenschnittstellen, COM-Programmierung, C-Codegenerierung und Kosimulationsmodule gewährleisten die optimale Weiterverwendung von Berechnungsdaten und die Verbindung zu CAD und CAE-Tools.

SimulationX vereinigt *Modellierungskonzepte* auf verschiedenen Niveaus. Zum einen kann man mit validierten Standardelementen aus den Bibliotheken des Lieferumfanges schnell und effizient Modelle aufbauen. Zum anderen steht versierten Modellentwicklern eine leistungsfähige Entwicklungsumgebung mit dem komfortablen Werkzeug *TypeDesigner* für den Aufbau nutzerdefinierter Elemente zur Verfügung. Dabei wird die Effizienz und Sicherheit der Entwicklung durch eine objektorientierte Modellierungssprache Modelica® gefördert. Im Sprachumfang von Modelica findet der Modellierer einen großen Freiraum für die Beschreibung seiner Systeme und Prozesse.

Die Komplexität der Modellbeschreibung setzt hohe Maßstäbe für die Lösung der Berechnungsaufgaben während der Simulation. In SimulationX ist dazu eine Reihe von analytischen und numerischen Verfahren implementiert, die auch für anspruchsvolle Modelle eine hohe Performance ergeben. In mehreren Compilerpässen wird das System aus algebraischen und Differenzialgleichungen (DAE) einer globalen symbolischen Analyse, Reduktion und Vereinfachung unterzogen. Das daraus entstehende kompakte Gleichungssystem wird an einen leistungsfähigen Solver übergeben. Standard ist hier das BDF-Verfahren, das auch für steife Systeme, wie sie bei heterogenen Modellen oft entstehen, eine gute Performance ergibt.

Ab Version 3 steht Anwendern zudem eine Implementation von CVODE zur Verfügung. Mit diesem explizitem Löser sind insbesondere bei Modellen, die auf gewöhnliche Differentialgleichungen führen, Rechenzeiteinsparungen möglich. Wenn Sie für Ihre Simulationen dieses Verfahren auswählen, profitieren Sie außerdem davon, dass Modell und Solver separat vor der Laufzeit kompiliert werden. Das bringt eine weitere, und zwar modellunabhängige Performance-steigerung.

Die in der industriellen Praxis zu untersuchenden Anwendungen enthalten fast immer Nichtlinearitäten und insbesondere Unstetigkeiten. SimulationX kann Unstetigkeiten mit einstellbarer Genauigkeit berücksichtigen.

Mit Datenbanken für Parameter, Modelle und Varianten sowie den dafür optimierten Schnittstellen wird der Übergang von der Grundlagenuntersuchung der Systeme zur projektierenden Arbeitsweise mit Daten realer Komponenten vollzogen.

## 1.3 Berechnungsumfang

Mit dem vorliegenden Programm werden Systeme untersucht, die als Netzwerk von (räumlich) konzentrierten Elementen beschrieben werden. Die domänen-spezifisch gültigen Umlauf- und Schnittgesetze für Differenz- und Flussgrößen sind Bestandteil des Systems. Außerdem wird die Modellierung auf der Basis von Signalflussplänen unterstützt. In der Notation der SimulationX-Modelle finden sich sequentielle Algorithmen, (algebraische) Gleichungen, (gewöhnliche) Differenzialgleichungen, analytische oder empirische (Kennlinien), Funktionen und logische Bedingungen für Zustandsübergänge. Damit lassen sich natürlich auch lineare Systeme und einfache Steuerungen abbilden. Die Vorteile des Programms zeigen sich jedoch vor allem bei der Untersuchung der *Dynamik von nichtlinearen Systemen*, bei Diskontinuitäten, im Zusammenspiel mit Regelungen und in der Berücksichtigung externer Messdaten und Kennfelder.

Ungeachtet der ausgefeilten Modellierungstools ist Modellentwicklung nach wie vor eine anspruchsvolle Arbeit, die Kreativität und Konzentration verlangt. Umso

mehr ist Wert darauf zu legen, dass ein einmal erstelltes Modell eines Systems auch gleich für verschiedene und verschiedenartige Analysen und Berechnungen genutzt werden kann. Welche Berechnungsaufgaben können mit SimulationX gelöst werden?

Mit der *Berechnung im Zeitbereich* (transiente Simulation) wird des gesamte Verhalten eines System innerhalb eines Zeitintervalls untersucht. Anregungen des Systems und vom Nutzer vorgegebene Anfangswerte werden dabei berücksichtigt.

Oft muss ein *Gleichgewichtszustand* des Systems ermittelt werden, der dann Ausgangspunkt für weitere Untersuchungen ist. Fachspezifisch spricht man vom statischen oder stationären Zustand.

In jedem Betriebspunkt kann SimulationX eine Strukturanalyse des linearisierten Systems ausführen und *dabei Eigenfrequenzen, Eigenvektoren, Frequenzgänge und Übertragungsfunktionen* berechnen.

Für die *Stationäre Simulation,* beispielsweise von Drehschwingungsanwendungen hat ITI bisher schon das Tool ITI-STAT angeboten. Jetzt ist ein noch leistungsfähigeres und universelles Analyseverfahren entwickelt und als Option in SimulationX bereitgestellt worden.

Auf der Basis der funktionalen Struktur des Modells und zusätzlicher lokaler Fehlerwahrscheinlichkeiten kann SimulationX ein Modell zur *Zuverlässigkeitsanalyse* generieren.

Wegen der einheitlichen Modellbasis lassen sich die Analysearten sinnvoll kombinieren. Darüber hinaus können umfangreiche Parameterstudien mit dem *Variantenassistenten* effektiviert werden.

## 1.4 Modelica

Nachdem für die Modellbeschreibung in den SimulationX-Bibliotheken bisher schon die Sprache Modelica® eingesetzt wurde, unterstützt SimulationX den vollen Sprachumfang von Modelica. Als Nutzer von SimulationX stehen Ihnen nun zusätzlich zum bisherigen Leistungsumfang auch die Modelica-Standardbibliotheken zur Verfügung. Für komplexe Neuentwicklungen von Modellen können Sie als Modellierungsexperte auf die umfangreichen Sprachkonstrukte, wie Arrays und Records zurückgreifen.

## 1.5 Softwareentwicklung

Das Programmpaket wurde mit modernen softwaretechnologischen Methoden entwickelt. Mit der bei ITI entwickelten, umfangreichen Klassenbibliothek, hauptsächlich auf Basis von C++, sind unsere Entwickler in der Lage, flexibel auf

die Anforderungen der Anwender zu reagieren. Qualitätssicherung war und ist integraler Bestandteil der Entwicklung von SimulationX.

Die Modelle der Standardbibliotheken wurden einzeln und im Zusammenspiel innerhalb komplexer Anwendungen getestet. Die Lösungsverfahren wurden im Rahmen des Tests extremen Anforderungen ausgesetzt. Da die Aufgaben der Anwender immer anspruchsvoller werden, verstehen wir die Perfektionierung unserer Software als ständige Aufgabe. Deshalb freuen wir uns über Ihre Erfahrungen, Kritik und Anregungen. Nutzen Sie unseren Service, wir helfen Ihnen gern bei der Lösung Ihrer Aufgaben. Über das Internet erfahren Sie alles über unseren HelpDesk, Workshops, Seminare und Schulungen.

ITI wünscht Ihnen viel Erfolg bei der Arbeit mit dem neuen SimulationX und Spaß beim Simulieren.

Dresden, 2012

SIMULATION X®

Powered by ITI

Installation

# Installation

# 2 Installation

## 2.1 Überblick und Grundlagen

Das Programm wurde für die aktuelle Generation von Windows-Betriebssystemen entwickelt. Es ist ab Windows XP (ab Service Pack 2) und neueren Windows-Versionen lauffähig. Die 64-bit-Versionen der Betriebssysteme werden unterstützt.

Zur Installation von SimulationX benötigen Sie die **Zugriffsrechte eines Administrators**. Sofern Sie nicht selbst über die nötigen Zugriffsrechte verfügen, lassen Sie die Installation von Ihrem Systemadministrator durchführen.

### 2.1.1 Hard- und Softwarevoraussetzungen für SimulationX

Damit SimulationX auf Ihrem System lauffähig ist, sind bestimmte Minimal-anforderungen zu erfüllen, siehe **Tabelle 2.1**.

| Komponente | Minimalanforderung |
|---|---|
| Betriebssystem | Windows 7, Windows Vista or Windows XP for 32 or 64 bit systems* |
| Hauptspeicher (RAM) | 1 GB RAM (2 GB on x64 systems) ** |
| Festplattenspeicher | minimal 200 MB |
| Grafik | Grafikkarte mit 3D Hardwarebeschleunigung und DirectX 9.0c mit einer Auflösung (1280 x 1024) oder besser |
| Laufwerke | CD-ROM Laufwerk zur Installation |
| * Die Installation der neuesten Service-Packs wird empfohlen<br>** Der effektiv benötigte Arbeitsspeicher ist von der Modellgröße und Modellkomplexität abhängig | |

**Tabelle 2.1:** Hard- und Softwarevoraussetzungen

Fall Sie sich einen neuen PC anschaffen wollen, empfiehlt ITI ein System mit den jeweils aktuellen Leistungsparametern für Aufgaben im CAE-Bereich. Prozessor-geschwindigkeit und Hauptspeicherausbau sind dabei die bestimmenden Kenn-größen. Mehrprozessorarchitekturen werden voll unterstützt. Zwar wird für die Simulationsrechnung jeweils nur ein Prozessorkern verwendet, jedoch stehen die übrigen Kerne für die Visualisierung (Ergebnisfensteraktualisierung und 3D-Animation) sowie für andere laufende Anwendungen zur Verfügung.

### 2.1.2 Voraussetzungen für Code-Export und externe Solver

Für die Erstellung von ausführbarem Code beim Code-Export (Executable Models u.a.) sowie für die Simulationsrechnung mit externen Solvern (CVODE,

FixedStep) wird ein Compiler benötigt. Die von SimulationX unterstützten Entwicklungsumgebungen entnehmen Sie bitte der folgenden Tabelle.

| Entwicklungsumgebung | SimulationX 32bit | | | SimulationX 64bit | | |
|---|---|---|---|---|---|---|
| | Ext. Solver | CE für x86 | CE für x64 | Ext. Solver | CE für x86 | CE für x64 |
| Microsoft Visual C/C++ 6.0 | x | x | | | x | |
| Microsoft Visual C/C++ .Net | x | x | | | x | |
| Microsoft Visual C/C++ .Net 2003 | x | x | | | x | |
| Microsoft Visual C/C++ 2005 | x | x | x | x | x | x |
| Microsoft Visual C/C++ Toolkit 2003[1] | x | x | | | x | |
| Microsoft Visual C/C++ 2008 | x | x | x | x | x | x |
| Microsoft Visual C/C++ 2008 Express Edition[1] | x | x | | | x | |
| Microsoft Visual C/C++ 2010 | x | x | x | x | x | x |
| Microsoft Visual C/C++ 2010 Express Edition[2] | x | x | x[3] | x[3] | x | x[3] |

[1] Kostenlos, aber nicht mehr erhältlich
[2] Kostenlos erhältlich über www.microsoft.com
[3] vorausgesetzt, die 64-bit Unterstützung wurde nachinstalliert (siehe Kapitel 2.1.2.1)

**Tabelle 2.2:** Voraussetzungen für externe Solver und Erstellung beim Code-Export

### 2.1.2.1 Kompilieren von x64 Code mit Microsoft Visual C++ 2010 Express

Mit der Standardinstallation von Microsoft Visual C++ 2010 Express können nur 32-bit Applikationen kompiliert werden. Die Unterstützung von 64-bit Applikationen ist nicht direkt beinhaltet, kann aber mit Hilfe der folgenden Schritte hinzugefügt werden.

- Als Erstes muss das Windows SDK Version 7.1 oder höher installiert werden.
- Nach dem das SDK installiert worden ist, muss es mit dem "Windows SDK Configuration Tool" aus dem Startmenü (Microsoft Windows SDK v7.1. → Visual Studio Registration) aktiv geschalten werden. Die grafische Bedienoberfläche des Tools enthält einen Fehler und sollte deshalb nicht benutzt werden. Stattdessen ist ein neues Kommandozeilenfenster in dem Ordner zu öffnen, der die Datei

"WindowsSdkVer.exe" enthält (auf diese Datei verweist die Verknüpfung im Startmenü), und das folgende Kommando auszuführen:

```
WindowsSdkVer.exe -version:v7.1
```

Damit ist das Windows SDK V7.1 aktiv.

- Falls das "Visual Studio 2010 SP1" installiert wurde oder die Windows Updates dieses installiert haben, wurden damit die x64 Compiler gelöscht. Um diese wiederherzustellen, muss das im Knowledgebase-Artikel KB2519277 bereitgestellte Packet installiert werden.

Mit dieser Microsoft Visual C++ 2010 Express Installation kann nun x64 Code erzeugt werden.

## 2.2 SimulationX – die Editionen

### 2.2.1 Alle Editionen im Überblick

| | Edition | | | |
|---|---|---|---|---|
| **Programmfunktion** | **Student** | **Viewer** | **Analyzer** | **Professional** |
| Datei neu | x | | | x |
| Datei öffnen | x | | x | x |
| Datei öffnen (verschlüsselt) | x | x | x | x |
| Datei speichern | x | | x | x |
| Datei speichern (verschlüsselt) | | x | x | x |
| Seitenansicht | | x | x | x |
| Drucken | x | x | x | x |
| Externe Typen laden | x | x | x | x |
| Externe Typen erzeugen/bearbeiten | x | | | x |
| Modelle bearbeiten (Struktur) | x | | | x |
| Modellparameter ändern | x | | x | x |
| Steuerelemente / Momentananzeigen | x | | | x |
| Datenbankanbindung | x | | Lizenz | Lizenz |

| | | | | |
|---|---|---|---|---|
| Transiente Simulation | x | x | Lizenz | Lizenz |
| 3D Ansicht | x | x | Lizenz | Lizenz |
| Scripteditor | x | | | x |

**Tabelle 2.3:** In den Editionen verfügbare (x) Programmfunktionen

Es gibt eine SimulationX Produkt-CD **für alle Editionen**. Die jeweilige Edition wird durch Auswahl **beim ersten Start** von SimulationX aktiviert.

### 2.2.2 TEST-Version (Evaluation)

SimulationX steht als voll funktionsfähige ***Professional Edition*** zum **Test** zur Verfügung. Diese Version beinhaltet Bibliotheken und Optionen, welche in zeitlich begrenztem Umfang aber mit KOMPLETTER FUNKTIONALITÄT getestet werden können.
Sie benötigen dazu lediglich eine zeitlich befristete und für Ihren Rechner erzeugte Lizenzdatei, welche Ihnen auf Anforderung (siehe auch **2.4**) von ITI zugesandt wird.

### 2.2.3 Die STUDENT Edition

Die Student Edition ermöglicht das Bearbeiten von Aufgaben im Rahmen des Studiums. Der eingeschränkte Funktionsumfang bezieht sich auf die Programm-funktionen (s. **Tabelle 2.2**) und die Festlegung, dass nur bestimmte Elementtypen in einer begrenzten Anzahl in einem Modell verwendet werden dürfen. Eine Aufzählung, welche Elemente wie oft benutzt werden können, findet man im ***Optionen***-Dialog auf der Seite ***Lizenzierung*** (s. **Bild 2.1**).

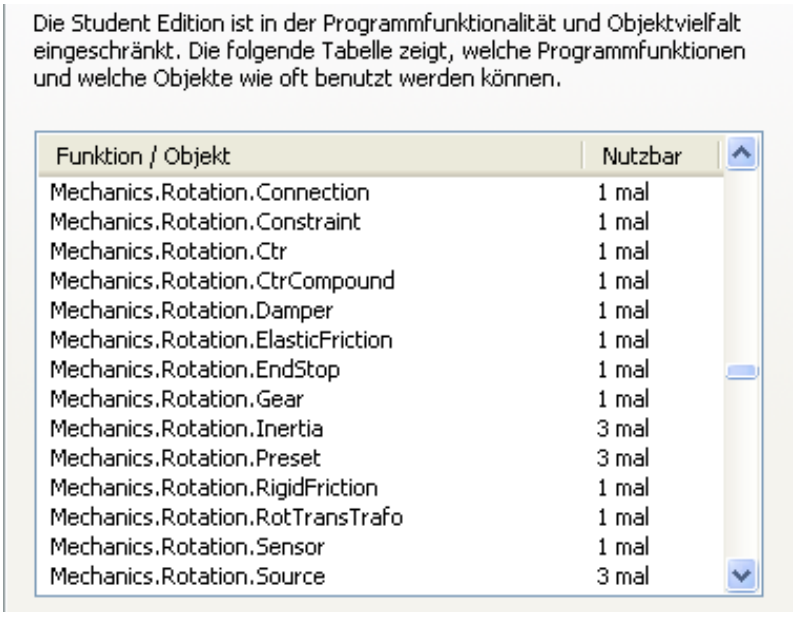



**Bild 2.1:** In der Student Edition nutzbare Elemente (Ausschnitt)

### 2.2.4 Die VIEWER Edition

Modelle können **verschlüsselt** mit der Viewer Edition an Kunden und Interessenten weitergegeben werden. Diese Modelle können mit der Viewer Edition geöffnet und ggf. berechnet werden. Ergebnisse kann man ausdrucken (s. **Tabelle 2.2**).

### 2.2.5 Die ANALYZER Edition

Auf der Basis vorbereiteter Modelldateien eignet sich die Analyzer Edition für Parameterstudien. Für diese Version benötigen Sie eine Lizenz.

### 2.2.6 Die PROFESSIONAL Edition

Mit der Professional Edition steht Ihnen ein mächtiges Werkzeug zur Modell-erstellung, Simulation und Analyse zur Verfügung.
Sie können alle erworbenen Programmfunktionen, Bibliotheken und Optionen uneingeschränkt nutzen.
Für diese Version benötigen Sie eine Lizenz.

### 2.2.7 Edition ändern

Die Edition kann jederzeit gewechselt werden. Dazu ist im Programm im ***Optionen***-Dialog die Seite ***Lizenzierung*** zu wählen (s. **Bild 2.2**).

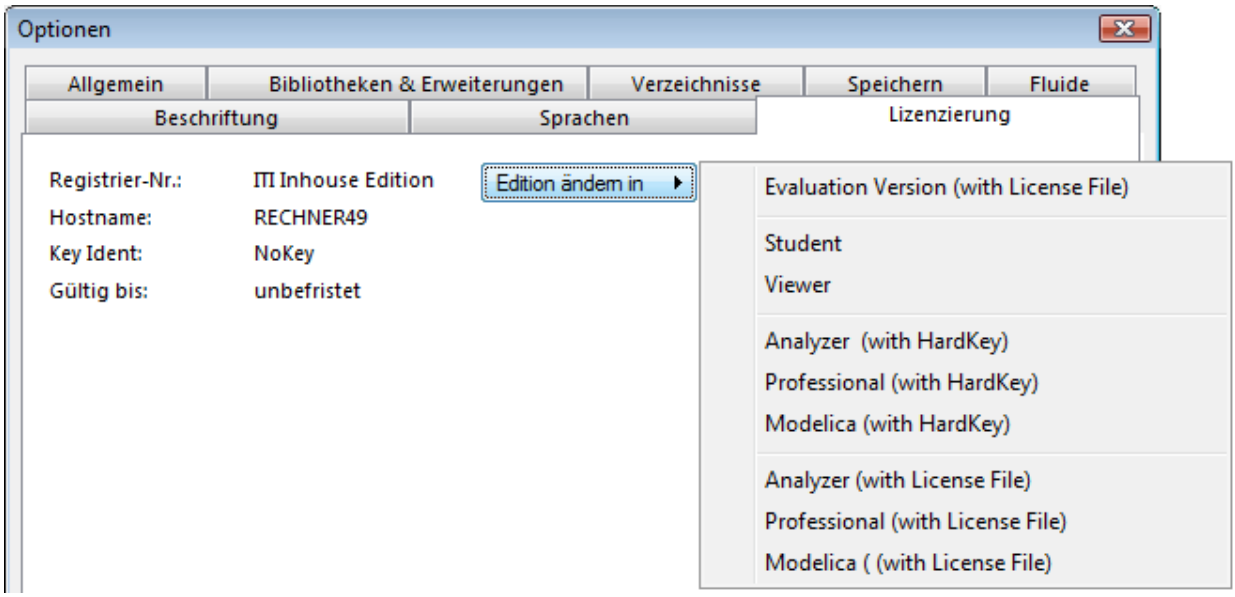



**Bild 2.2:** Umschalten der Edition in SimulationX

Klicken Sie auf ***Edition ändern in*** und wählen Sie eine Edition für den nächsten Programmstart aus. Zu beachten ist, dass für die Evaluation, Analyzer und Professional Editionen eine entsprechende Lizenz vorhanden sein muss.
**Tabelle 2.3** gibt einen Überblick über die in den einzelnen Editionen verfügbaren Programmfunktionen.

## 2.3 Installation von SimulationX

Nach Einlegen der CD meldet sich das Startprogramm, das Sie durch alle notwendigen Installationsschritte führt. Sollte die Autorun-Funktion Ihres CD-Laufwerks ausgeschaltet sein, starten Sie bitte das Programm "Autorun.exe" im CD-Hauptverzeichnis. Es wird ein Auswahldialog angezeigt, der die Sprache des Installationsassistenten zur Auswahl anbietet.

Auf der „Willkommen“-Seite werden neben der Software Installation noch Informationen, Dokumentationen, die ITI Vertragsbedingungen und das Identifizieren des Computers angeboten.

Zuerst wählen Sie ***Software installieren***.

Über die weiteren Schaltflächen können Sie die auf der CD enthaltenen Dokumente ansehen, die Identifizierung Ihres Computers durchführen und die ITI-Vertragsbedingungen lesen.

**Bild 2.3:** Startbildschirm

### 2.3.1 Installation

Nach Auswahl der Sprache wird der Installationsassistent gestartet.

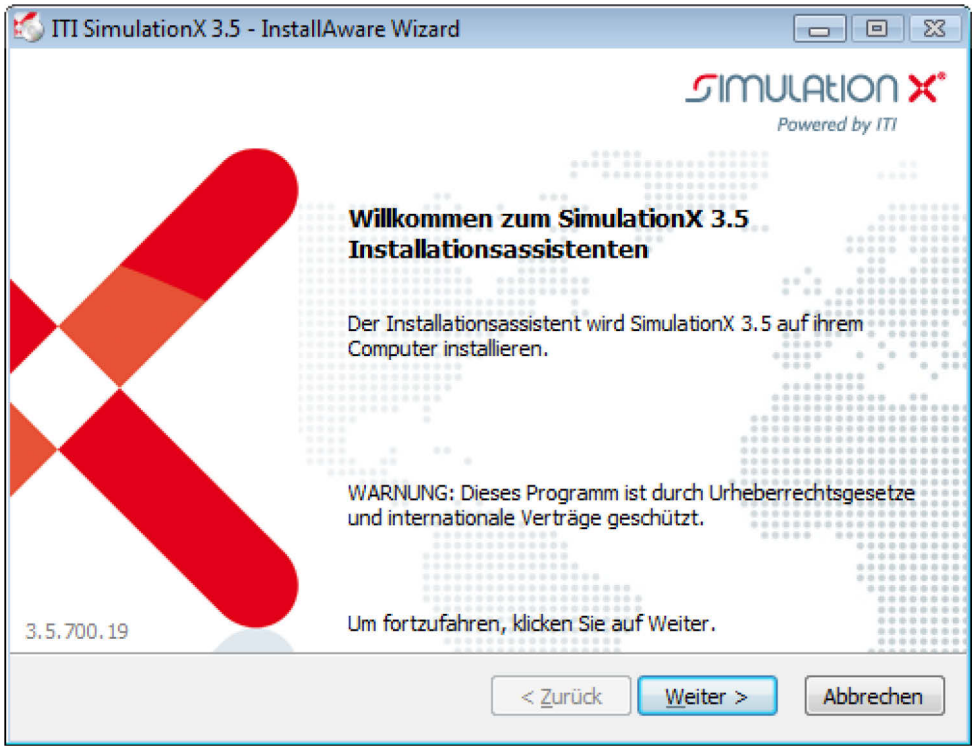



**Bild 2.4:** Start des Installationsassistenten

Sollte sich auf Ihrem System eine ältere SimulationX-Version befinden, so erkennt das der Installationsassistent und bietet Ihnen die Möglichkeit die ältere Version zu deinstallieren (siehe **Bild 2.5**).

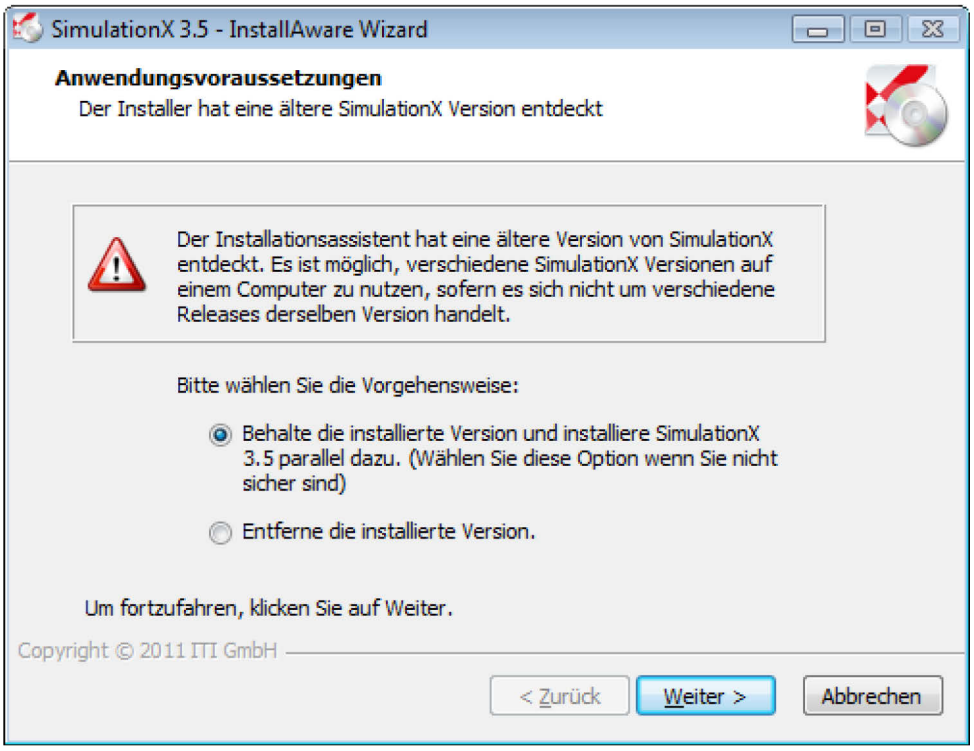



**Bild 2.5:** SimulationX Versionen lassen sich wahlweise parallel betreiben

Die Deinstallation einer älteren SimulationX-Version ist nicht unbedingt notwendig. Verschiedene SimulationX-Versionen lassen sich parallel auf einem

Rechner installieren und funktionieren unabhängig voneinander. Lediglich unterschiedliche Releases der Software lassen sich nicht parallel betreiben. Sollte der Installationsassistent ein älteres Release derselben SimulationX-Version auf dem Zielcomputer finden, so entfernt er dieses und installiert außerdem die Anwendungsvoraussetzungen bevor die eigentliche Installation beginnt.

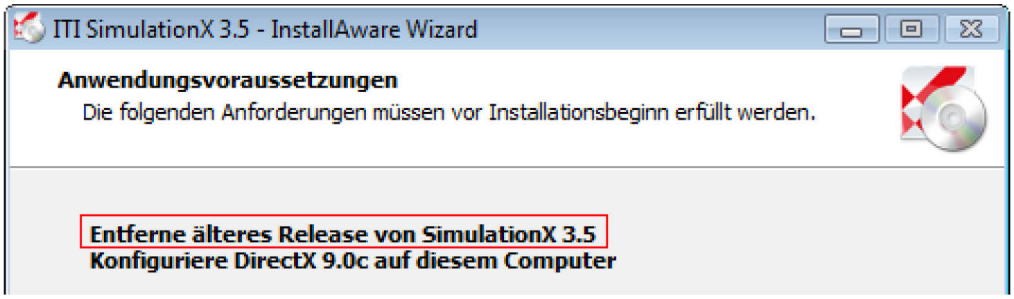



**Bild 2.6:** Ältere Releases derselben SimulationX-Version müssen deinstalliert werden

Für das Fortsetzen der Installation müssen Sie die Lizenzvereinbarung akzeptieren.

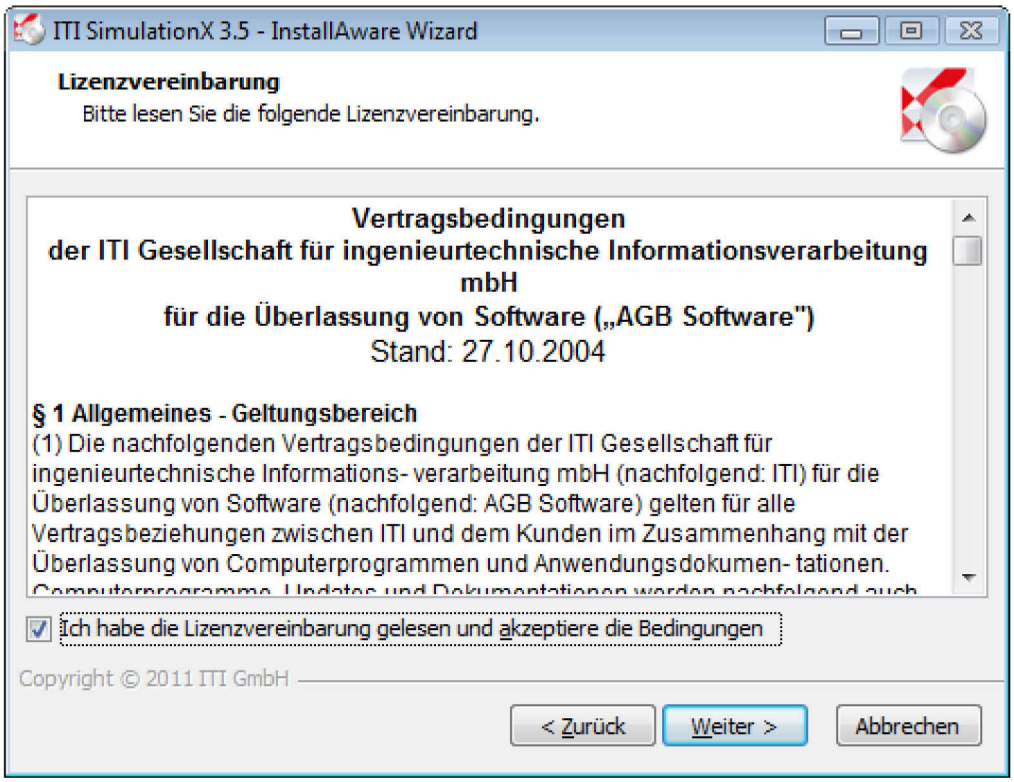



**Bild 2.7:** Lizenzvereinbarung

Im folgenden Dialog ist der Umfang der SimulationX-Installation zu wählen.

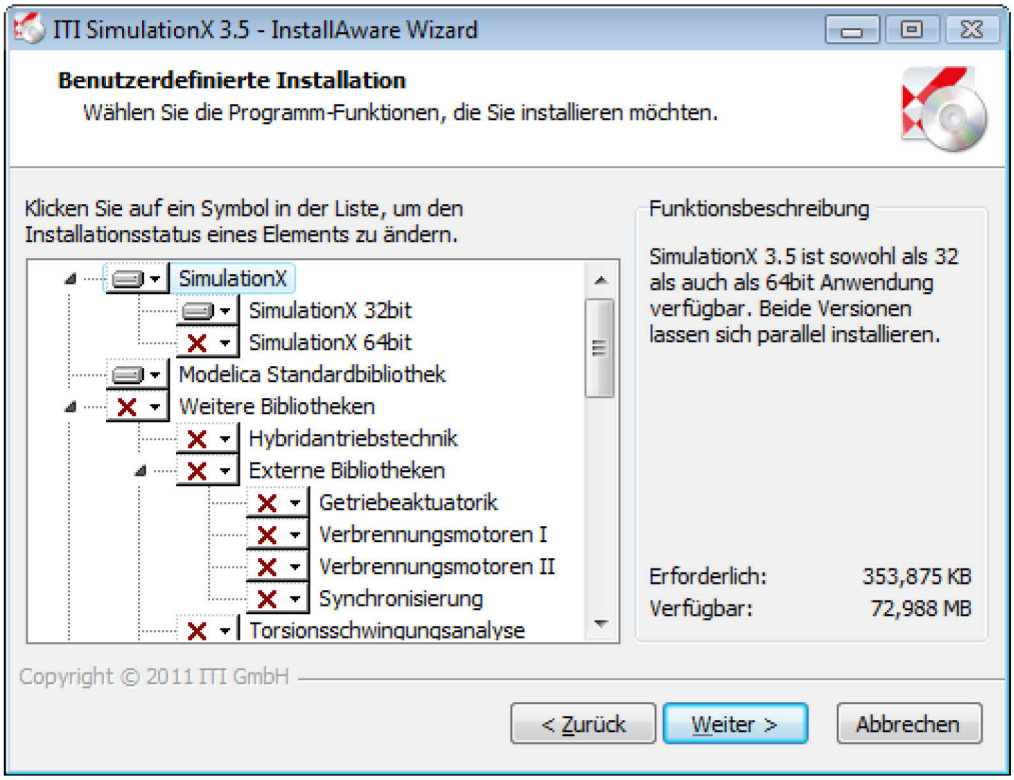



**Bild 2.8:** Auswahl des Umfanges der Installation

Es stehen folgende Arten zur Verfügung:

| | |
|---|---|
| Standard | Es werden alle standardmäßigen Funktionen von SimulationX installiert, die eine grundlegende Nutzung des Programms realisieren. |
| Angepasst | Der Nutzer kann selbst auswählen, welcher Umfang installiert werden soll. Dies kann für jedes Modul oder jede Option einzeln erfolgen. |
| Vollständig | Es erfolgt eine vollständige Installation. Dabei werden auch die Dokumentation und die Beispiele auf die Festplatte kopiert. |

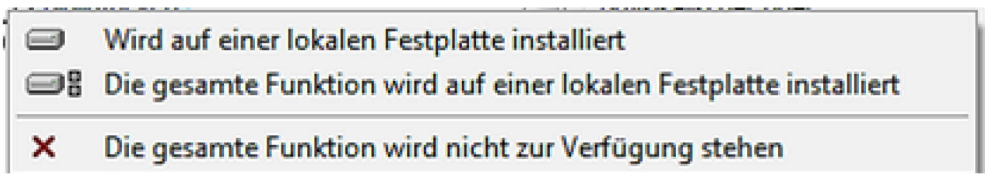



**Bild 2.9:** Auswahl der Module und Optionen bei der angepassten Installation

Für alle Module, die ständig benötigt werden, ist die Option „Wird auf einer lokalen Festplatte installiert“, zu wählen.

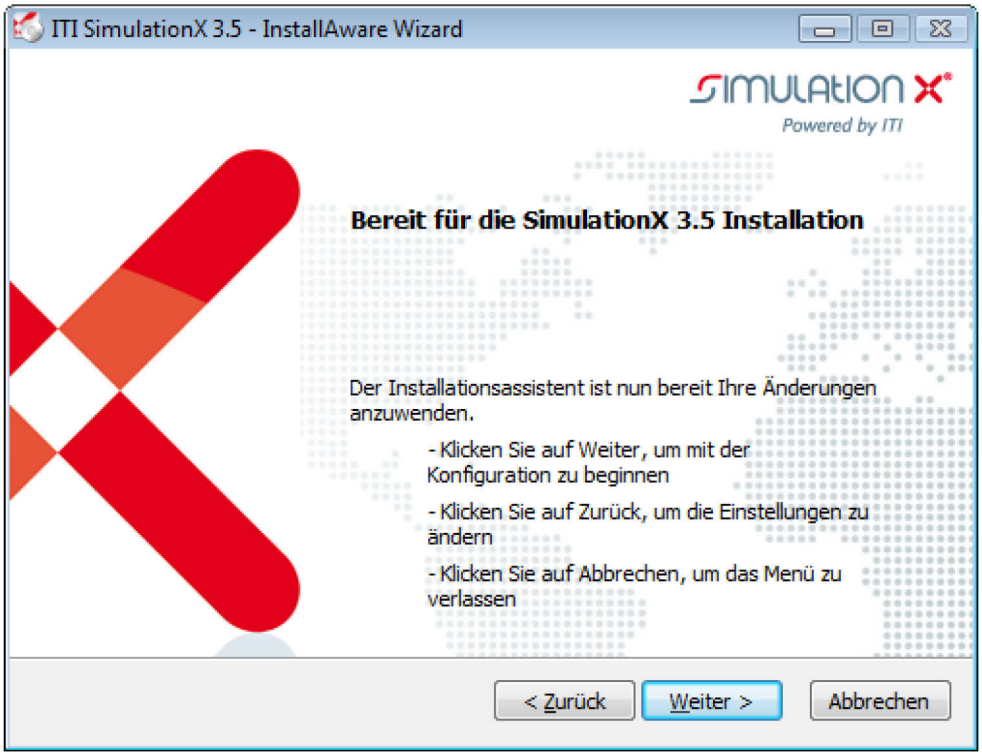



**Bild 2.10:** Bereitschaft zur Installation

Nach dem Betätigen von „Weiter“ beginnt, die Installation. Diese benötigt entsprechend der vorher getroffen Auswahl einige Minuten.

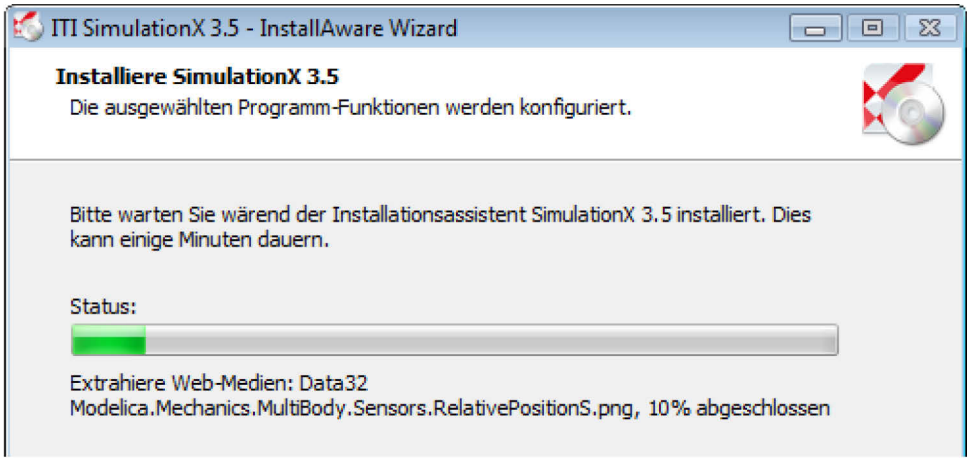



**Bild 2.11:** Anzeige des Installationsfortschritts

Mit der Anzeige des finalen Installationsbildschirmes ist die Installation von SimulationX beendet.

## 2.3.2 Erneute Installation bzw. Aktualisierung

Falls Sie die Installation erneut durchführen, obwohl SimulationX bereits installiert ist, sind in Abhängigkeit vom bereits installierten Release zwei Reaktionen möglich:

1. Sie wollen ein neues oder älteres Release installieren.
   Beim Installieren der Voraussetzungen für SimulationX wird die installierte Version vorher entfernt.
2. Sie wollen das gleiche Release noch einmal installieren.
   In diesem Fall wird die Wartungsseite des Setup-Assistenten gestartet.

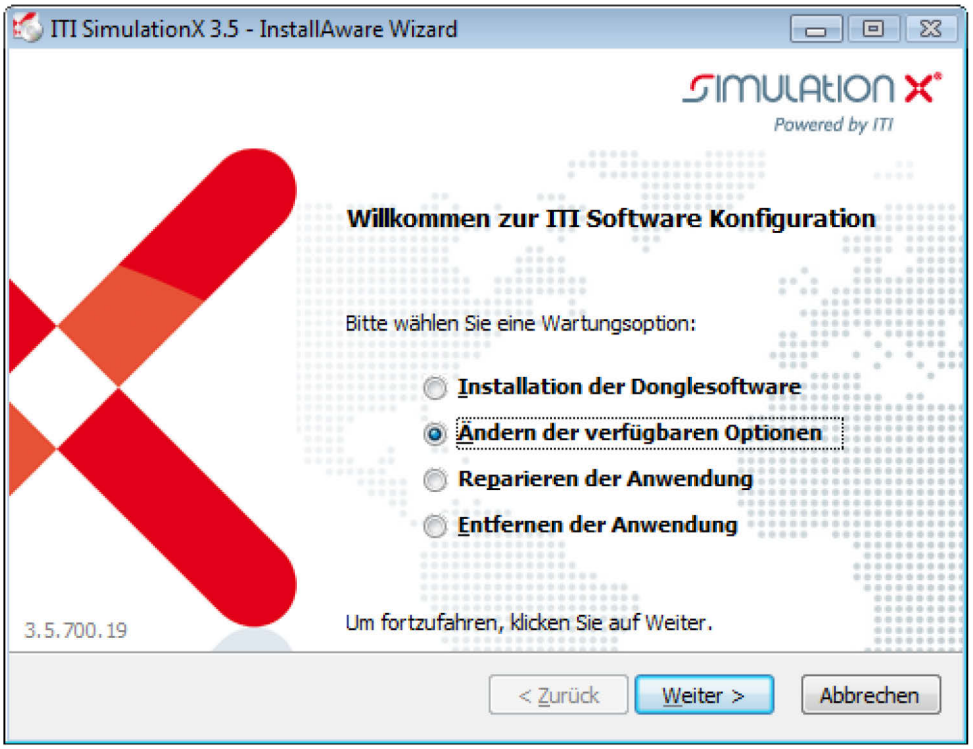



**Bild 2.12:** Dialog der Wartungsseite

Sie können das vorhandene Release **reparieren**, Module und Optionen **ändern** oder SimulationX **entfernen**.
Außerdem bietet der Dialog die Möglichkeit die Programmschutz-Software zu installieren, die für die Benutzung eines Hardware-Dongles benötigt.

Programmschutz-Software (Treiber, Dienste) muss nur auf Rechnern installiert werden, an denen ein Dongle angesteckt wird.

Zum Beispiel:

- an einem Arbeitsplatzrechner mit einer lokalen Professional oder Analyzer Edition als Einzelplatzversion mit Dongle oder
- an einem Rechner (Server bzw. Arbeitsstationen) mit Dongle, welcher vom Dongle Lizenzen im Netzwerk zur Verfügung stellt.

Die Installation der Programmschutzsoftware ist im **Abschnitt 2.5.2.1** nachzulesen.

Ein Computer muss identifiziert werden, wenn Sie eine Professional oder Analyzer Lizenz mit Lizenzdatei erworben haben. Die Identifikation ist im **Abschnitt 2.5.2.2** beschrieben.

## 2.4 Erster Programmstart

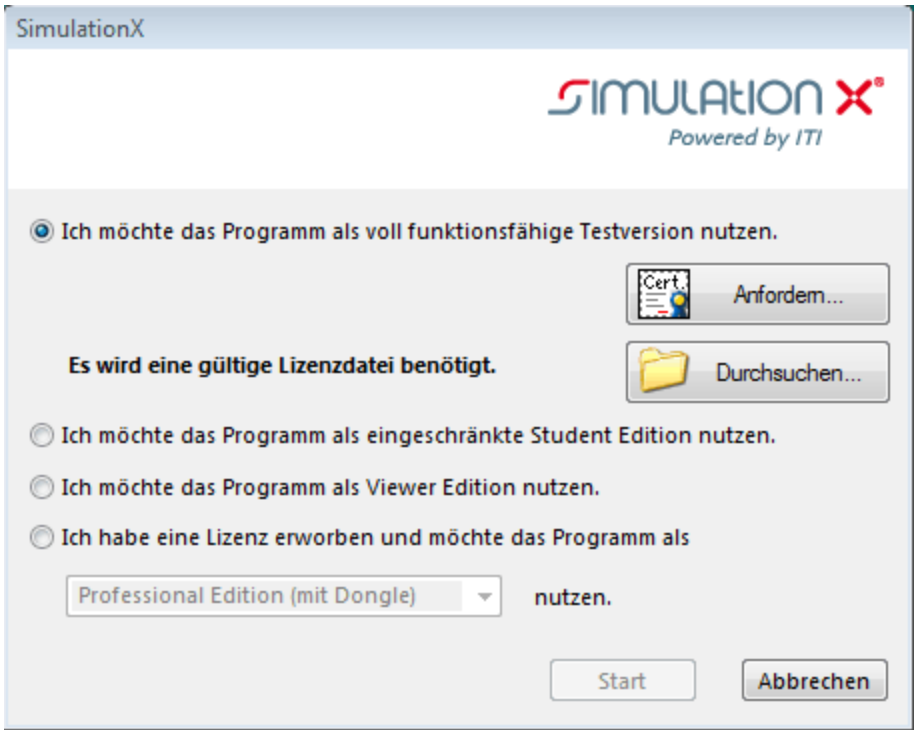



**Bild 2.13:** Editionsauswahldialog

Wenn SimulationX das erste Mal auf Ihrem Computer gestartet wird, meldet sich das Programm mit dem Dialog zur Auswahl der Edition (s. **Bild 2.13**).

Wählen Sie hier bitte die von Ihnen erworbene Edition aus und klicken Sie auf ***Start***. Siehe auch die **Kapitel 2.2.2** bis **2.2.6**.

### 2.4.1 SimulationX – die Testversion

Für die Ausführung des Programms als Testversion benötigen sie eine Lizenzdatei. Klicken Sie dazu auf die Schaltfläche ***Anfordern…*** (s. **Bild 2.13**).

Nach einem Hinweis, dass Geräte mit nichtpermanenter MAC-Adresse (z.B. PDA's) vom Computer getrennt werden sollen, erscheint der Anforderungsdialog (s. **Bild 2.14**).

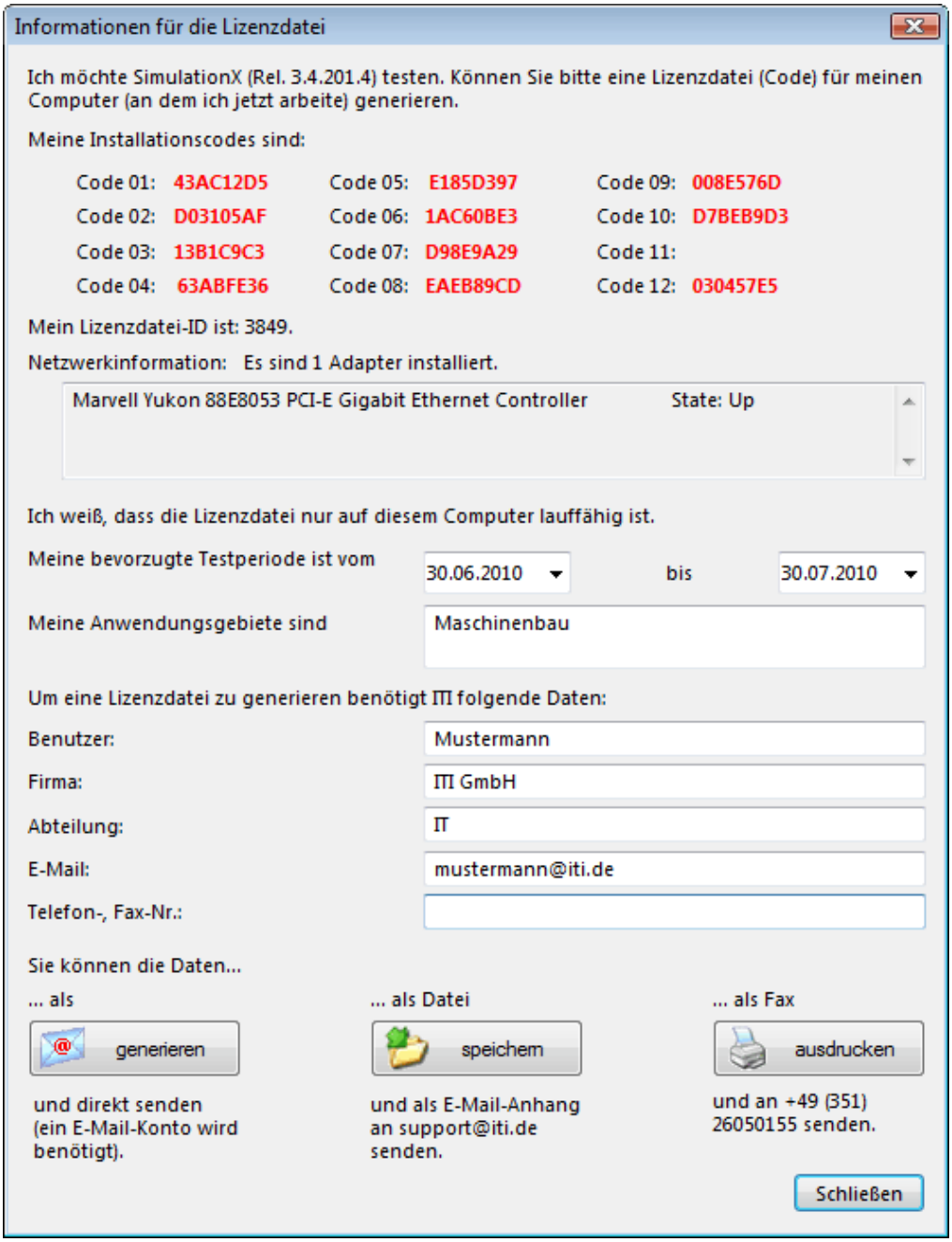



**Bild 2.14:** Dialog zum Anfordern einer Lizenzdatei

Ergänzen Sie bitte Ihre gewünschte Testperiode, geplante Anwendungsgebiete und persönliche Daten.
Sofern ein Email-Client auf Ihrem System installiert ist, können Sie die Daten direkt übermitteln. Mit Klick auf die Schaltfläche ***generieren*** wird eine E-Mail erzeugt, die Sie nur noch absenden brauchen (s. **Bild 2.15**).

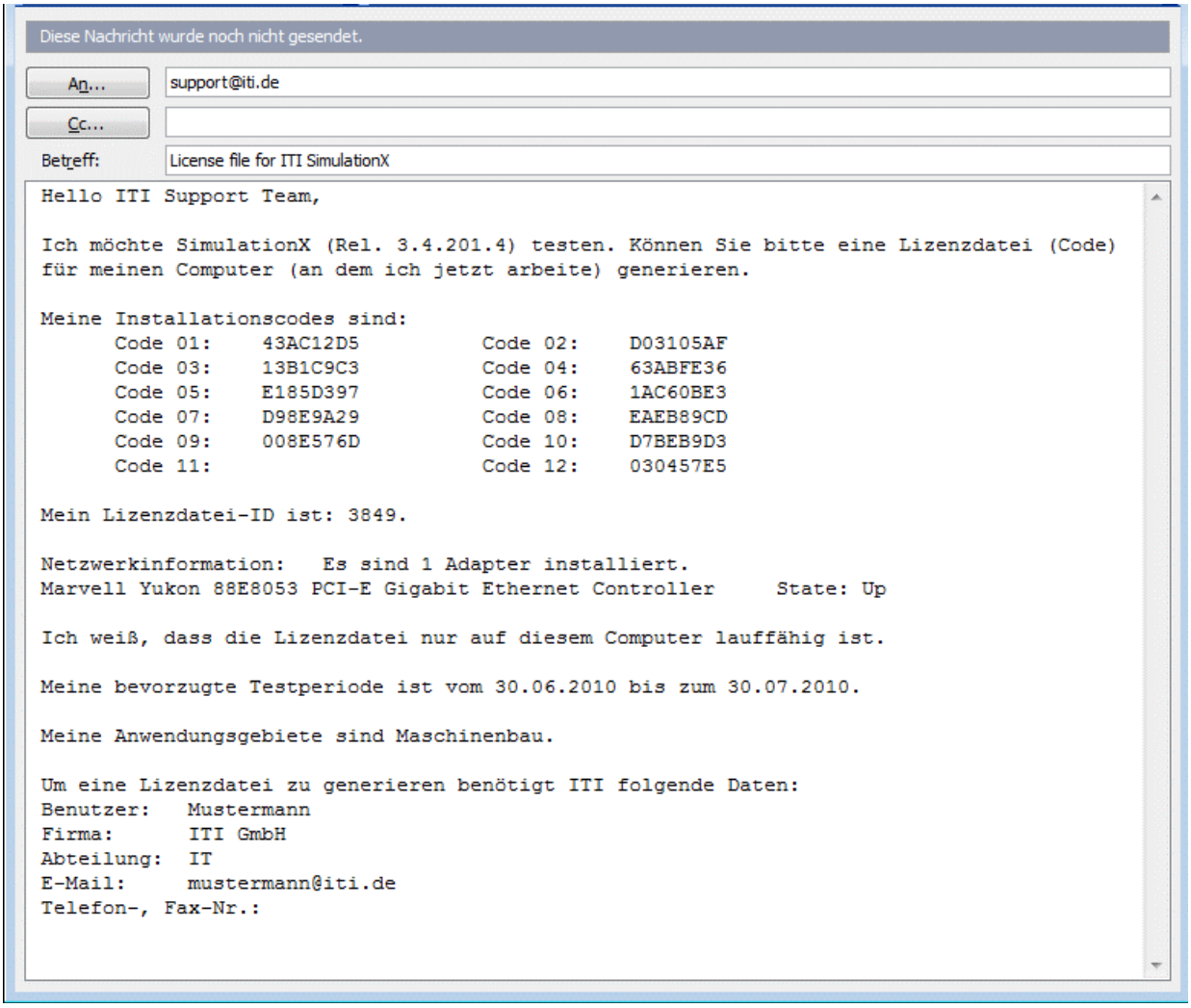



**Bild 2.15:** Email zur Anforderung einer Lizenzdatei

Nachdem Sie die E-Mail versandt haben, wird Ihnen das ITI Team nach Prüfung zur Berechtigung die Lizenzdatei für Ihren Rechner zur Verfügung stellen.

Die Lizenzdatei wird dem Programm zum Start (s. **Bild 2.13**) mit Klick auf die Schaltfläche ***Durchsuchen…***) zur Verfügung gestellt. Es sind **keine** weiteren Installationen (Treiber, Dienste etc.) nötig.

Bei jedem weiteren Programmstart wird ein Dialog entsprechend **Bild 2.16** mit der **Dauer der Testperiode** angezeigt. Mit der Schaltfläche ***Aktualisieren…*** haben Sie während der Testperiode die Möglichkeit, dem Programm eine geänderte Lizenzdatei zur Verfügung zu stellen.

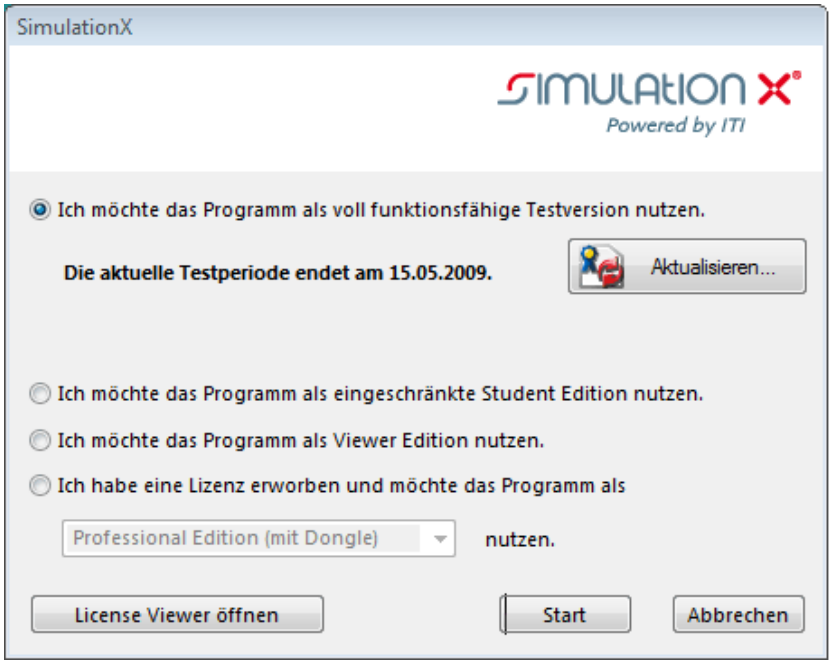



**Bild 2.16:** Startdialog der Testversion

Wenn die Testperiode abgelaufen ist, erscheint ein entsprechender Hinweis und der Startdialog entsprechend **Bild 2.13**. Sie können eine neue Lizenzdatei anfordern oder das Programm als **Student oder Viewer Edition** weiternutzen.

Beachten Sie bitte, dass die von ITI vergebene Lizenzdatei nur mit dem Computer arbeitet, von dem sie angefordert wurde!

Wenn Sie in der Zwischenzeit eine Lizenz mit Hardware-Key/Dongle erworben haben, **installieren Sie bitte erst die Key/Dongle-Software** von der Produkt CD.

Diese finden Sie im Verzeichnis Dongle auf der CD. Dazu ist die „SentinelProtectionInstaller.exe“ auszuführen (für 64-Bit Systeme nutzen Sie „SDD_USB_x64_Installer.exe“). Die Installation ist unter **2.5.2.1** detailliert beschrieben.

## 2.4.2 Student- oder Viewer-Edition

Für die Arbeit mit dem Programm als Student- oder Viewer-Edition benötigen Sie keine weiteren Angaben.

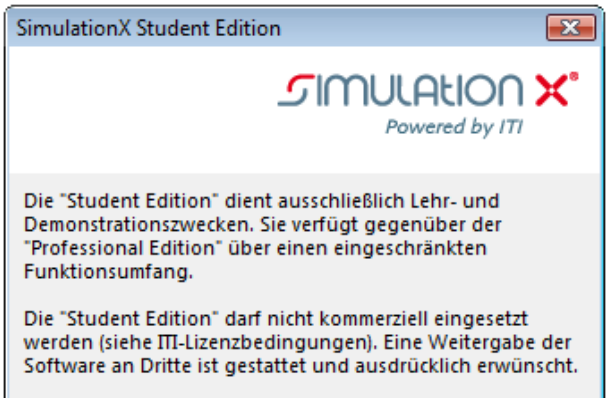



**Bild 2.17:** Infodialog bei Programmstart von SimulationX als Student Edition

Nach entsprechender Auswahl startet das Programm mit einem Dialog (s. **Bild 2.17**).

### 2.4.3 Analyzer- oder Professional-Edition

Soll das Programm als Analyzer oder Professional-Edition genutzt werden ist der Erwerb einer Lizenz (siehe auch **2.6**) Voraussetzung. Diese Lizenz ist beim Start des Programms auszuwählen.

Nachdem Sie entsprechend **Bild 2.13** die Programmversion mit Lizenz gewählt haben, wird der Dialog zur Auswahl eines **Keys/Dongles** (siehe **2.7**) oder der Dialog zur Auswahl einer **Lizenzdatei** (siehe **2.8)** geöffnet. Nach der Auswahl einer gültigen Lizenz kann das Programm gestartet werden.

## 2.5 Hinweise für Administratoren

### 2.5.1 Installation im Stapelbetrieb (ohne Nutzerinteraktion)

Dieser Installationsmode eignet sich besonders für Netzwerke zur Softwareverteilung. Die Installation von SimulationX kann dazu im Stapelbetrieb (Batchbetrieb) ohne Interaktion des Installierenden durchgeführt werden.

Die Installation im Stapelbetrieb wird durch den Kommandozeilenschalter „/s“ initiiert. Dabei wird ohne Angabe von Optionen eine Standardinstallation ausgeführt (siehe 2.3.1).

Im Stapelverarbeitungsmodus werden die folgenden Parameter akzeptiert:

| | |
|---|---|
| ADDLOCAL=ALL | Dies installiert alle Funktionen von SimulationX die bei einer vollständigen Installation installiert werden (siehe 2.3.1) |
| TARGETDIR=“Pfad“ | Diese Option bestimmt das Zielverzeichnis der Installation |
| CHECKPRE=[TRUE\|FALSE] | Mit diesem Schalter lässt sich die Prüfung der Installationsvoraussetzungen abschalten.<br>**Achtung: Fehlende Installationsvoraussetzungen können den ordnungsgemäßen Betrieb der Software gefährden.** |

Beispiel:
Die automatische Installation kann vom Kommandozeilendialog gestartet werden (Klick ***Start/Ausführen***):

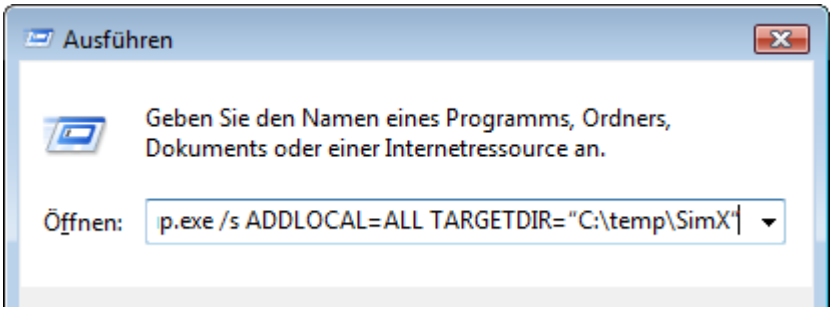



**Bild 2.18:** Kommandozeilendialog

```
D:\Setup\Setup.exe /s ADDLOCAL=ALL TARGETDIR="C:\temp\SimX"
```

Ähnliche Zeilen können auch in Batch-Dateien verwendet werden.

## 2.5.2 Installationen zum Programmschutz

Im Gegensatz zu den frei verfügbaren Editionen „STUDENT“ und „VIEWER“ sind die Editionen „ANALYZER“ und „PROFESSIONAL“ mit einem Programmschutz versehen. Für die Nutzung von SimulationX als Analyzer oder Professional Edition müssen die Programmschutzbestandteile installiert werden.

Der Programmschutz kann **entweder** mittels **Hardware-Key/Dongle** (siehe **2.5.2.1**) **oder** mittels **Lizenzdatei** (siehe **2.5.2.2**) realisiert werden. Mehr über die Lizenzierung erfahren sie im **Abschnitt 2.6.**

Nach der Installation der Software ist die Einrichtung des Softwareschutzes notwendig.

Dazu benötigen Sie die Zugriffsrechte eines Administrators auf dem Zielsystem.

### 2.5.2.1 Hardwareschutz (Key / Dongle)

Diese Installationen sind nur an jenen Computern auszuführen, an denen auch tatsächlich ein Key (Dongle) steckt. Mit dieser Software werden beide Komponenten (Treiber und Server) installiert. Dabei sorgt der *Treiber* dafür, dass SimulationX mit dem Key kommunizieren kann. Und der *Server* verwaltet die Lizenz(en) – sowohl für den Einzelplatz als auch im Netzwerk.

Dies kann nach der erfolgreichen Installation von SimulationX erfolgen, indem Sie entweder:

- Das Setup erneut starten
- über das SimulationX Programmmenü den Punkt „ITI Software Konfiguration“ wählen
- über den Windowsdialog für installierte Anwendungen in der Systemsteuerung ihres Betriebssystems und den Eintrag „ITI SimulationX V.V“ (in der jeweiligen Version deren Installation Sie bearbeiten möchten)

Bei jedem der oben genannten Wege erscheint der SimulationX Wartungsdialog (siehe **Bild 2.19**). Hier wählen Sie den Punkt „Installation der Donglesoftware).

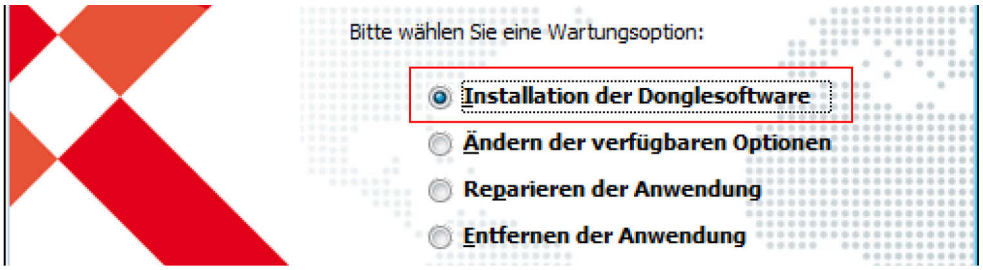



**Bild 2.19:** Der SimulationX Wartungsdialog

Die Installation der Treibersoftware wird gestartet.

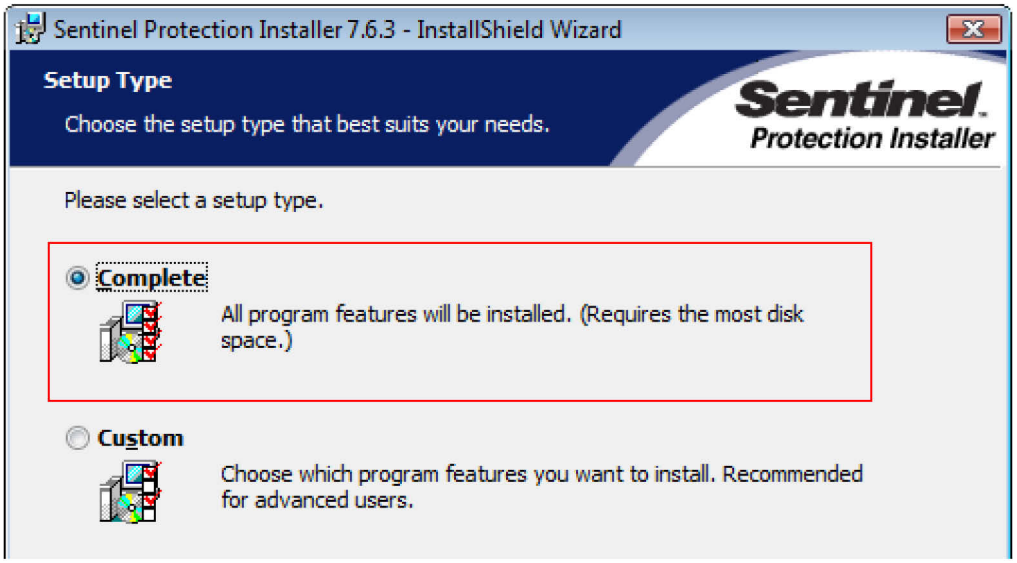



**Bild 2.20:** Key (Dongle) Treiber Installation – Setup Typ

Nach der „Willkommen“-Seite, der Lizenzbestätigung und der Verzeichniswahl wählen Sie als Setup Typ ***Complete*** aus (s. **Bild 2.20**).

Um Speicherplatz zu sparen, können Sie die Installation auch anpassen. Wählen Sie dazu die Option ***Custom***. In diesem Fall sind als nächstes die zu installierenden Komponenten auszuwählen.

Deaktivieren Sie bitte die für Ihren Key **nicht** benötigte Komponente:

- ein paralleler Key benötigt keinen "USB System Driver"
- ein USB Key benötigt keinen "Parallel System Driver".

Klicken Sie dazu auf das entsprechende Festplattensymbol und wählen den Menüeintrag „This feature will not be available“ (s. **Bild 2.21**).

**Bild 2.21:** Key (Dongle) Treiber Installation – Komponentenauswahl

Achten Sie bitte darauf, dass USB Keys während der Treiberinstallation NICHT ANGESTECKT sind. Stecken Sie den Key erst NACH der Installation an.

Die Installation der Hardwareschutzkomponenten ist nur **einmalig** auf Ihrem System auszuführen. Ein weiterer Start der Installationen ruft den Programmpflegedialog der jeweiligen Installation auf.

### 2.5.2.2 Softwareschutz mit Lizenzdatei

Für diese Lizenzart sind **keinerlei Installationen** notwendig. Da die Lizenzdatei an eine konkrete Hardware gebunden werden muss, ist die Identifikation des Computers, auf dem die Lizenzdatei später gespeichert werden soll, notwendig.

Dazu muss auf diesem Computer das Programm zur Identifizierung ausgeführt werden (siehe **Bild 2.22**).
Nachdem Sie die Daten an ITI übermittelt haben, wird Ihnen das ITI Team nach Prüfung zur Berechtigung die Lizenzdatei für Ihren Rechner zur Verfügung stellen. Speichern Sie die Lizenzdatei auf dem Computer in einem Verzeichnis, in dem **alle Nutzer Schreibrechte** haben (die Lizenzdatei wird ständig aktualisiert).

Standardmäßig sollte die Lizenzdatei im Verzeichnis "`C:\Benutzer\All Users\Anwendungsdaten\ITI GmbH\SimulationX 3.5`" gespeichert werden.

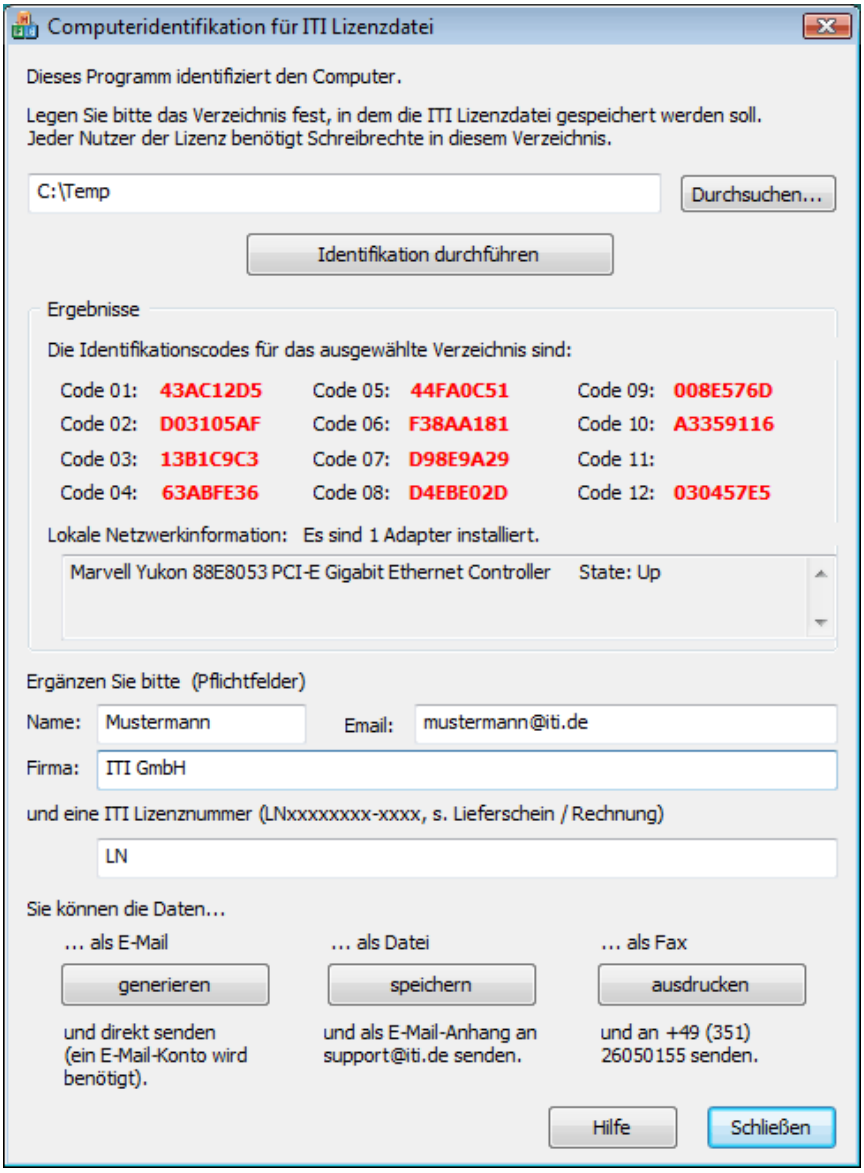



**Bild 2.22:** Computeridentifikation

## 2.6 SimulationX – die Lizenzierung

Kostenpflichtige Lizenzen benötigen Sie für die Analyzer und Professional Editionen. Die Lizenzierung für SimulationX erfolgt mittels Hardware-Key/Dongle oder mit einer Lizenzdatei.
SimulationX ist ein modular aufgebautes Programmsystem. Für die von Ihnen bestellten Module wurde von ITI eine spezifische Lizenz generiert.

### 2.6.1 Einzelplatzlizenz

Diese Lizenz ist nur am lokalen Computer nutzbar.

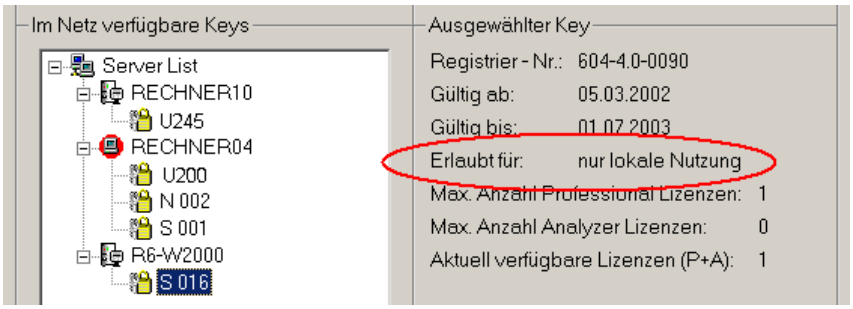



**Bild 2.23:** Auswahl Einzelplatzlizenz

Stecken Sie den Key an die entsprechende Schnittstelle (parallel bzw. USB). Installieren Sie alle Bestandteile des Softwarepakets entsprechend **2.2.5, 2.2.6** und **2.5**. Wählen Sie beim ersten Start der SimulationX Professional oder Analyzer Edition die Verbindung zu Ihrem lokalen Key aus (s. **Bild 2.25**). Schalten Sie das „Öffnen des Verbindungsdialogs bei jedem Programmstart aus“ (s. **2.7**).

### 2.6.2 Netzwerklizenz

Diese Lizenz ist im gesamten Netzwerk nutzbar.

Wählen Sie einen Computer im Netzwerk aus (siehe **Bild 2.23**), welcher die Lizenzen verwalten soll. Auf diesem Computer muss ein Windows NT Betriebssystem (2000, XP Prof. und neuere) laufen. Stecken Sie den Key an die entsprechende Schnittstelle dieses Computers. Installieren Sie auf diesem Computer die Komponenten zum Programmschutz entsprechend **2.5**.

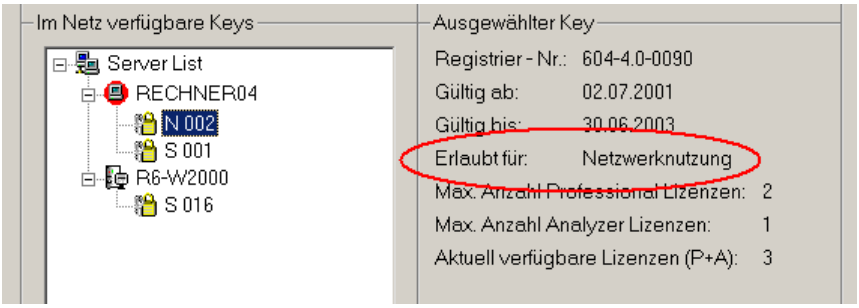



**Bild 2.24:** Auswahl Netzwerklizenz

Installieren Sie auf allen Arbeitsstationen, auf denen SimulationX laufen soll, das Programm SimulationX entsprechend **2.3**. Dazu können Sie beispielsweise die Installation im Stapelbetrieb (s. **2.5.1**) nutzen.

Der Anwender wählt beim ersten Start von SimulationX den Netzwerk-Key aus (s. **Bild 2.23**). Falls nur ein Key im Netzwerk vorhanden ist, empfiehlt sich das Ausschalten des Verbindungsdialogs bei jedem Programmstart (s**. Bild 2.26**).

### 2.6.3 Mehrere unterschiedliche Lizenzen im Netzwerk

SimulationX nimmt nicht die erst beste Lizenz im Netzwerk, sondern verlangt die gezielte Verbindung mit einer bestimmten Lizenz. Deshalb ist es möglich, mehrere SimulationX Lizenzen mit **unterschiedlichem Funktionsumfang** in einem Netzwerk zur Verfügung zu haben.

Beispiel:

Sie benötigen **2 Professionallizenzen** mit

- 1D Mechanik,
- Signalgliedern,
- Antriebstechnik,
- Eigenfrequenzen und
- Variantenassistent

zur Modellentwicklung. Außerdem müssen Sie ständig im Stapelbetrieb Variantenrechnungen mit diesen Modellen durchführen. Sie benötigen **1 Analyzerlizenz** mit gleichem Funktionsumfang.

Darüber hinaus soll "im Nachbarzimmer" SimulationX für Untersuchungen von Hydraulikkomponenten eingesetzt werden. Dazu benötigen Sie **1 Professionallizenz** mit

- 1D Mechanik,
- 3D Mechanik,
- Signalgliedern,
- Antriebstechnik,
- Hydraulik,
- Magnetik,
- Eigenfrequenzen und
- Variantenassistent

Da in einem Key immer **genau eine Modulkombination** (ein Funktionsumfang) gespeichert werden kann, liefert ITI für diesen Anwendungsfall zwei Keys.

Sie installieren die Software im Netzwerk entsprechend **2.6.2**. Dabei ist es unerheblich, ob beide Keys an einem Computer oder an unterschiedlichen Computern stecken. Wichtig ist nur, dass sie im Netzwerk sichtbar sind.

Der Anwender wählt beim Start von SimulationX den gewünschten Netzwerk-Key aus (s. **2.7**). Mit der Option „Diesen Dialog bei Programmstart anzeigen“ kann der Anwender entscheiden, ob bei jedem Programmstart gefragt werden soll, oder ob eine Vorzugseinstellung bis auf Widerruf gilt (Häkchen entfernen). Die Auswahl kann jederzeit über das Menü ***Optionen/Extras*** auf der Seite ***Lizenzierung*** durch Klick auf die Schaltfläche ***Neue Verbindung*** geändert werden.

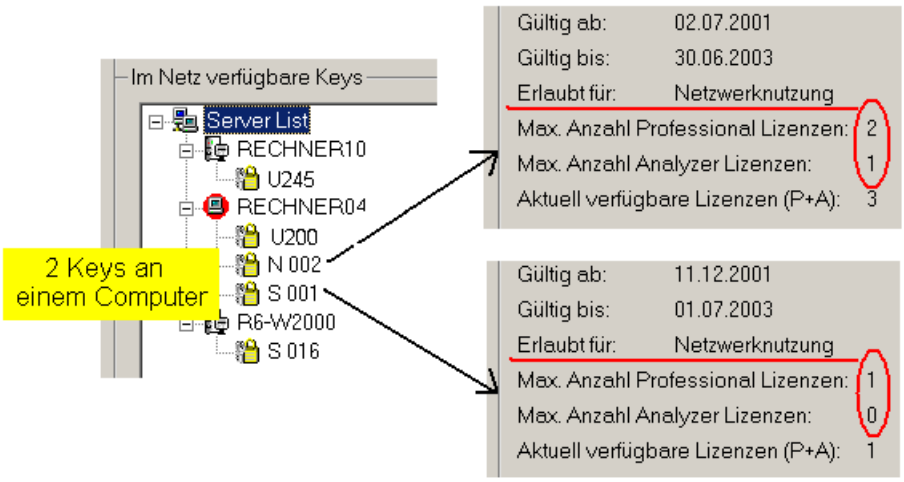



**Bild 2.25:** Unterschiedliche Lizenzen im Netzwerk

Die Auswahl einer Lizenz ist nutzerbezogen. D. h., auf einer Maschine können unterschiedliche Nutzer ohne Rückfrage (Häkchen für Verbindungsdialog entfernt) unterschiedliche Lizenzen nutzen.

## 2.7 Auswahl eines Hardware-Key / Dongle

Welchen Key/Dongle SimulationX beim Start nutzen soll, können Sie festlegen. Insbesondere nach einer Erstinstallation ist die Auswahl noch nicht definiert. Dazu wird der Dialog zur Auswahl eines Keys/Dongles geöffnet (s. **Bild 2.26**). Dieser Dialog kann aber auch jederzeit über das Menü ***Extras/Optionen*** auf der Seite ***Lizenzierung*** mit Klick auf die Schaltfläche ***Neue Verbindung*** geöffnet werden. Sie entscheiden im ersten Schritt ob überhaupt („Abbrechen“), bzw. wo (lokal oder im Netz) nach Lizenzen gesucht werden soll (s. **Bild 2.25**).

**Bild 2.26:** Start zur Suche nach verfügbaren Lizenzen

Die Suche nach verfügbaren Keys kann jederzeit wiederholt werden. Klicken Sie dazu auf die entsprechende Schaltfläche im Feld "Suche nach verfügbaren Keys". Die Suche erfolgt entweder anhand von Hostnamen oder mittels IP-Adressen. Die zu nutzende Variante wählen Sie bitte mit den Radioknöpfen „Hostname gestützte ...“ bzw. „IP Adressen gestützte ...“ aus. Sollten Sie unsicher sein, welche Auswahl für Ihr System zutreffend ist, wählen Sie die Standardeinstellung „Hostname gestützte Auswahl“.

Im Feld „Im Netz verfügbare Keys“ werden alle im aktuelle Subnetz zur Verfügung stehenden Keys in einer Baumstruktur angezeigt. Der lokale Computer ist mit einem roten Kreis gekennzeichnet.

Wenn Sie in der Baumstruktur auf einen SimulationX Key klicken (gekennzeichnet durch den Key Ident der Form <Buchstabe> <Zahl> -> X ###), werden im Feld „Ausgewählter Key“ wichtige Inhalte dieses Keys angezeigt. Steht auf diesem Key mindestens eine gesuchte Lizenz (Analyzer bzw. Professional) aktuell zur Verfügung, wird die Schaltfläche ***Diesen Key aktivieren*** freigegeben.
Wenn Sie diese Schaltfläche klicken, wird der gerade im Baum ausgewählte Key zur „Aktuellen Verbindung“ erklärt. D.h., SimulationX wird beim Starten eine Lizenz der dieses Keys (konkreter Computer, konkreter Key) nutzen.

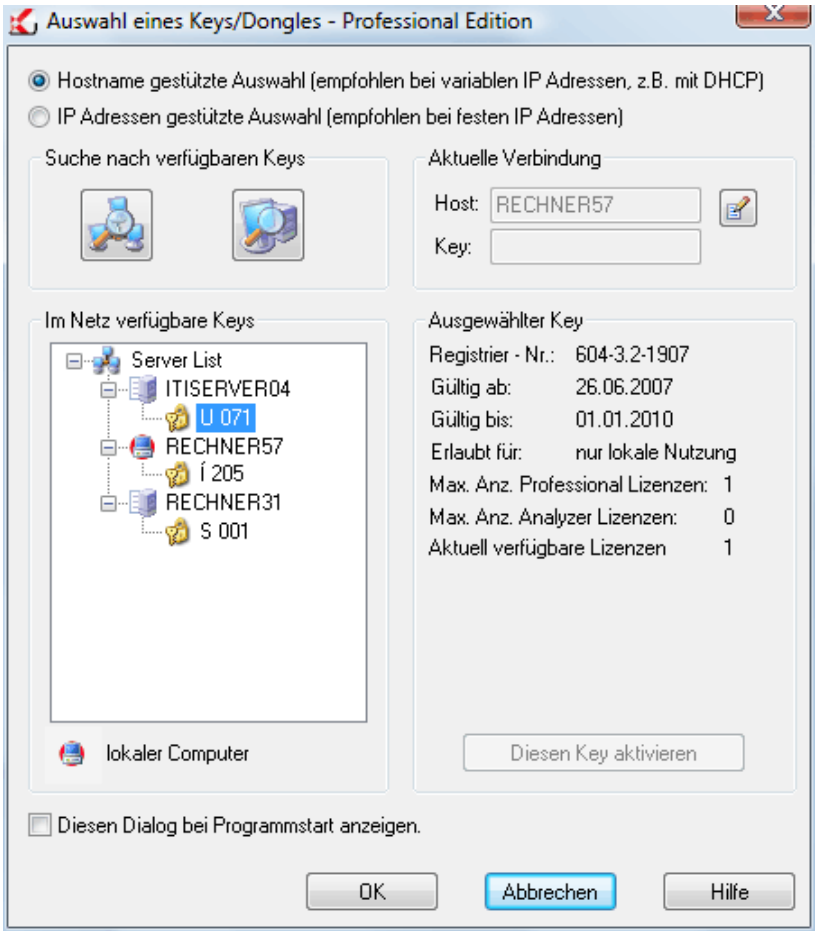



**Bild 2.27:** Dialog zur Verbindungsauswahl für SimulationX Lizenzen

Die Suche nach Keys im Netzwerk wird über so genannte Broadcast-Rufe nach dem Motto: „Ich rufe 'Hallo Sentinel' – mal sehen, wer antwortet“ realisiert. Möglicherweise werden im Feld „Im Netz verfügbare Keys“ nicht alle tatsächlich vorhandenen Keys angezeigt. Häufigste Ursache dafür ist ein Verbot von Broadcasts in Ihrem Netzwerk, bzw. die Keys stecken an Rechnern in einem anderen Subnetz. Wenn das so ist, können Sie natürlich nicht den gewünschten Key aus der Baumstruktur, wie oben beschrieben, wählen. Sie müssen in diesem Fall die aktuelle Verbindung „per Hand“ einstellen.

Dazu klicken Sie die Schaltfläche im Feld „Aktuelle Verbindung“. Daraufhin werden die Editfelder „Host“ und „Key“ zum Editieren freigegeben. Tragen Sie Rechnername und Key Ident in die entsprechenden Felder ein. Die Auswahl in der Baumstruktur und Klicken der Schaltfläche „Diesen Key aktivieren“ entfällt.
Soll der Verbindungsauswahl-Dialog nicht bei jedem Start von SimulationX geöffnet werden (z. B. bei einer Einzelplatzlizenz) ist das Häkchen „Diesen Dialog bei Programmstart anzeigen“ zu entfernen.

Wird SimulationX im Stapelbetrieb (z. B. über COM) genutzt, sollte das Häkchen generell entfernt werden, da immer mit einer definierten Lizenz zu arbeiten ist.

Klicken Sie auf ***OK*** und SimulationX wird gestartet, bzw. Sie können im ***Optionen*** -Dialog weiterarbeiten.

### 2.7.1 Änderung einer Lizenz

Es kommt vor, dass der Key umprogrammiert werden muss, beispielsweise weil

- die Nutzungsdauer (der Mietzeitraum) abgelaufen ist und Sie die Nutzung verlängern möchten, oder
- sich der Funktionsumfang durch Zukauf von Modulen geändert hat, oder
- sich die Lizenzanzahl durch Zukauf vergrößert hat, oder
- Sie Ihre Einzelplatzlizenz jetzt im Netzwerk nutzen wollen.

Diese Operation können Sie bei sich „vor Ort“ schnell und unkompliziert ausführen. Dazu bestellen Sie bei ITI einen Updatecode (eine Textdatei, z. B. per E-Mail). Mit dieser Textdatei gehen Sie wie folgt um:

1. Speichern Sie den Updatecode auf Ihrer Festplatte.
2. Starten Sie SimulationX (z. B. als Student Edition falls der Key abgelaufen sein sollte).
3. Öffnen Sie über das Menü ***Optionen/Extras*** den Optionsdialog.
4. Aktivieren Sie die Seite ***Lizenzierung***.
   Kontrollieren Sie in der linken oberen Ecke, dass der richtige Key für das Update eingestellt ist (s. **Bild 2.27**).

   Falls das nicht der Fall ist (z.B. weil Sie die Student Edition gestartet haben), wählen Sie erst mit ***Edition ändern in*** die „Professional (mit Dongle)“ aus. Anschließend wählen Sie mittels ***Neue Verbindung*** den zu „updatenden“ Key aus (die Anzeige in der linken oberen Ecke erfolgt erst nach dem Programmneustart).
5. Öffnen Sie mit Klick auf die Schaltfläche ***Update Key*** den Dateidialog.
6. Wählen Sie die unter 1. abgespeicherte Datei aus und quittieren Sie den Dialog mit ***Öffnen***.

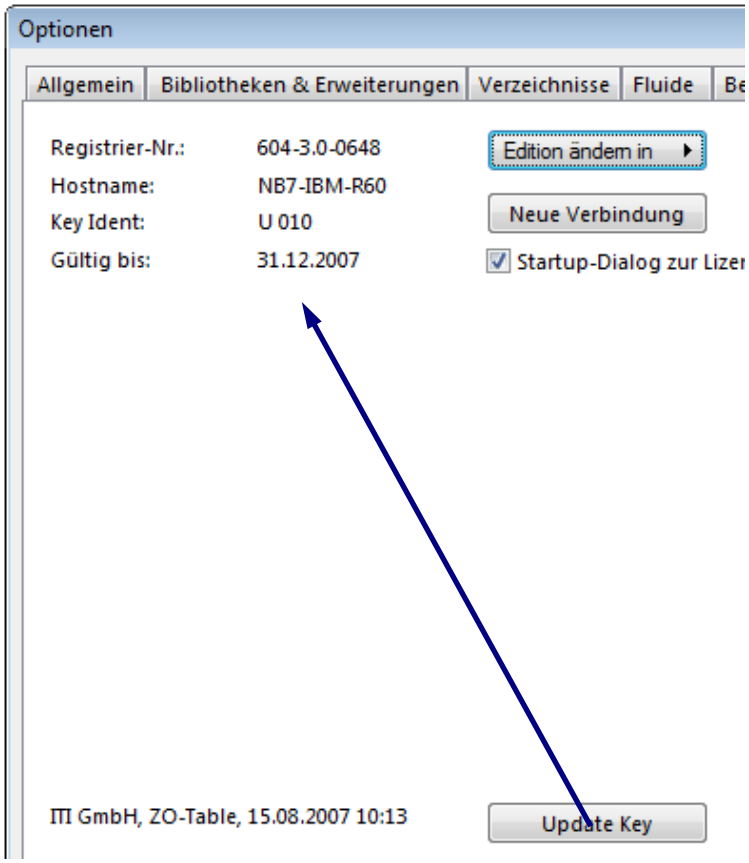



**Bild 2.28:** Update eines SimulationX Keys

Das Key-Update wird durchgeführt. Dazu wird die Konsistenz des Codes geprüft (beim Übertragen und Speichern sollte nichts verloren gegangen sein). Außerdem wird geprüft, ob der Code für den eingestellten Key bestimmt ist. Der Updatevorgang wird mit einer Meldung (Erfolg oder Fehler) beendet.

**Beim Update darf nur ein Key angesteckt sein, sonst erhalten Sie eine Fehlermeldung!**

## 2.8 Auswahl einer Lizenzdatei

Die von Ihnen erworbene(n) Lizenzdatei(en) müssen dem Programm „bekannt“ gemacht werden. Darüber hinaus sollen die Lizenzdateien auch problemlos verwaltet werden. Diesen Zwecken dient der Dialog ***Auswahl einer Lizenzdatei*** (**Bild 2.29**).

Insbesondere beim ersten Start von SimulationX mit Lizenzdatei ist die Lizenzdatei einmalig der Verwaltung hinzuzufügen.

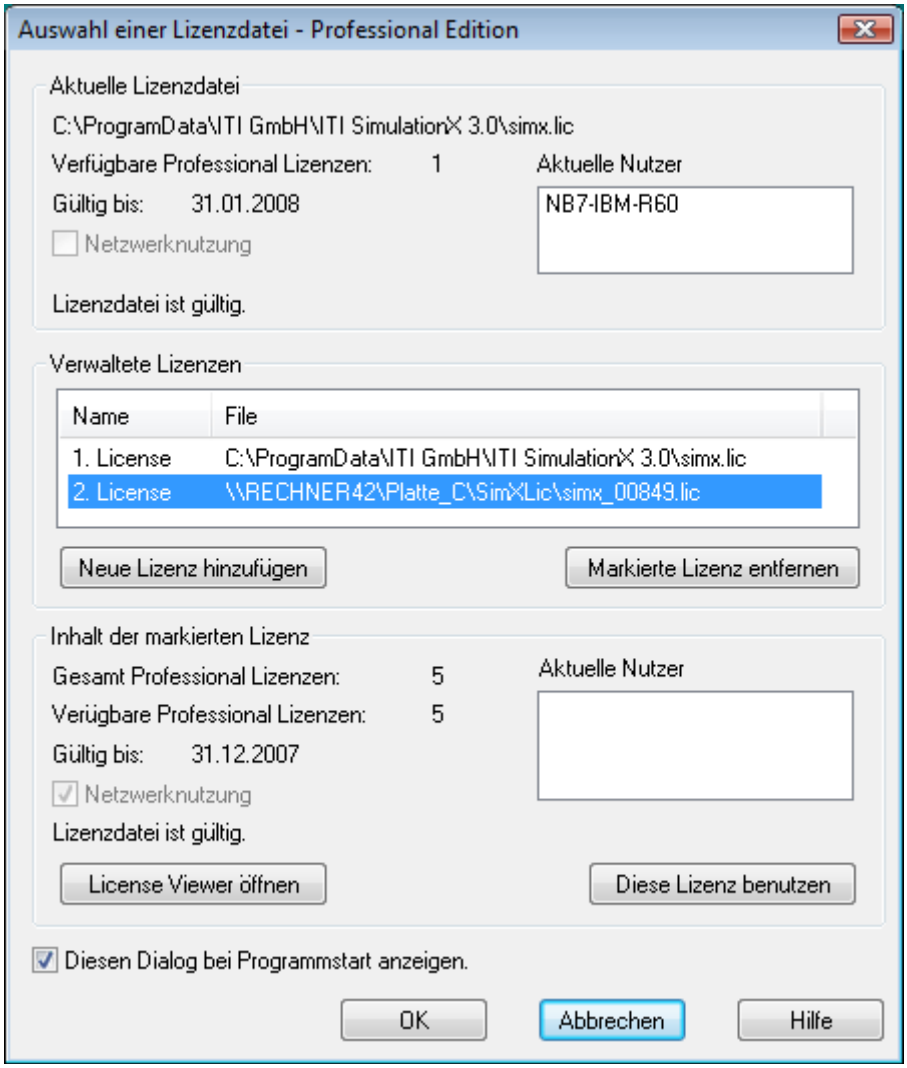



**Bild 2.29:** Dialog zur Auswahl einer Lizenzdatei

Der Dialog gliedert sich im Wesentlichen in drei Teile:

- Aktuelle Lizenzdatei,
- Verwaltete Lizenzen und
- Inhalt der markierten Lizenz.

Im Bereich **"Aktuelle Lizenzdatei"** werden wichtige Informationen wie

- Dateiname,
- verfügbare Lizenzen,
- Gültigkeitsdauer,
- Netzwerknutzung erlaubt,
- aktuelle Nutzer und
- eine Gültigkeitsinformation (s. **Tabelle 2.3**)
- Lizenzerneuerungsintervall bei Netzwerklizenzen

zur aktuell genutzten Lizenzdatei angezeigt.

| Gültigkeitsinformation | Bedeutung |
|---|---|
| "Lizenzdatei ist gültig" | Alles in Ordnung. Die Lizenzdatei kann genutzt werden. |
| "Lizenzdatei ist abgelaufen" | Die Gültigkeitsdauer der Lizenzdatei ist vorüber. Sie können diese Lizenzdatei nicht mehr nutzen. |
| "Lizenzdatei ist ungültig (falscher INST-Code)." | Die Lizenzdatei befindet sich auf einem Computer, für den sie nicht generiert wurde. Sie ist nicht nutzbar. |
| "Keine freie Lizenz vorhanden." | Alle Lizenzen dieser Datei werden aktuell genutzt. Die Nutzer werden wie folgt angezeigt (bei Einzelplatzlizenz - der Computername, bei Netzwerklizenz - Nutzer+++Computername). Wählen Sie eine andere Lizenzdatei aus. |

**Tabelle 2.4:** Gültigkeitsinformationen für Lizenzdateien

Im Bereich „**Verwaltete Lizenzen**“ werden alle Lizenzdateien, die Sie mit SimulationX nutzen wollen, aufgelistet. Für eine bessere Transparenz kann jede Lizenzdatei mit einem selbsterklärenden Namen versehen werden.

Die Lizenzdateiverwaltung **muss mindestens einen Eintrag enthalten** - Sie haben ja mindestens eine Lizenzdatei von ITI erhalten, welche hier hinzugefügt werden muss.

Der Verwaltung können Lizenzdateien hinzugefügt und von dieser entfernt werden.

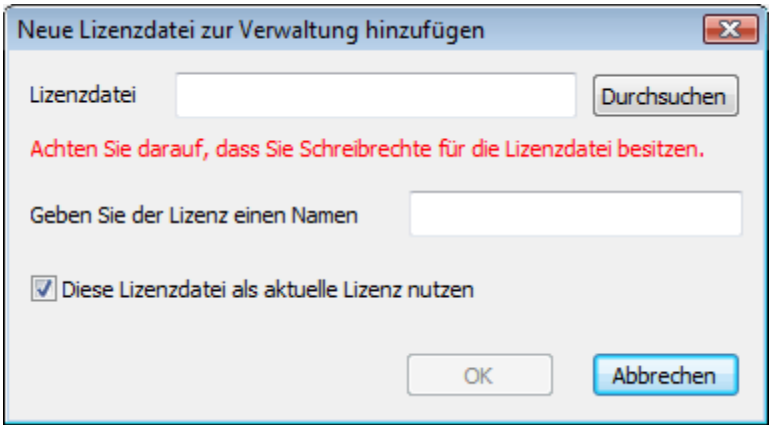



**Bild 2.30:** Neue Lizenzdatei der Verwaltung hinzufügen

Klicken Sie auf die Schaltfläche ***Neue Lizenz hinzufügen***. Ein Dialog (s. **Bild 2.29**) wird geöffnet. Klicken Sie auf ***Durchsuchen*** um die hinzuzufügende Lizenzdatei auszuwählen. Vergeben Sie einen aussagekräftigen Namen und setzen Sie das Häkchen für die Nutzung als aktuelle Lizenzdatei. Anschließend können Sie SimulationX starten (vorausgesetzt Sie haben eine gültige Lizenzdatei gewählt).

Wenn Sie in der Liste der verwalteten Lizenzen eine Zeile per Mausklick markiert haben, können Sie diese markierte Lizenz mit Klick auf die Schaltfläche ***Markierte Lizenz entfernen*** aus der Verwaltung löschen. Die Lizenzdatei selbst wird dabei **nicht gelöscht**.

Eine Editierung der verwalteten Lizenzdateien (beispielsweise die Änderung des Namens) ist nicht vorgesehen. In diesem Falle entfernen Sie die Datei aus der Verwaltung und fügen Sie sie neu hinzu.

Per Mausklick können Sie eine Zeile in der Liste der verwalteten Lizenzdateien markieren. Jetzt werden im Dialogbereich „**Inhalt der markierten Lizenz“** die wichtigsten Informationen - Gesamtanzahl Lizenzen, verfügbare Lizenzen, Gültigkeitsdauer, aktuelle Nutzer, Netzwerkfähigkeit und eine Gültigkeitsinformation (s. Tabelle 2.3) - der markierten Lizenzdatei angezeigt.

Klicken Sie auf die Schaltfläche ***License Viewer öffnen*** um den License Viewer Dialog für die markierte Lizenzdatei zu öffnen. Sie bekommen hier umfassende Informationen über die Lizenzdatei.

Die Schaltfläche ***Diese Lizenzdatei benutzen*** ist freigegeben, wenn die markierte Lizenzdatei eine gültige Lizenzdatei ist. Ein Klick auf die Schaltfläche setzt die Lizenzdatei als aktuell und sie wird zum Start von SimulationX genutzt.

Insbesondere in einem Netzwerk können mehrere Lizenzdateien zur Verfügung stehen. Mit Hilfe der Verwaltung kann man schnell zwischen verschiedenen Lizenzen wählen.

Soll der Auswahldialog nicht bei jedem Start von SimulationX geöffnet werden (z. B. bei einer Einzelplatzlizenz) ist das Häkchen „Diesen Dialog bei Programmstart anzeigen“ zu entfernen. Soll der Dialog wieder angezeigt werden, öffnen Sie über das Menü ***Extras/Optionen*** den Optionsdialog. Gehen Sie auf die Seite ***Lizenzierung***. Und hier kann man das Häkchen „Startup-Dialog zur Lizenzauswahl anzeigen“ wieder setzen.

Alle Informationen in diesem Dialog sind **nutzerbezogen**. D.h., jeder Nutzer dieses Computers verwaltet seine Lizenzdateien selbst.

## 2.9 Besonderheiten für Netzwerklizenzen

### 2.9.1 Netzwerklizenzen mit Dongle

Bei Netzwerklizenzen kommt es vor, dass alle zur Verfügung stehenden Lizenzen in Benutzung sind.
Als Anwender möchte man wissen, wer die Lizenzen gerade nutzt. Dazu gibt es bei Einsatz eines Dongles folgende Möglichkeit:

Auf dem Rechner, an dem der Dongle angesteckt ist, läuft außer dem Lizenz-serverdienst ein s. g. Lizenzmonitor. Diesen öffnen Sie in Ihrem Browser mit der Eingabe

http://RechnerMitDongle:6002

Dabei ist „RechnerMitDongle“ der Name oder die IP-Adresse des Rechners, an dem der Dongle steckt.

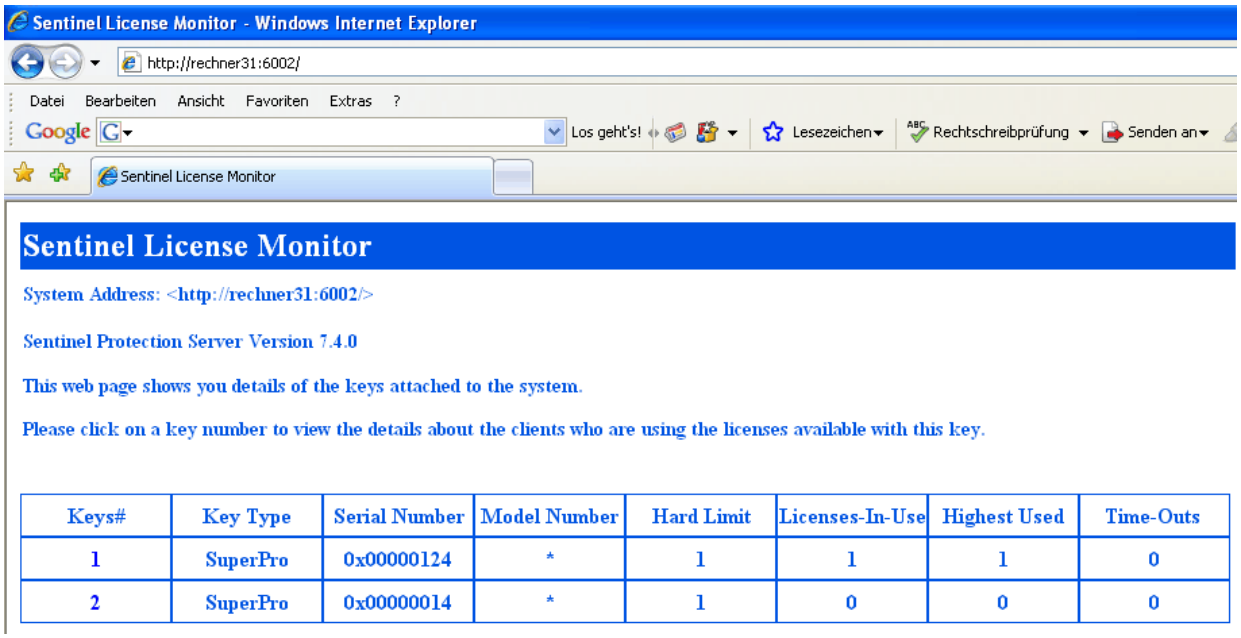



**Bild 2.31:** Sentinel Lizenzmonitor im Browser

Klickt man auf eine Key-Nummer, bekommt man weitere Informationen zu diesem Dongle (s. **Bild 2.31**).

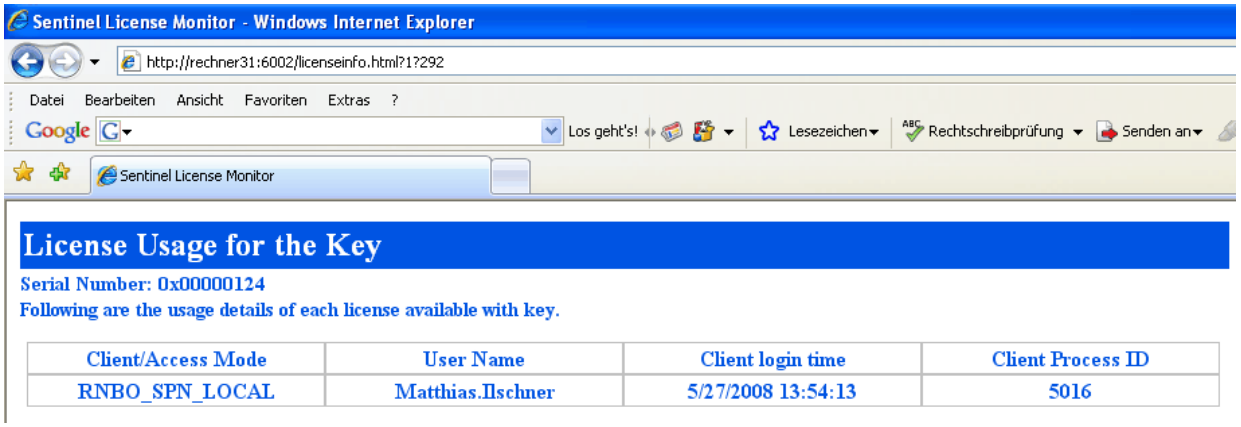



**Bild 2.32:** Dongleinformationen

## 2.9.2 Netzwerklizenzen mit Lizenzdatei

### 2.9.2.1 Lizenzerneuerungsintervall

Für Lizenzen die im Netzwerk genutzt werden, ist es notwendig, ein s. g. Lizenzerneuerungsintervall einzustellen. Das passiert beim Programmstart automatisch. Es werden Standardwerte genutzt.

Das Lizenzerneuerungsintervall sorgt dafür, dass verlorene Lizenzen von Bibliotheken und Optionen (etwa nach einem Programmabsturz) automatisch wieder freigegeben werden (s. **2.9.2.2**).

Wird die Lizenz in einem LAN oder WLAN Netzwerk genutzt, muss der Anwender im Normalfall KEINE WEITEREN EINSTELLUNGEN vornehmen.

Sollen Lizenzen in einem WAN (Wide Area Network) Netzwerk genutzt werden, kann es passieren, dass bei fehlender Bandbreite (zu langsames Netzwerk) ein Ressourcenproblem auftritt. D.h., noch während SimulationX startet und die Lizenzen für alle benötigten Module abfragt werden, ist das Lizenzerneuerungs-intervall um und die Lizenzabfrage für alle Module beginnt erneut. Dieses Problem äußert sich z.B. in einem sehr, sehr langsamen Startvorgang.

Um dieses Problem zu beheben, besteht die Möglichkeit, das Lizenzerneuerungs-intervall manuell einzustellen (siehe **Bild 2.32**).

Die Freigabe der Editbox erfolgt erst nach einem Klick auf die Editierungs-schaltfläche und der Bestätigung eines Warnhinweises.

ACHTUNG! Stellen Sie das Lizenzerneuerungsintervall nur so groß wie nötig ein.

Bedenken Sie, verlorene Lizenzen werden erst nach einer **Wartezeit von 150% des Lizenzerneuerungsintervalls** wieder freigegeben.

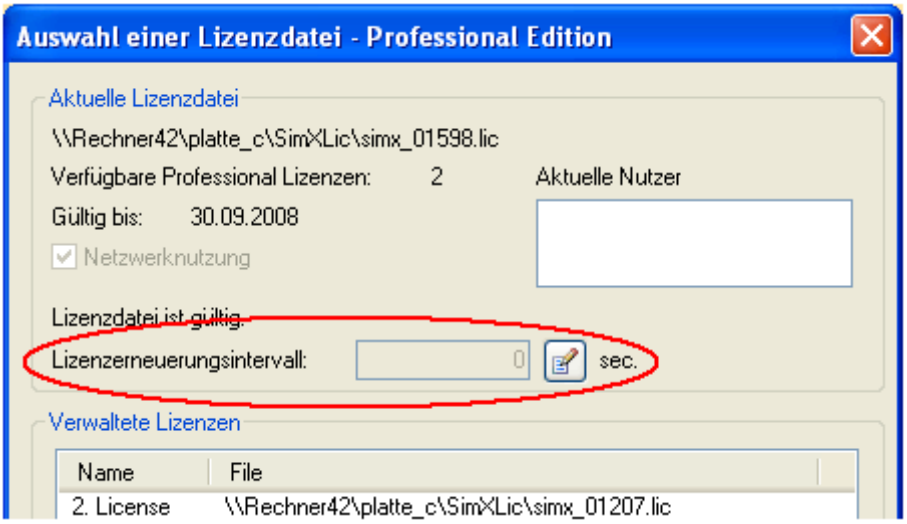



**Bild 2.33:** Eingabemöglichkeit für das Lizenzerneuerungsintervall

Lizenzerneuerungsintervall = 0 bedeutet, Standardwerte werden benutzt.

Standardwerte für das Lizenzerneuerungsintervall sind:

| | | | | |
|---|---|---|---|---|
| 1 | Liz. | = | 4 | sec |
| 5 | Liz. | = | 20 | sec |
| 10 | Liz. | = | 41 | sec |
| 20 | Liz. | = | 84 | sec |
| 50 | Liz. | = | 214 | sec |
| 100 | Liz. | = | 440 | sec |
| 500 | Liz. | = | ca. 40 | min |
| 1000 | Liz. | = | ca. 87 | min |
| 2000 | Liz. | = | ca. 190 | min |

Wie kann ein brauchbares Lizenzerneuerungsintervall in etwa ermittelt werden? P sei die Laufzeit einer Lizenzanfrage über das Netzwerk zwischen Lizenzserver und entferntem Computer. Eine Lizenz besteht aus q Modulen (im License Viewer durch zählen zu ermitteln), für welchen je eine Lizenzanfrage erforderlich ist. Alle Lizenzanfragen sind innerhalb des Lizenzerneuerungsintervalls abzuarbeiten, d.h.

Lizenzerneuerungsintervall > p*q seconds.

Falls von dem Lizenzserver mehrere Lizenzen verwaltet werden, ist das Lizenzerneuerungsintervall noch mit der Lizenzanzahl zu multiplizieren.

#### 2.9.2.2 Verwaiste Lizenzeinträge in der Lizenzdatei

Folgende Ausführungen gelten nur für Lizenzdateien **mit mehr als einer Lizenz**.

Netzwerklizenzen können temporär inaktiv sein, werden aber innerhalb des Lizenzerneuerungsintervall (Herzschlagfunktion) immer wieder aktiviert (s. **2.9.2.1**). Bei Programmstart wird eine freie Lizenz für den „Professional"- bzw. „Analyzer"-Modul gesucht. Hierbei müssen natürlich inaktive Lizenzen mit berücksichtigt werden, damit einem im Moment gerade inaktiven Nutzer nicht seine Lizenz „entzogen" wird. Das würde zum Chaos führen, hat aber den Nachteil, dass die Lizenz eines in der inaktiven Phase abgestürzten Programms nicht wieder genutzt werden kann.

Startet ein Nutzer sein abgestürztes Programm neu, wird überprüft, ob sein Eintrag ev. noch in der Eintragsliste des „Professional"- bzw. „Analyzer"-Moduls vorhanden ist. Wenn ja, bekommt er seine Lizenz sofort zurück. Wenn nicht, wird eine freie Lizenz gesucht.

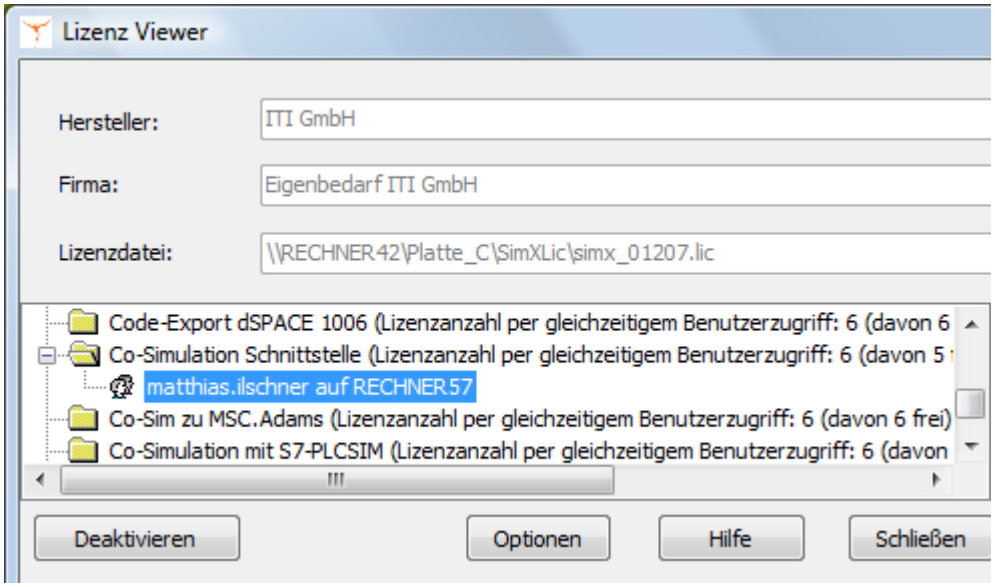
Langzeituntersuchungen haben gezeigt, dass folgende fehlerhafte Situationen auftreten können:

- in Bibliotheken und Optionen bleiben nach Programmende/-absturz Lizenzeinträge von aktiven Nutzern zurück
- im „Professional"- bzw. „Analyzer"-Modul bleiben nach einem Programmabsturz Lizenzeinträge von inaktiven Nutzern zurück

Beide Situationen können bereinigt werden.

- Lizenzeinträge von aktiven Nutzern, welche gar nicht mehr aktiv sind, werden mit dem License Viewer deaktiviert.

  Öffnen Sie dazu den License Viewer entweder vom Lizenzauswahldialog für die zu bearbeitende Lizenzdatei (s. **2.8**) oder für die aktuelle Lizenzdatei über das Menü ***Extras/Optionen/Lizenzierung***.




Markieren Sie den zu deaktivierenden Eintrag und klicken Sie auf ***Deaktivieren***. Der Eintrag wird in Klammern angezeigt (inaktiv) und steht für weitere Nutzung wieder zur Verfügung.

ACHTEN Sie darauf, dass nur tatsächlich nicht mehr genutzte Einträge deaktiviert werden.

- Lizenzeinträge von inaktiven Nutzern im „Professional“- bzw. „Analyzer“-Modul werden mit dem Dialog zum „Aufräumen“ der Lizenzdatei entfernt.

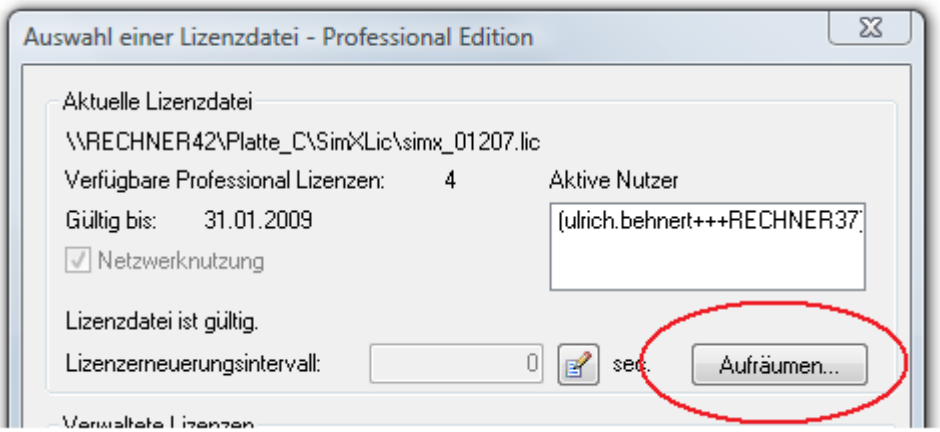



**Bild 2.34:** Startdialog mit "Aufräumen" Schaltfläche

Die folgenden Schritte sind nur möglich, wenn kein weiterer Nutzer aktiv ist. Wenn es ausschließlich inaktive Nutzer (Anzeige von Einträgen in Klammern) gibt, wird im Startdialog die ***Aufräumen*** Schaltfläche angezeigt (s. **Bild 2.33**).
Klicken Sie auf die entsprechende Schaltfläche wird der Dialog zum „Aufräumen“ der Lizenzdatei geöffnet (s. **Bild 2.34**)

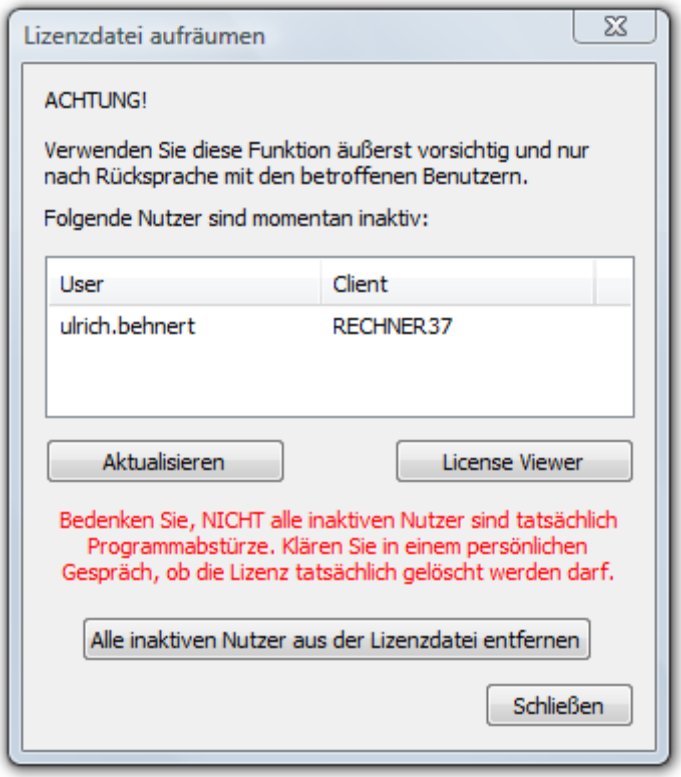



**Bild 2.35:** Dialog zum "Aufräumen" der Lizenzdatei

Da sich über der Zeit die aktiven und inaktiven Nutzer ändern können, kann mit der Schaltfläche ***Aktualisieren*** der gerade aktuelle Stand abgerufen werden. Mit der "License Viewer" Schaltfläche wird der License Viewer für die aktuelle Lizenzdatei geöffnet.

Wenn Sie sich sicher sind, dass die aktuell angezeigten inaktiven Nutzer aus der Lizenzdatei entfernt werden sollen, klicken Sie auf die entsprechende Schaltfläche. Danach stehen die Lizenzen wieder zur Nutzung zur Verfügung.

SIMULATION X®

Powered by ITI

Getting Started

# Getting Started

# 3 Erste Schritte

Dieses Kapitel soll Ihnen einen Schnelleinstieg in die Arbeit mit SimulationX vermitteln. In einer allgemeinen Darstellung wird auf die Nutzung mitgelieferter Modelle eingegangen. Am Beispiel eines Zwei-Massenschwingers wird der Aufbau eines Modells Schritt für Schritt erklärt und kann am eigenen Rechner nachvollzogen werden. Detaillierte Kenntnisse zur Bedienung und zum Modellaufbau erhalten Sie im **Kapitel 5 "Modellierung"**. Die im Folgenden verwendeten Begriffe sind in **Kapitel 12 "Begriffsdefinitionen"** erklärt.

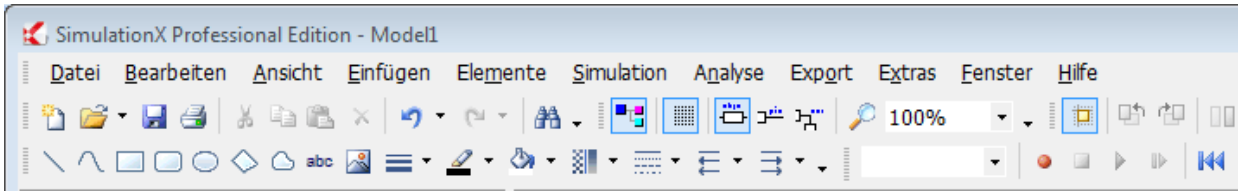
## 3.1 Arbeiten mit Beispielmodellen

Mit der Installation von SimulationX haben Sie auch eine Sammlung von Beispielen für die Modellierung erhalten. Die vorgefertigten Modelle können Sie öffnen, Parameter ändern und Simulationen durchführen. Abgelegt sind diese im Verzeichnis (*C:\Benutzer\Öffentlich\Öffentliche Dokumente\SimulationX 3.5\Samples*).

Die Willkommen-Seite beim Starten von SimulationX erlaubt einen direkten Zugriff auf die Beispiele.

Beispiele, die Elemente aus für Sie nicht lizenzierten Bibliotheken enthalten, können Sie mit der **Viewer-Edition** einsehen. Das Umschalten in die Viewer-Edition ist in **Kapitel 2.2.8** beschrieben.

### 3.1.1 Öffnen von Modellen




Das Öffnen von Beispiel-Modellen ist über das Menü ***Datei/Öffnen...*** oder die Schaltfläche möglich.

So öffnen Sie ein vorhandenes Modell

1. Klicken Sie im Menü ***Datei*** auf Öffnen.
2. Wählen Sie im Dialogfeld ***Öffnen*** das entsprechende Verzeichnis (…\SimulationX 3.5\Samples) und das zu öffnende Modell aus.
3. Klicken Sie auf ***Öffnen***.

Nach dem Laden erscheint die Modellstruktur, eine kurze übersichtliche Modellbeschreibung, sowie eventuell abgespeicherte Ergebnisfenster mit den berechneten Zeitverläufen. Des Weiteren sind teilweise Datenblätter, Basisdaten und erklärende Abbildungen in den Beispielen enthalten.

### 3.1.2 Ändern von Parametern

Vor der Modifikation von Parametern sollte ein bereits gerechnetes Modell zuerst zurückgesetzt werden (Betätigen der Schaltfläche oder Aufrufen der Option ***Rücksetzen*** im Menü ***Simulation***).

Doppelklick auf das Symbol eines Elementes öffnet dessen Eigenschaftsfenster. Auf der Registerseite ***Parameter*** können Sie in die Eingabefelder Zahlen, mathematische Ausdrücke und/oder logische Operatoren eintragen.

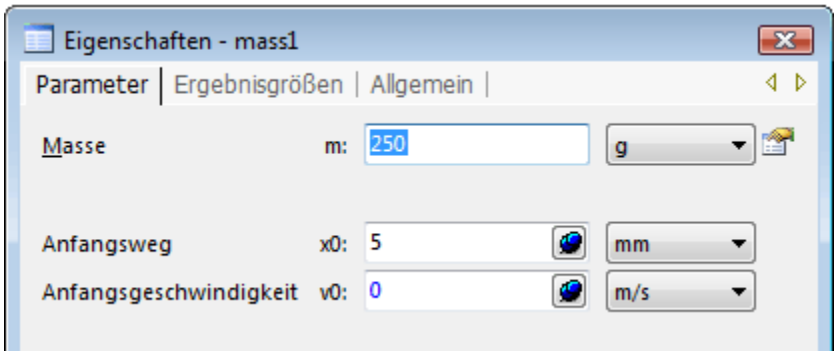



**Bild 3.1:** Registerseite „Parameter“

So ändern Sie Parameter eines Elements

1. **Doppelklick** auf ein Element. Es öffnet sich der Eigenschaftendialog.
2. Wählen Sie die Registerkarte ***Parameter*** aus.
3. Ändern Sie den Wert oder die Maßeinheit eines Parameters.
4. Ein Klick außerhalb des Eingabefeldes übernimmt den Wert

Hinweise zu den Elementen, deren Eigenschaften und Parameter finden Sie in der ***Online-Hilfe***.
Eine Änderung der Einheit führt zu einer automatischen Umrechnung des Zahlenwertes!

Um Rechenergebnisse zu protokollieren, ist die Aktivierung des Protokollattributes der entsprechenden Ergebnisgröße auf der Registerseite ***Ergebnisgrößen*** notwendig.

**Bild 3.2:** Registerseite „Ergebnisgrößen“

So aktivieren Sie ein Protokollattribut

1. **Doppelklick** auf ein Element. Es öffnet sich der Eigenschaftendialog.
2. Wählen Sie die Registerkarte ***Ergebnisgrößen*** aus.
3. Klicken Sie auf das Symbol für das Ergebnisprotokoll um es ein oder auszuschalten.

### 3.1.3 Simulation starten

Über das Menü ***Simulation/Start*** oder mit der Schaltfläche wird die Berechnung gestartet. Die Simulation erfolgt nun bis zur vorgegebenen Endzeit. Zur Änderung der Endzeit, setzen Sie die Simulation zurück und öffnen den Dialog für die Simulationssteuerung (Menü ***Simulation/Einstellungen...***). Jetzt können Sie in der Registerseite ***Allgemein*** die Werte für die Simulation (z. B. `tStop`) individuell anpassen (s. **Bild 3.8**).

### 3.1.4 Ergebnisfenster öffnen

Wurde im Register ***Ergebnisgrößen*** des Eigenschaftsfensters eines Elementes das Protokollattribut für Ergebnisgrößen aktiviert, so können diese Ergebnisse der Simulation grafisch dargestellt werden.

So öffnen Sie ein Ergebnisfenster

1. **Klicken** Sie mit der **rechten Maustaste** auf ein Element.
2. Wählen Sie unter Ergebnisgrößen die entsprechende Variable aus. Es öffnet sich das Ergebnisfenster.

Die ausgewählte Ergebnisgröße wird als *y=f(t)*-Diagramm dargestellt.

Werden keine Ergebnisgrößen angezeigt, so wurde kein Protokollattribut aktiviert ( ).

Im **Kapitel 7.3 "Ergebnisfenster"** wird die Arbeit mit diesen detailliert erklärt. Mehrere Ergebnisgrößen sind in einem Fenster darstellbar und die *y(x)*-Darstellung von Wertepaaren ist möglich.

## 3.2 Erstellen von eigenen Modellen

Die folgenden Abschnitte erläutern den Modellaufbau Schritt für Schritt anhand eines Zwei-Massenschwingers. Zur Vertiefung verweisen wir jeweils auf die entsprechenden detaillierten Abschnitte.

Zuerst sollten Sie sich ein leeres Arbeitsblatt verschaffen (Schaltfläche oder Menü ***Datei/Neu***).

### 3.2.1 Auswahl der Elemente des Modells

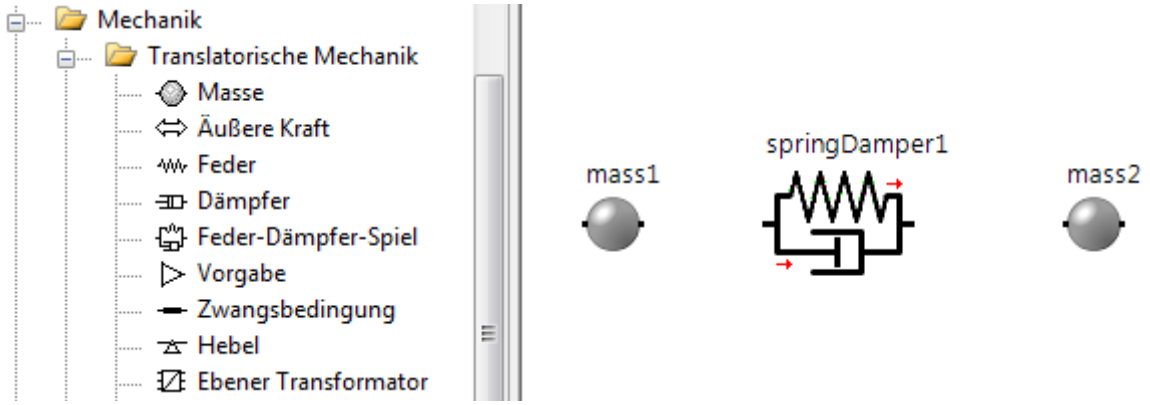



**Bild 3.3:** Bibliotheksleiste und Modell

Auf der linken Seite des Bildschirmes befindet sich die Bibliotheksleiste (**Bild 3.3**). Unterbibliotheken können mittels Mausklick geöffnet werden.

Möchten Sie eines der Elemente benutzen, ziehen Sie es einfach mittels Drag & Drop auf das Arbeitsblatt (ausführlich siehe **Kapitel 5.2.3.1**).

Die erzeugten Elemente können auf dem Arbeitsblatt verschoben, gedreht oder gespiegelt werden.

Ein Doppelklick auf ein Element öffnet sein Eigenschaftsfenster (Details in **5.3.1**).

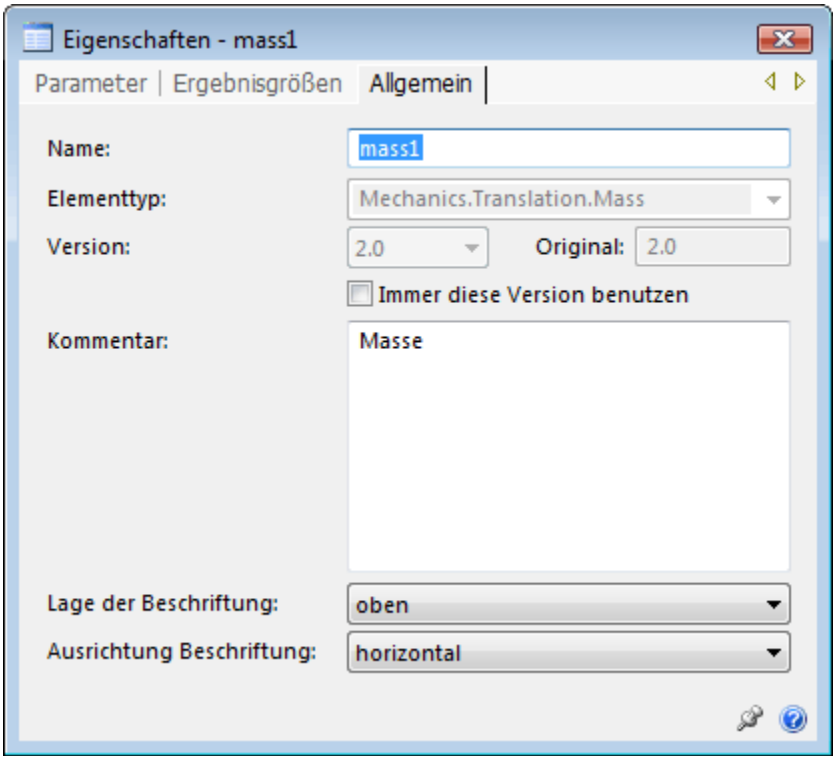



**Bild 3.4:** Eigenschaften von „Mass1“

Auf der Seite ***Allgemein*** können Sie den Namen, einen Kommentar, sowie die Lage und Ausrichtung der Beschriftung ändern.

Beachten Sie dass der Name ein eindeutiger Bezeichner sein und einer bestimmten Syntax gehorchen muss (siehe **Kapitel 10**). Als Kommentar sind beliebige Eingaben erlaubt.

**Aufgabe:**
Erzeugen Sie aus der Bibliothek Translatorische Mechanik zwei Massen und ein Feder-Dämpfer-Spiel Element, so dass folgendes Modell entsteht:

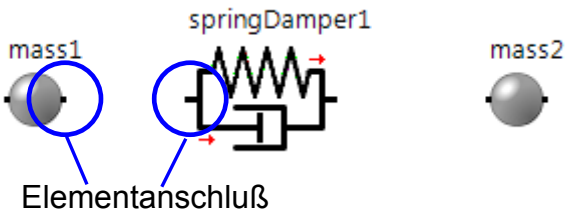



**Bild 3.5:** Modell

### 3.2.2 Verbinden der Elemente

Haben Sie alle Elemente auf dem Arbeitsblatt angeordnet, können Sie zur gewünschten Modellstruktur verbunden werden (Details in **5.2.3.1**). Zum Verbinden bewegen Sie den Mauszeiger über einen Elementanschluss. Der Anschluss färbt sich dann rot und der Mauszeiger schaltet um. Ziehen Sie mit gedrückter Maustaste die Verbindung zum anderen Element. Ein schwarzer Kreis signalisiert, dass Sie sich an einer passenden Stelle befinden.

Beachten Sie, dass nur Elemente mit gleichartiger physikalischer Domäne (also z. B. mechanische Feder nicht mit hydraulischer Drossel) verbunden werden können.
Andernfalls erfolgt kein Aufbau einer Verbindung am anderen Element!
Für derartige Verbindungen stehen spezielle Elemente zur Verfügung, die jeweils über beide Anschlussarten verfügen (z. B. Rotatorisch-Translatorische Transformation in der Mechanik).

Die entstehenden Verbindungen sind auch Komponenten des Modells. Sie können Parameter und Ergebnisgrößen besitzen.

SimulationX ist mit einem Algorithmus zur automatischen Verbindungsführung (Autorouting) ausgestattet. Beim Verschieben von Elementen werden die angeschlossenen Verbindungen neu gezogen. **Kapitel 5.2.3.2** beschreibt, wie Sie die Verbindungsführung selbst beeinflussen können.

**Beispiel:**
Verbinden Sie die Elemente zur fertigen Modellstruktur.

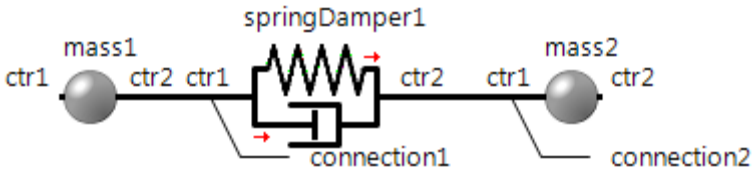



**Bild 3.6:** Modellstruktur

### 3.2.3 Parametereingabe

Parameter geben Sie im Eigenschaftsdialog (Doppelklick auf das Element) oder im Modellexplorer (detaillierte Informationen in **Kapitel 5.3)** ein. Der Modellexplorer befindet sich im unteren Bildschirmbereich (**Bild 3.7**).

Er zeigt das Modell als Baumansicht, sowie die Parameter und Ergebnisgrößen in getrennten Tabellen an. Alle Parameter haben neben ihrem Wert auch eine physikalische Einheit. Der Wert wird umgerechnet, wenn eine andere Einheit ausgewählt wird (Details in **5.3.2**).

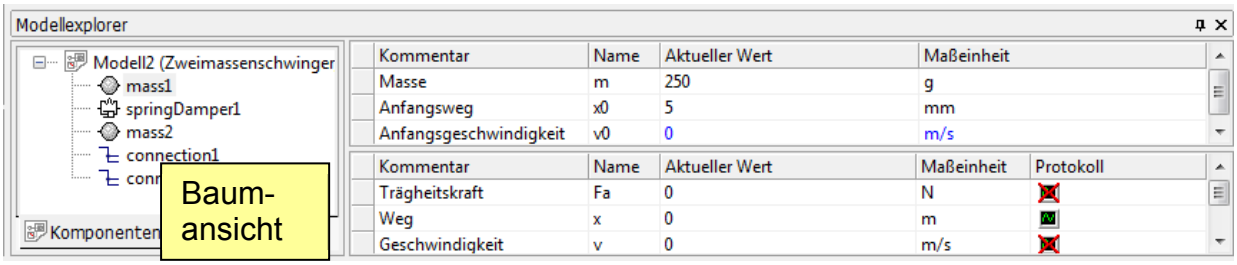



**Bild 3.7:** Parameter im Modellexplorer

Beachten Sie, dass als Dezimaltrennzeichen ein Punkt (kein Komma!) eingegeben werden muss.

**Beispiel:**
Geben Sie den Parametern folgende Werte:

| | | | | |
|---|---|---|---|---|
| Mass1 | Masse | m | 250 | g |
| | Anfangsweg | x0 | 5 | mm |
| Mass2 | Masse | m | 2 | kg |
| SpringDamper1 | Steife Anlage | k | 0.25 | N/mm |
| | Dämpfung Anlage | b | 2 | Ns/m |

**Tabelle 3.1:** Parameter

Beachten Sie, dass Anfangswerte mit festen Vorgaben fixiert werden müssen. Drücken Sie dazu auf die Schaltfläche des Anfangswertes im Eigenschaftsfenster (Details siehe **5.3.3**). Anschließend ist der Anfangswert fixiert.

Soll der Zeitverlauf von Ergebnisgrößen während bzw. nach der Simulation ausgewertet werden, müssen Sie entsprechende Protokollattribute setzen. Nur Ergebnisgrößen mit Protokollattribut werden während der Rechnung aufgezeichnet und können in Ergebnisfenstern dargestellt werden. Schalten Sie dazu in der Ergebnisgrößentabelle durch Anklicken mit der Maus die Schaltfläche auf um.

**Beispiel:**
Aktivieren Sie das Protokollattribut folgender Ergebnisgrößen:

- Mass1 Weg x
- Mass2 Weg x
- SpringDamper1 Fi
- SpringDamper1 dx

## 3.2.4 Simulation im Zeitbereich

Auf das nun fertige Beispielmodell können alle SimulationX Berechnungsmethoden angewendet werden (siehe **Kapitel 6**):

- Simulation im Zeitbereich
- Berechnung des Gleichgewichtszustandes
- Lineare Modellanalyse (Eigenfrequenzen und Schwingformen).

Wir beschränken uns im Folgenden auf die Simulation über der Zeit. **Bild 3.8** zeigt den Dialog zur Simulationssteuerung (Menü ***Simulation/Einstellungen***).

Hier kann das Verfahren gewählt und verschiedene Parameter zur Schritt-weiten- und Genauigkeitssteuerung eingestellt werden (Details siehe **6.1.1**). Für unser Modell müssen Sie keine Änderungen vornehmen.

Gestartet wird die Simulationsrechnung mit der Schaltfläche . Mit kann die Rechnung jederzeit angehalten werden. setzt das Modell wieder in den Anfangszustand zurück.

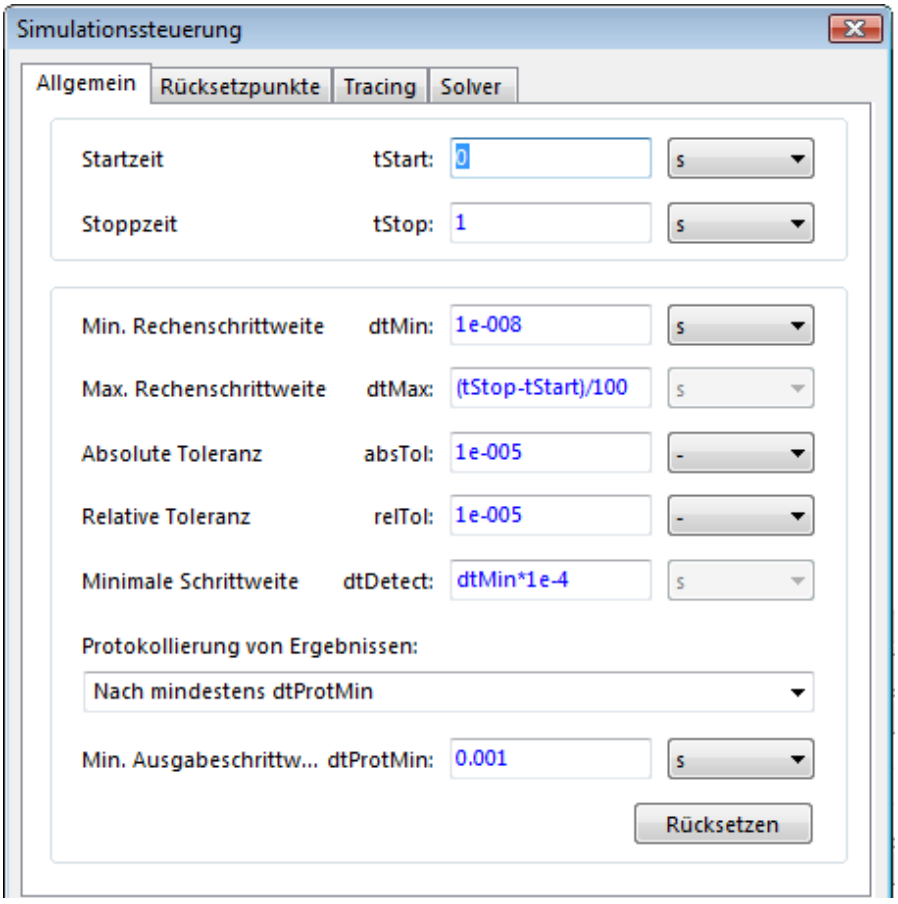



**Bild 3.8:** Eigenschaftsfenster „Simulationssteuerung“

**Beispiel:**
Berechnen Sie das Modell bis zu einer Endzeit von 1 s.

### 3.2.5 Ergebnisauswertung

Die Ergebnisse der Simulation werden in Ergebnisfenstern dargestellt. Um das entsprechende Ergebnisfenster anzuzeigen, nutzen Sie die Drag & Drop – Funktion, wie sie im **Kapitel 7.3.3** beschrieben wird.

**Beispiel:**
Stellen Sie die Ergebnisgröße Mass1.x in einem *y = f(t)* Diagramm dar:

Das Element "*Mass1*" wird durch Anklicken in der Strukturansicht ausgewählt. Anschließend gehen Sie zum Register „*Ergebnisgrößen*". Per Drag & Drop ziehen Sie das Protokollattribut für den Weg von "*Mass*1" in die Strukturansicht. Beim Loslassen der *Maustaste* wird das Ergebnisfenster (**Bild 3.9**) sichtbar.

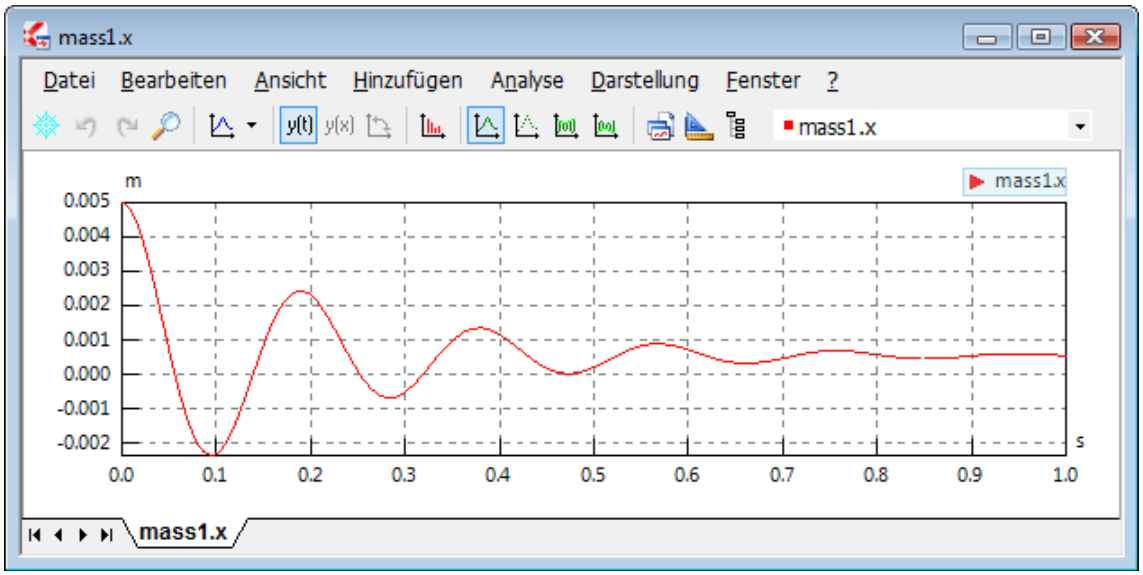



**Bild 3.9:** Ergebnisfenster für „*Mass1.x*"

Auch die Anzeige mehrerer Ergebnisgrößen in einem Fenster ist möglich.

**Beispiel:**
Stellen Sie im vorhandenen Ergebnisfenster zusätzlich die Ergebnisgröße „*Mass2.x*" dar:

Das Element "*Mass2*" auswählen und das Protokollattribut für den Weg per Drag & Drop in das vorhandene Ergebnisfenster ziehen (**Bild 3.9**).

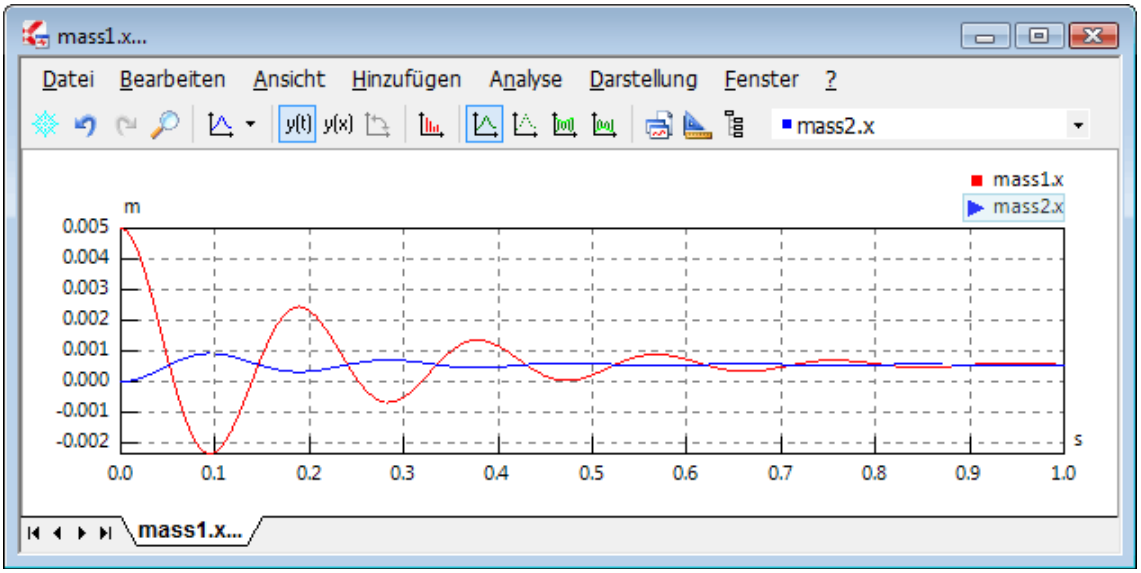



**Bild 3.10:** Darstellung mehrerer Ergebnisse

Zwei Ergebnisgrößen sind in einem *y(x)-Diagramm* darstellbar.

**Beispiel:**
Es soll die Wegdifferenz (*dx)* über der Inneren Kraft*(Fi)* des Feder-Dämpfer-Elements dargestellt werden. Dazu erstellen Sie ein Ergebnisfenster mit den entsprechenden zwei Ergebnisgrößen über der Zeit und schalten mit y(x) in die y(x)Darstellung um.

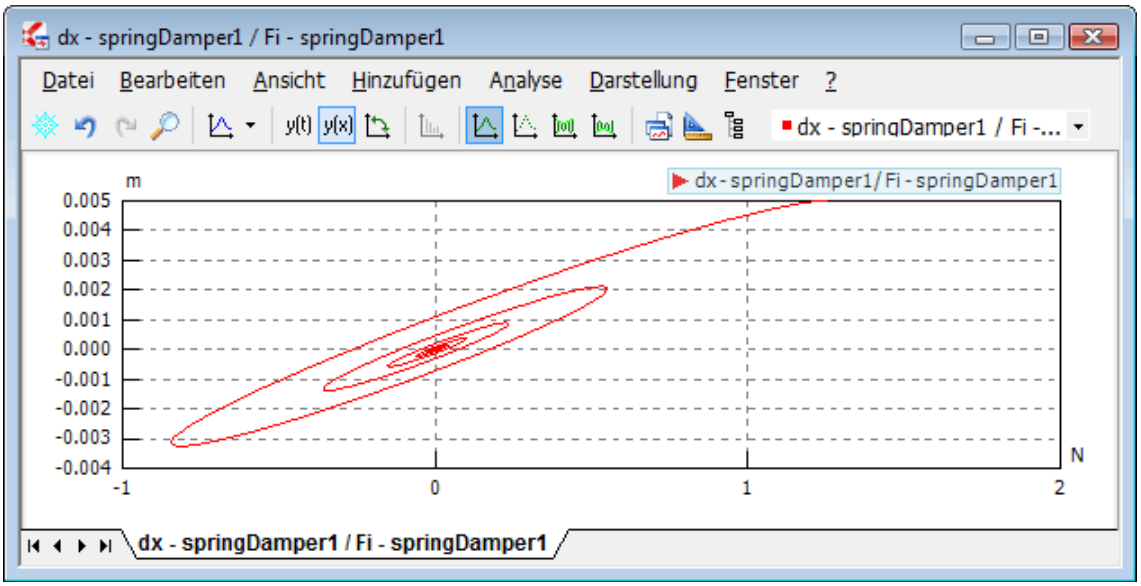



**Bild 3.11:** y(x)-Diagramm

Sie können die Simulation nun mit der Schaltfläche ⏮ zurücksetzen, Parameter- oder Strukturänderungen am Modell vornehmen und die Simulation erneut starten. Beim Rücksetzen werden alle Ergebniskurven ebenfalls zurückgesetzt.

Wenn Sie vor dem Zurücksetzen die Ergebniskurve mit der Schaltfläche einfrieren, so bleibt die Kurve beim Rücksetzen erhalten und Sie können die Wirkung Ihrer Parameteränderung nach der nächsten Simulation vergleichen!

## 3.2.6 Stationäre Simulation im Frequenzbereich

Im folgenden Abschnitt werden Sie anhand eines einfachen Beispiels in die stationäre Simulation eingeführt. Eine detaillierte Beschreibung erfolgt in **Kapitel 7.3**.

### 3.2.6.1 Stationäre Simulation bei einem Antriebsstrang

Erstellen Sie das in **Bild 3.12** gezeigte SimulationX-Modell eines Antriebsstrangs. Bitte entnehmen Sie den Abschnitten **3.2.1** bis **3.2.3** die grundlegenden Arbeitsschritte für die Modellerstellung.

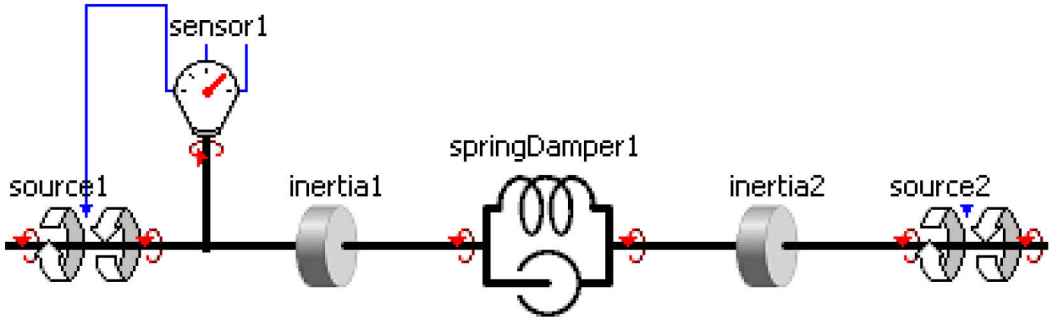



**Bild 3.12:** SimulationX-Modell eines einfachen Antriebsstrangs für die stationäre Simulation

Die im Modell zu ändernden Parameter sind in **Tabelle 3.2** zusammengefasst.

| | | | | |
|---|---|---|---|---|
| springDamper1 | Eigenschaften | kind | Feder-Dämpfer | |
| | Steife Anlage | k | 20000 | Nm/rad |
| | Dämpfung Anlage | b | 10 | Nms/rad |
| source1 | Moment | T | siehe unten | N |

**Tabelle 3.2:** Parameter für den Antriebsstrang

Das Erregermoment wird im Parameterdialog von *source1* auf den Wert

*source1.T = 1000*exp(-10*(cos(0.5* in1)+1))*

gesetzt. Dieser in **Bild 3.13** dargestellte Momentenverlauf ist mit zwei Umdrehungen des Stranges periodisch, so wie das auch bei einem Einzylinder-Viertaktmotor der Fall ist.

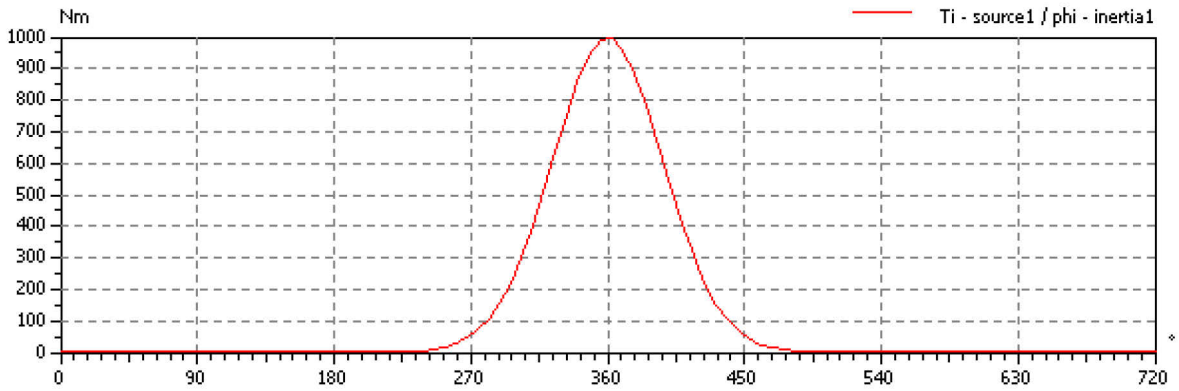



**Bild 3.13:** Erregermoment des Antriebsstranges

Die Momentenquelle *source2* dient dem Ausgleich des mittleren Antriebsmoments von *source1*. Mit ihr wird sichergestellt, dass die mittlere Strangdrehzahl konstant bleibt und sich eine stationäre Lösung einstellen kann. Den Parameter *Moment* dieser Quelle können Sie auf dem Standardwert *in1* stehen lassen. Die stationäre Analyse berechnet das notwendige Ausgleichsmoment selber.

Vor der stationären Simulation aktivieren Sie die Protokollattribute der interessierenden Ergebnisgrößen. Der Handlungsablauf ist kurz in Abschnitt **3.2.3** erläutert. In diesem Beispiel benötigen wir die folgenden Ergebnisgrößen:

- das Beschleunigungsmoment `Ta` von *inertia1*,
- die Winkelgeschwindigkeit `om` von *inertia1*.

Stellen Sie die Auswahlbox in der Menüleiste auf Stationär und öffnen Sie den Einstellungsdialog der stationären Simulation durch Drücken des Startsymbols oder über den Menüpunkt ***Stationäre Simulation*** im Menü ***Analyse***.

Für dieses Beispiel sind (wie in vielen Anwendungsfällen) nur Einstellungen auf der in **Bild 3.14** zu sehenden ***System***-Seite des Einstellungsdialoges erforderlich.

Wählen Sie aus der Auswahlbox für die *Bezugsgröße* die Winkelgeschwindigkeit `inertia1.om` aus und geben Sie für den *Start*- und den *Stopp*wert der Bezugsgröße 100rad/s bzw. 300rad/s ein. Damit legen Sie fest, dass die stationäre Simulation die stationär eingeschwungenen Zustände für alle mittleren Drehzahlen von inertia1 im Intervall von 100rad/s bis 300rad/s berechnen soll.

Der Drehwinkel `inertia1.phi` wird automatisch als *Periodenvariable* selektiert.
Den Standardwert $4\pi$ für die Periodenlänge können Sie in diesem Beispiel beibehalten, da das erregende Moment mit $4\pi$ im Drehwinkel *inertia1.phi* periodisch ist. Die Werte für die *Grundordnung* und die *höchste Ordnung* belassen Sie auch auf ihren Standardwerten. Die Grundordnung wird der Schwingung mit der ange-

gebenen *Periodenlänge* also $4\pi$ zugeordnet. Die Schwingung erster Ordnung hat dann die Periodenlänge $2\pi$ und so weiter.

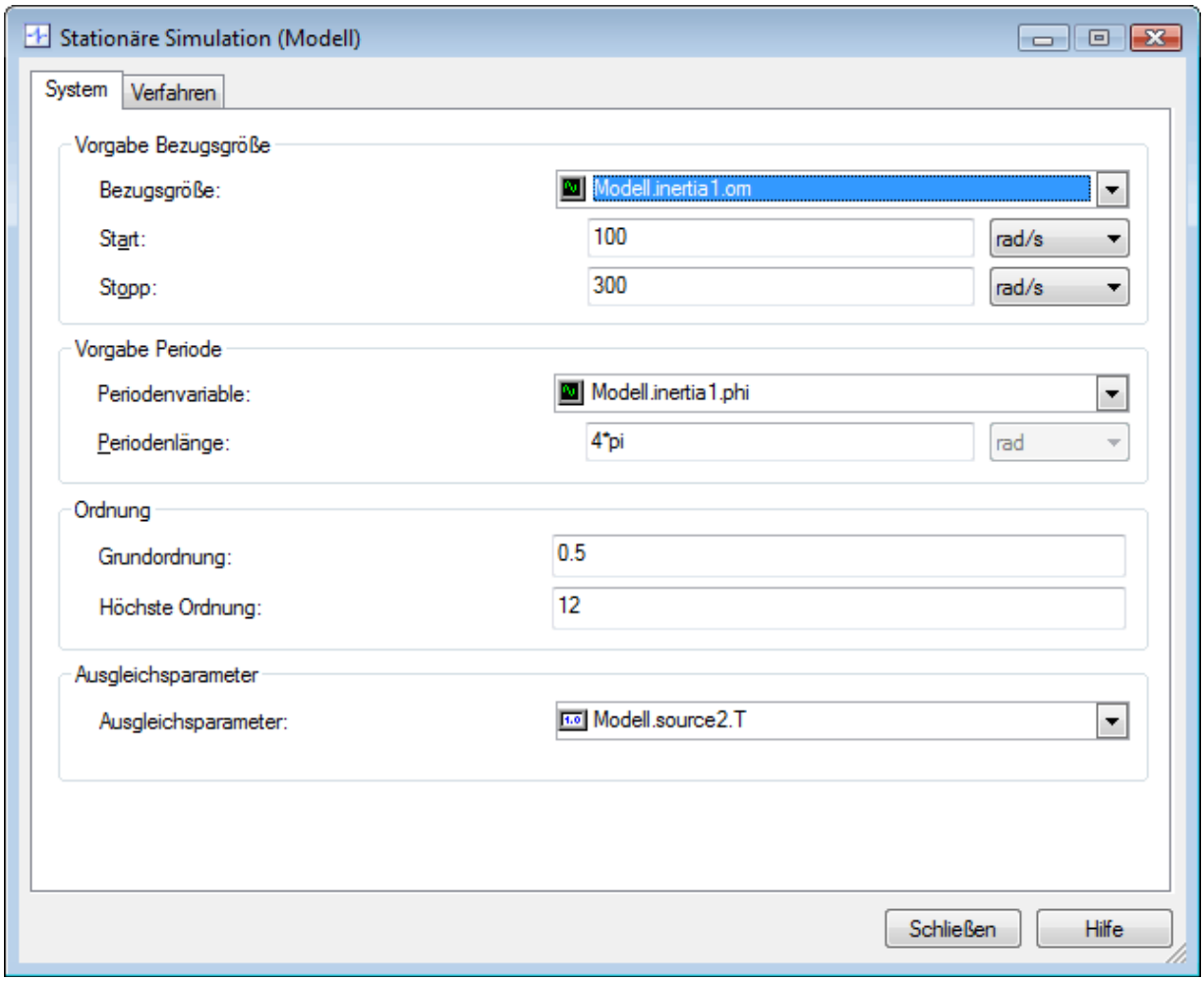



**Bild 3.14:** Einstellungsdialog für die stationäre Analyse

Als Ausgleichsparameter wählen Sie das Moment *source2.T*. Mit dem Ausgleichsparameter kompensiert die stationäre Simulation das mittlere Antriebsmoment von *source1.T* so, dass die über eine Periode gemittelte Drehzahl konstant bleibt.

Mit Drücken des Startknopfes

des Einstellungsdialoges läuft die stationäre Simulation an. Während der Simulation können Sie sich bereits Ergebnisverläufe anzeigen lassen. Klicken Sie dazu die Ergebnisgröße *Drehgeschwindigkeit om* im Untermenü ***Ergebnisgrößen (stationär)*** des Kontextmenüs der Drehmasse *inertia1* an (siehe **Bild 3.15**).

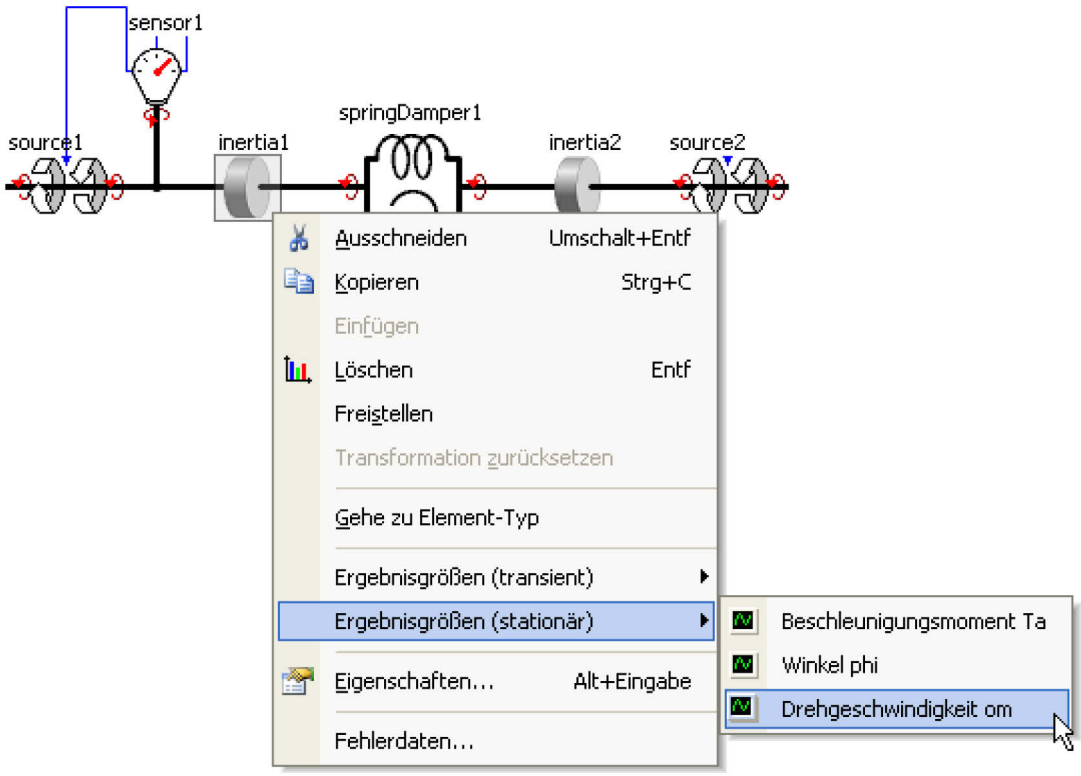



**Bild 3.15:** Auswahl einer protokollierten Ergebnisgröße der stationären Simulation

Es öffnet sich das in **Bild 3.16** zu sehende Ergebnisfenster für *inertia1.om*. Mit den Schaltflächen ☑ können Sie einzelne Kurven ein- und ausschalten. In **Bild 3.16** wurde bereits die *Mittelwert*komponente abgeschaltet um die wesentlich kleineren Spektralkomponenten besser sichtbar zu machen.
Standardmäßig wird bei den Drehgeschwindigkeiten der Drehmassen der *Ungleichförmigkeitsgrad* dargestellt. Alternativ dazu können Sie in der Auswahlbox Amplitudenverlauf, Erregung, Phase, Realteil oder Imaginärteil wählen.

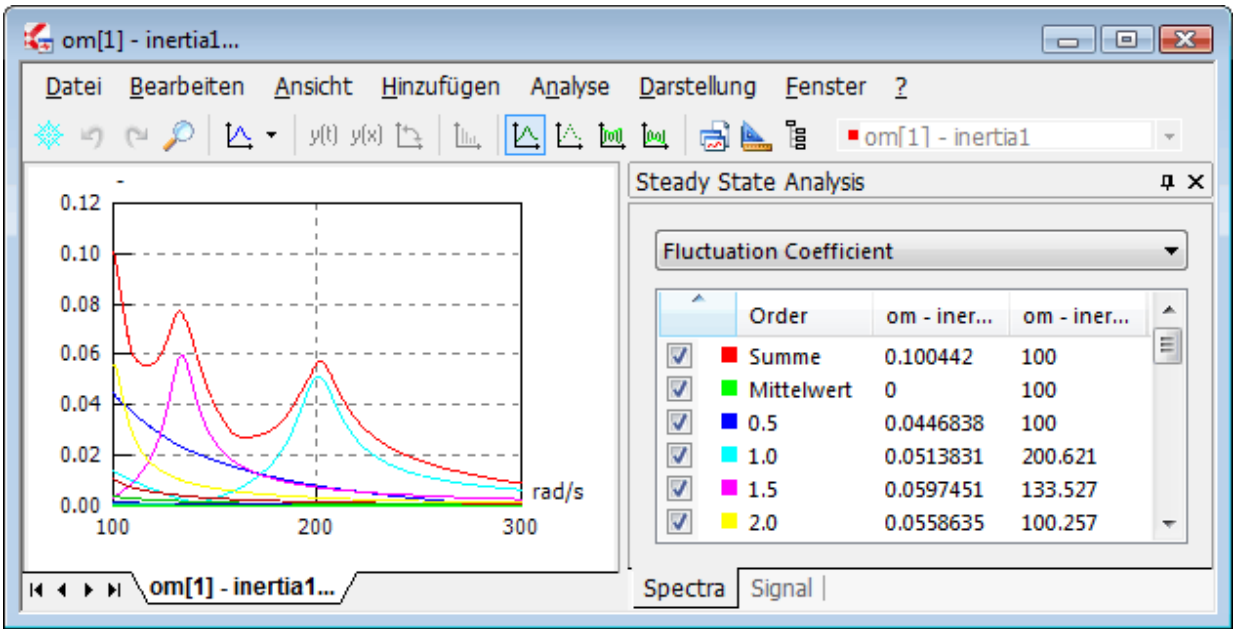



**Bild 3.16:** Ergebnisfenster für die Ergebnisgröße *inertia1.om*

Mit den rechts unten im Ergebnisfenster befindlichen Reitern ***Spektren/Verläufe*** Spektren | Verläufe schaltet man zwischen der Darstellung der Ergebnisse im Frequenz- und Zeitbereich um.

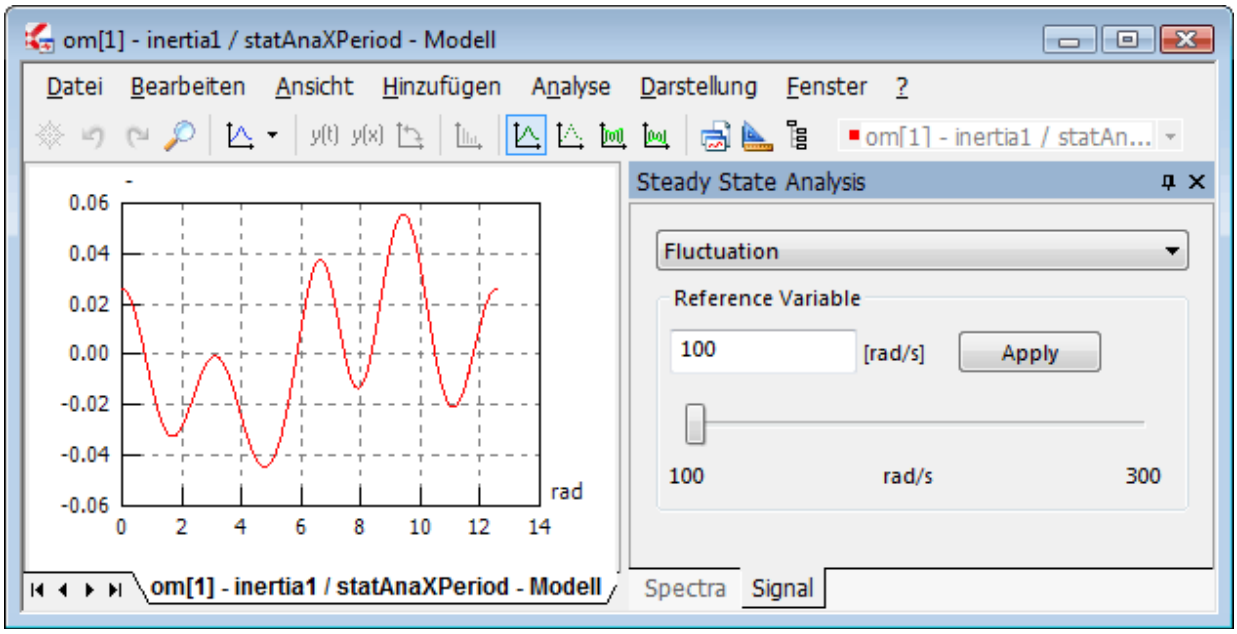



**Bild 3.17:** Darstellung des Ergebnisverlaufes im Zeitbereich über dem Winkel

Klickt man auf Verläufe, so wechselt das Panel im Ergebnisfenster zur Form in **Bild 3.17**. Die Auswahlbox bietet die Darstellung von Auslenkung, Auslenkung+ Mittelwert und Ungleichförmigkeit an. Bei der Drehgeschwindigkeit von Drehmassen ist die Ungleichförmigkeit die Standardeinstellung.
Mit dem Schieberegler lässt sich der Wert der Bezugsgröße (in unserem Beispiel der Mittelwert von *inertia1.om*), für den die Zeitbereichsergebnisse dargestellt werden, ändern. Sie können den Wert der Bezugsgröße auch in das Eingabefeld schreiben und die Eingabe dann mit einem Klick auf die Taste *Übernehmen* bestätigen.

## 3.3 SimulationX für die ganzheitliche Systemmodellierung und -analyse

Nutzen Sie die vielfältigen Möglichkeiten von SimulationX für eine schnelle und effiziente Problemlösung sowie bei der Beurteilung und Optimierung technischer Anlagen:

- transiente und stationäre Simulation (siehe **6.1** und **6.3**)
- Parameterstudien (Schaltfläche oder Menü ***Analyse / Variantenassistent ...***., siehe **6.6)**
- Lineare Systemanalyse: Eigenfrequenzen und Schwingformen (Schaltfläche oder Menü ***Analyse / Eigenfrequenz ...***, siehe **6.5.1**)
- Lineare Systemanalyse: Übertragungsverhalten (Schaltfläche oder Menü ***Analyse/Übertragungsverhalten***,siehe **6.5.2**)
- Erweitern vorhandener Elemente (Schaltfläche oder **Elemente/Ableiten** – siehe **8.2.3**)
- Erstellen von Submodellen (Compounds) (Schaltfläche oder **Elemente/Zusammenfassen** – siehe **8.2.4**)
- Erstellung eigner Typen, Fluida mit dem **TypeDesigner/FluidDesigner** (**Kapitel 8.3)**
- Einbindung von eigenen Algorithmen mit dem TypeDesigner (siehe **8.2.6**)
- Simulatorkopplung ...( siehe **9.3**)
- Code-Export (Schaltfläche oder Menü Export) für beschleunigte Simulationsrechnung („ausführbares Modell“) oder Modellintegration (siehe **9.4**).

SIMULATION X®

Powered by ITI



# Graphical User Interface

# 4 Bedienoberfläche

## 4.1 Übersicht

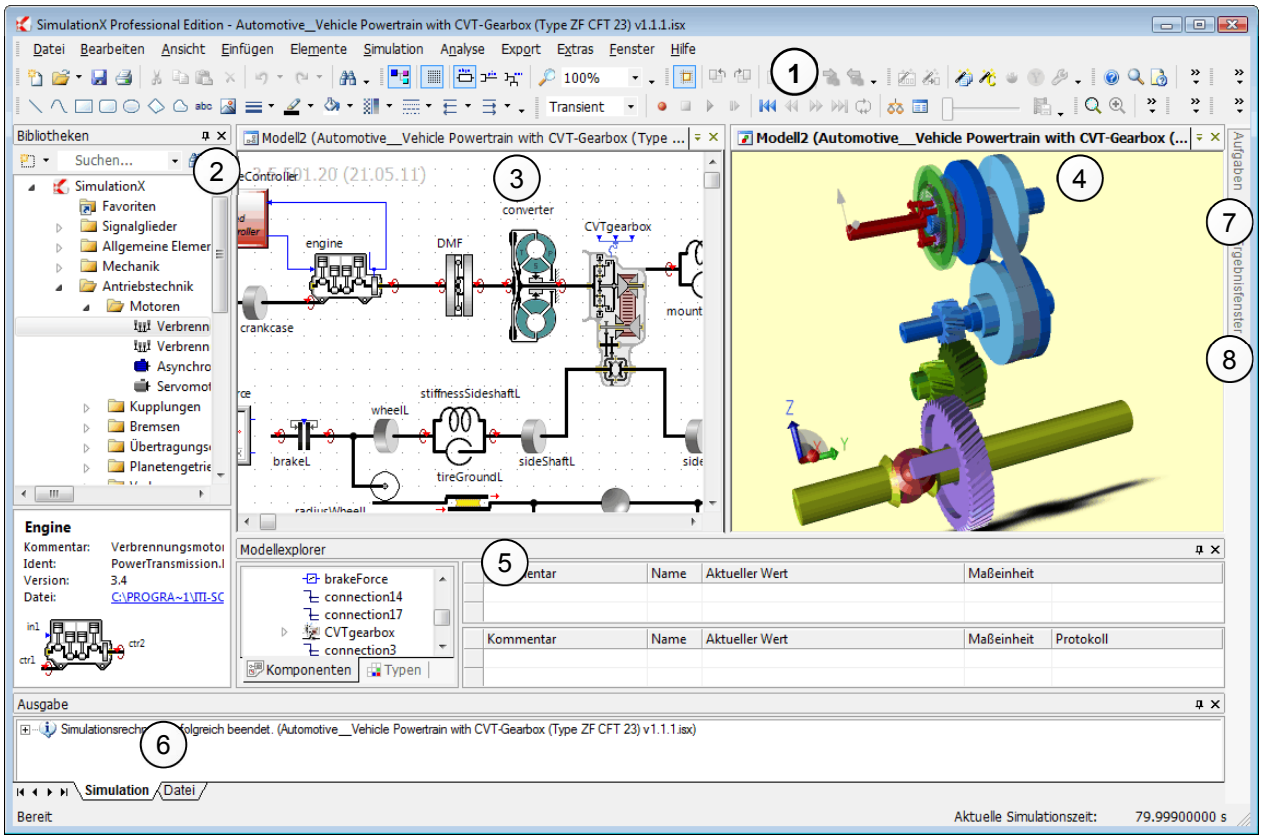



**Bild 4.1:** SimulationX-Arbeitsbereich:
(1) Menü- und Symbolleisten, (2) Bibliotheksleiste, (3) Strukturansicht
(4) 3D-Ansicht, (5) Modellexplorer, (6) Ausgabebereich,
(7) Aufgabenbereich, (8) Ergebnisfenstermanager

Die Programmoberfläche von SimulationX ist in verschiedene Fenster und Bereiche unterteilt, die nachfolgend beschrieben werden.

Die SimulationX-Programmoberfläche enthält zwei Hauptfenstertypen:

- Toolfenster
- Dokumentenfenster

Diese beiden Fenstertypen weisen leichte Verhaltensunterschiede auf.

### 4.1.1 Toolfenster

Toolfenster werden im Menü Ansicht aufgelistet und können unterschiedlich dargestellt werden:

- Automatisch ein- oder ausgeblendet
- Angedockt an die Ränder der SimulationX-Programmoberfläche
- Nicht angedockt („schwebend“)

### 4.1.2 Dokumentenfenster

Dokumentenfenster werden dynamisch erstellt, wenn Sie Modelle öffnen oder erstellen. Die Liste der geöffneten Dokumentenfenster wird im Menü Fenster angezeigt. Registerkartengruppen bieten Ihnen die Möglichkeit, den beschränkten Arbeitsbereich besser zu nutzen, wenn Sie mit zwei oder mehr geöffneten Modellansichten arbeiten. Sie können mehrere Dokumentenfenster zu vertikalen oder horizontalen Registerkartengruppen zusammenfassen und auch problemlos Ansichten von einer Registerkartengruppe in eine andere verschieben.

Weitere Anpassungen können über das Menü ***Ansicht*** vorgenommen werden. Hier können Sie in einen Vollbild-Modus (Ganzer Bildschirm) wechseln, der den gesamten Bildschirm zur Darstellung der Dokumentenfenster nutzt.

## 4.2 Bedienung

### 4.2.1 Menü- und Symbolleisten

Die Menü- und Symbolleisten (1) erlauben den Zugriff auf alle Werkzeuge und Befehle von SimulationX. Sie können das Layout, die Position und den Inhalt von Symbolleisten ändern. Sie können den Symbolleisten Schaltflächen hinzufügen und das zugewiesene Symbol für eine beliebige Schaltfläche ändern.

So fügen Sie eine neue Symbolleiste hinzu

1. Klicken Sie im Menü ***Extras*** auf ***Anpassen***.
2. Wählen Sie im Dialogfeld ***Anpassen*** die Registerkarte ***Symbolleisten*** aus.
3. Wählen Sie ***Neu*** aus.
4. Geben Sie im Dialogfeld ***Neue Symbolleiste*** einen Namen für ***Symbolleiste*** ein.
5. Klicken Sie auf ***OK***.

So fügen Sie einer Symbolleiste eine Befehlsschaltfläche hinzu

1. Klicken Sie im Menü ***Extras*** auf ***Anpassen***.
2. Wählen Sie im Dialogfeld ***Anpassen*** die Registerkarte ***Symbolleisten*** aus.
3. Wählen Sie die anzupassende Symbolleiste aus, und aktivieren Sie diese.
4. Wählen Sie die Registerkarte ***Befehle*** aus.
5. Wechseln Sie in ***Kategorien*** zu der Kategorie, die den Befehl enthält, der der Symbolleiste hinzugefügt werden soll.
6. Wählen Sie in ***Befehle*** ein Element aus.
7. Ziehen Sie das Element unter ***Befehle*** aus dem Dialogfeld ***Anpassen*** auf die gewünschte Symbolleiste.
8. Wenn vom Mauszeiger ein Pluszeichen (+) angezeigt wird, legen Sie das Element auf der Symbolleiste ab.

So ändern Sie Symbole für Symbolleisten-Schaltflächen

1. Klicken Sie im Menü ***Extras*** auf ***Anpassen***.
2. Wählen Sie im Dialogfeld ***Anpassen*** die Registerkarte ***Symbolleisten*** aus.
3. Wählen Sie die Symbolleiste mit dem zu ändernden Schaltflächensymbol aus, und aktivieren Sie diese.
4. Klicken Sie bei geöffnetem Dialogfeld ***Anpassen*** auf die zu ändernde Schaltfläche. Ein schwarzes Quadrat oder Rechteck zeigt an, dass die Schaltfläche zur Bearbeitung markiert ist.
5. Wählen Sie die Registerkarte ***Befehle*** aus.
6. Wählen Sie ***Auswahl ändern*** aus.
7. Wählen Sie ***Schaltflächensymbol ändern*** und anschließend im Untermenü das gewünschte Bild aus. Wenn die verfügbaren Bilder nicht Ihren Vorstellungen entsprechen, können Sie ein ausgewähltes Bild des Systems anpassen.

So bearbeiten Sie Schaltflächensymbole

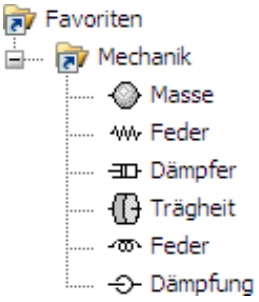
1. Klicken Sie im Menü ***Extras*** auf ***Anpassen***.
2. Wählen Sie im Dialogfeld ***Anpassen*** die Registerkarte ***Symbolleisten*** aus.
3. Wählen Sie die Symbolleiste mit dem zu ändernden Schaltflächensymbol aus, und aktivieren Sie diese.
4. Klicken Sie bei geöffnetem Dialogfeld ***Anpassen*** auf die zu ändernde Schaltfläche. Ein schwarzes Quadrat oder Rechteck zeigt an, dass die Schaltfläche zur Bearbeitung markiert ist.
5. Wählen Sie die Registerkarte ***Befehle*** aus.
6. Wählen Sie ***Auswahl ändern*** aus.
7. Wählen Sie ***Schaltflächensymbol ändern*** aus.
8. Passen Sie im Dialogfeld ***Schaltflächen-Editor*** das Symbol für die ausgewählte Schaltfläche an.

Im Dialogfeld ***Anpassen*** können auch Menüs angepasst werden.

### 4.2.2 Bibliotheksleiste

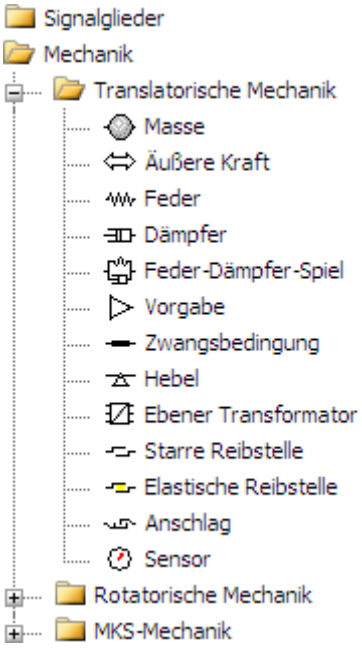
Die Bibliotheksleiste (2) dient der Auswahl, Verwaltung und Bearbeitung von Elementtypen. Logisch zusammengehörige Elementtypen werden in Gruppen zusammengefasst, die, vergleichbar mit der Ordneransicht im Windows-Explorer, hierarchisch gegliedert in einer Baumansicht dargestellt werden. Die in der Bibliotheksleiste enthaltenen Ordner lassen sich in folgende Gruppen aufteilen:

## Favoriten




Unter Favoriten können per *Drag & Drop* Verknüpfungen zu häufig benutzten Elementtypen oder Bibliotheken abgelegt werden. Das Anlegen neuer Gruppen gestattet, unabhängig von der tatsächlichen Namenshierarchie der enthaltenen Elementtypen, den Aufbau einer angepassten Ordnerstruktur. Der Inhalt des Ordners Favoriten wird für jeden Anwender individuell in der Datei *usersettings.mo* im Verzeichnis für die Anwendungsdaten des Benutzers gespeichert (*z.B. C:\Benutzer\<user>\ AppData\Roaming\ITI GmbH\ SimulationX 3.5*).

## SimulationX – Bibliotheken

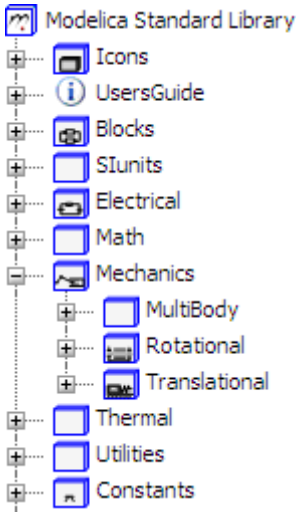



Innerhalb dieser Ordner werden alle im Lieferumfang von SimulationX enthaltenen und lizenzierten Bibliotheken und Elementtypen angezeigt. Die hier enthaltenen Elementtypen können weder verändert noch gelöscht werden. Sie können jedoch die Eigenschaften eines Elementtyps auf einen neuen Elementtyp übertragen, indem Sie eine Ableitung des vorhandenen Typs erstellen (siehe **Kapitel 8**). Der so erstellte neue Elementtyp kann im TypeDesigner bearbeitet werden. Wird eine bestimmte SimulationX-Bibliothek nicht benötigt, kann Sie vom automatischen Laden zum Programmstart ausgeschlossen werden.

So schließen Sie eine Bibliothek vom automatischen Laden zum Programmstart aus:

1. Klicken Sie im Menü ***Extras*** auf ***Optionen***.
2. Wählen Sie im Dialogfeld ***Optionen*** die Registerkarte ***Bibliotheken & Erweiterungen*** aus.
3. Entfernen Sie die Auswahlmarkierung für die Bibliothek, die Sie vom automatischen Laden zum Programmstart ausschließen wollen.

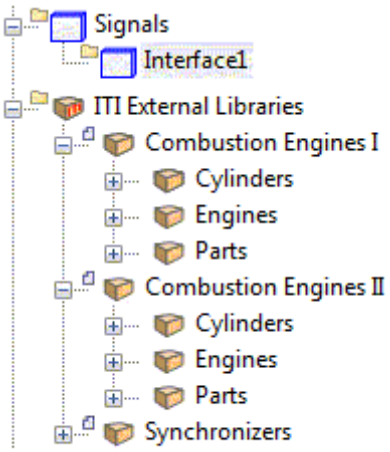
## Modelica Standard Bibliothek




Unter diesem Ordner finden Sie den Inhalt der Modelica Standard Bibliothek. Die darin enthaltenen Modelica-Typen können Sie im TypeDesigner öffnen und sich so einen Überblick über die darin enthaltenen Komponenten sowie den Modelica-Quellcode verschaffen. Sie können die Elementtypen jedoch weder verändern noch löschen.

Hinweise zur Installation sind in **Kapitel 10.2.1** zu finden.

## Weitere Modelica Bibliotheken




Unter diesem Ordner finden Sie den Inhalt weiterer unter dem Modelica-Suchpfad installierter Bibliotheken. Zu dieser Kategorie gehören auch die ITI External Libraries (z.B. Combustion Engines, Synchronizer).

Die in diesem Ordner enthaltenen Gruppen und Elementtypen können umbenannt, per *Drag & Drop* verschoben oder kopiert, im TypeDesigner bearbeitet oder auch gelöscht werden. Die Bibliotheken können Sie durch neue Typen erweitern. Eine neue Modelica-Bibliothek erstellen Sie, indem Sie in der Baumansicht der Bibliotheksleiste den Wurzeleintrag „ITI SimulationX“ markieren und über die Auswahlbox in der Symbolleiste oder das Kontextmenü als neue Modelica Klasse „Package“ hinzufügen. Zur weiteren Bearbeitung des Packages wird automatisch der TypeDesigner geöffnet. Auf die gleiche Art und Weise können Sie das so erstellte Package um neue Modelica-Klassen erweitern.

**Externe Typen**
Unter diesem Ordner finden Sie alle benutzerdefinierten Elementtypen, die in den Verzeichnissen für Externe Typen gespeichert sind. Die in diesem Ordner enthaltenen Gruppen und Elementtypen können umbenannt, per *Drag & Drop* verschoben oder kopiert, im TypeDesigner bearbeitet oder gelöscht werden. Neue Gruppen, Elementtypen oder andere Modelica-Klassen erstellen Sie, in dem Sie in der Baumansicht der Bibliotheksleiste zunächst die Gruppe auswählen, die den neu zu erstellenden Typ enthalten soll. Über das Kontextmenü können Sie dann den gewünschten Typ hinzufügen. Zur weiteren Bearbeitung wird der TypeDesigner geöffnet (siehe **Kapitel 8.2)**, oder im Fall einer neuerstellten Gruppe, ein entsprechendes Dialogfeld angezeigt.

### 4.2.2.1 Allgemeine Hinweise zur Bibliotheksleiste

Als Beschriftung der einzelnen Einträge in der Baumansicht wird standardmäßig der Kommentar angezeigt. Für die SimulationX-Bibliotheken steht dieser Kommentar in der jeweiligen Anzeigesprache (deutsch oder englisch) zur Verfügung.
Für die Elemente der Modelica Standard Bibliothek wird hingegen anstelle des Kommentars der Name des Typs angezeigt.

Mit der Schaltfläche können die Anzeige der Symbole und das Verhalten beim Anklicken geändert werden.

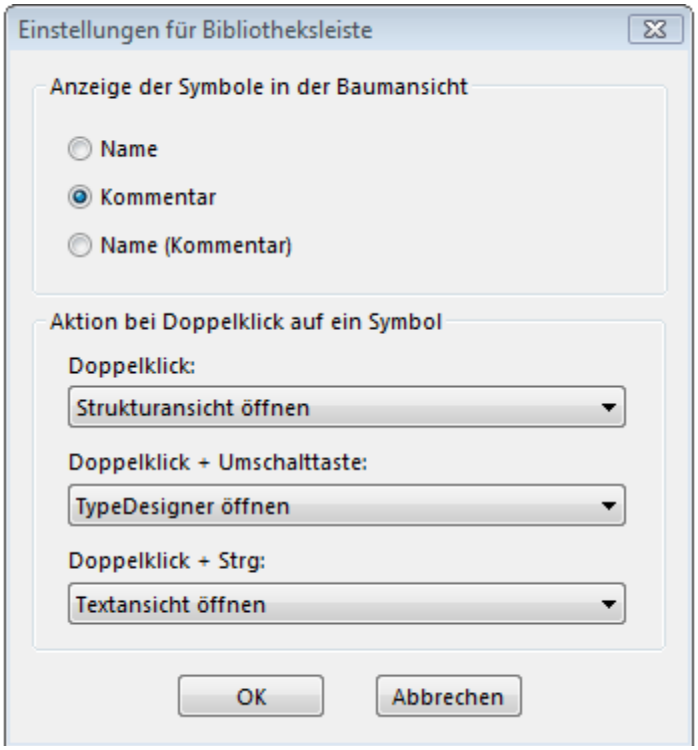



**Bild 4.2:** Einstellungsdialog für die Biliotheksleiste

In der Symbolleiste der Bibliotheksleiste befindet sich ein Eingabefeld zur Suche von Elementtypen. Nach Betätigung der Schaltfläche ***Suchen*** wird die Baum-

ansicht, beginnend mit dem aktuell ausgewählten Elementtyp, nach der eingegebenen Zeichenkette durchsucht. Bei der Suche werden sowohl Name als auch Kommentar der Typen berücksichtigt. Bei erfolgreicher Suche wird der gefundene Eintrag selektiert. Mit der erneuten Betätigung der Schaltfläche ***Suchen*** kann die Suche fortgesetzt werden.
Die Symbole für die Elementtypen in der Baumansicht können durch eine zusätzliche Kennzeichnung besonders hervorgehoben sein:

**Speicherung des Elementtyps als Datei**
Der Elementtyp wird als Datei gespeichert. Weitere in diesem Typ enthaltene Typen werden ebenfalls in dieser Datei gespeichert. Den zugehörigen Dateinamen finden Sie im Vorschaufenster am unteren Rand der Bibliotheksleiste.

**Speicherung einer Gruppe oder eines Packages als Verzeichnis**
Das Package bzw. die Gruppe wird als *package.mo* Datei in einem separaten Verzeichnis gespeichert. Die innerhalb dieses Packages deklarierten Typen können in separaten Dateien oder innerhalb der package.mo Datei selbst gespeichert werden. Innerhalb des Packages enthaltene Packages können auch als Verzeichnis gespeichert werden.

**Verschlüsselter Elementtyp oder Gruppe**
Der Elementtyp ist verschlüsselt und kann nur nach Eingabe des korrekten Passworts bearbeitet werden. Verschiedene Modelica-Tools verwenden im Allgemeinen jeweils unterschiedliche Methoden der Verschlüsselung.

**Gesperrter Elementtyp oder Modelica Klasse**
Der Elementtyp oder die Modelica-Klasse ist im TypeDesigner oder der Strukturansicht geöffnet. Eine weitere Bearbeitung ist erst nach Schließen des Typedesigners bzw. der entsprechenden Strukturansicht möglich.

Über ein Kontextmenü stehen nachfolgende Befehle für den jeweils ausgewählten Eintrag der Baumansicht zur Verfügung. Die am Rand dargestellten Symbole geben Auskunft darüber, auf welche Elementtypen der jeweilige Befehl prinzipiell anwendbar ist. Die Verwendbarkeit eines Befehls kann jedoch von weiteren Bedingungen abhängen, was im Text gesondert erwähnt wird.

**Neu…**
Verwenden Sie diesen Befehl, um einen neuen Typ der angegebenen Modelica-Klasse zu erstellen. Die Befehle **Model, Connector, Record, Block, Function und Package** stehen nur unter *Externe Typen* zur Verfügung. Der neuangelegte Typ wird zur Bearbeitung im TypeDesigner geöffnet. Für ein neuerstelltes Package sind in einem Dialogfeld Name und Kommentar sowie, wenn erforderlich, das

jeweilige Zielverzeichnis einzugeben. Es kann später im TypeDesigner wie ein Modelica-Package bearbeitet werden.

**Löschen**

Verwenden Sie diesen Befehl um den ausgewählten Typ einschließlich der darin enthaltenen Typen zu löschen. Beachten Sie, dass dabei auch die Datei bzw. das Verzeichnis gelöscht wird, in dem der jeweilige Typ gespeichert ist. Eine entsprechende Warnung müssen Sie vor dem Löschen bestätigen. Soll der Typ nur temporär aus der Baumansicht entfernt werden, können Sie den Befehl **Entladen** verwenden.

**Kommentar ändern**

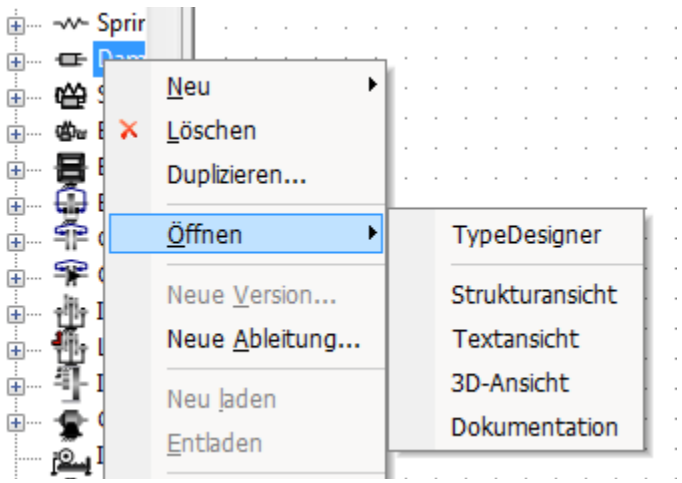
Mit diesem Befehl kann der Kommentar von Typen geändert werden. Die Bearbeitung der Zeichenkette erfolgt dabei direkt innerhalb der Baumansicht. Mit der Return-Taste schließen Sie die Bearbeitung ab. Die Escape-Taste bricht die Bearbeitung ohne die Übernahme des neuen Kommentars ab.

**Bearbeiten…**

Wählen Sie diesen Befehl um den ausgewählten Typ im TypeDesigner zu bearbeiten. Solange der TypeDesigner geöffnet ist, bleibt der betreffende Typ für die weitere Bearbeitung gesperrt. Typen der Modelica Standard Bibliothek können im TypeDesigner geöffnet, aber nicht gespeichert werden.

**Öffnen**

Wählen Sie diesen Befehl um die Modellstruktur des ausgewählten Typs im TypeDesigner, in der Strukturansicht, der Textansicht, der 3D-Ansicht oder der Dokumentations-Ansicht zu bearbeiten. Solange die Strukturansicht geöffnet ist, bleibt der betreffende Typ für die weitere Bearbeitung gesperrt. Typen der Modelica Standard Bibliothek können geöffnet, aber nicht gespeichert werden.
Es stehen nicht für jeden Typen alle Features zur Verfügung.
Beachten Sie den Hinweis in **Kapitel 7.6.5**.



**Neue Version…**

Verwenden Sie diesen Befehl, um eine neue Version des ausgewählten Typs zu erstellen. Die neuerstellte Version ersetzt den Ursprungstyp in der Bibliotheksansicht. Die Ursprungsversion steht jedoch weiter zur Verfügung. Modelle, die einen Typ verwenden, für den mehrere Versionen zur Verfügung stehen, verwenden standardmäßig die jeweils aktuelle Version. Es kann jedoch auch die Verwendung einer bestimmten Version erzwungen werden. Die konsequente Verwendung von Versionen kann die Reproduzierbarkeit von Berechnungsergebnissen erleichtern, da sich so Modelle entwickeln lassen, die von späteren Änderungen an den zugrundeliegenden Typen unberührt bleiben.

**Neue Ableitung…**

Mit diesem Befehl können Sie eine Ableitung des ausgewählten Typs erstellen. Ableitungen gestatten die Erweiterung bestehender Elementtypen um z.B. einen zusätzlichen Anschluss oder Parameter. Mit einer Ableitung können Sie jedoch das Verhalten des Elements nicht grundlegend ändern.

**Neu laden**

Verwenden Sie diesen Befehl, um eine Gruppe von Elementtypen, beginnend mit dem ausgewählten Typ, neu zu laden. Dies kann erforderlich sein, wenn Sie eine mo-Datei außerhalb von SimulationX geändert haben und diese Änderungen in SimulationX übernehmen wollen. Beachten Sie, dass vor der Ausführung dieses Befehls alle geöffneten Modelle geschlossen werden. Modelländerungen können dabei gespeichert werden.

**Entladen**

Um Speicherplatz zu sparen, oder den Umfang der Bibliotheksleiste zu reduzieren, können Sie einzelne oder Gruppen von Typen entladen. Anders als beim Löschen werden dabei keine Dateien oder Verzeichnisse gelöscht. Nach einem Neustart oder der Ausführung des Befehls **Neu laden** stehen die betreffenden Typen wieder zur Verfügung. Beachten Sie, dass vor der Ausführung dieses Befehls alle geöffneten Modelle geschlossen werden. Modelländerungen können dabei gespeichert werden.

**Drag & Drop**

Elementtypen können per *Drag & Drop* innerhalb der Bibliotheksleiste sowie in die Baumansicht des Modellexplorers kopiert oder verschoben werden. Typen der Modelica Standard Bibliothek können nicht verschoben werden. Geschützte Typen sind prinzipiell von *Drag & Drop* Operationen ausgenommen. Um einen solchen Type zu kopieren oder zu verschieben, müssen Sie zuvor im TypeDesigner den Passwortschutz aufheben.

**Achtung:**
Das **Verschieben** von Elementtypen kann dazu führen, dass Modelle, die diese Typen verwenden, nicht mehr ladbar sind. Daher sollten Sie von dieser Möglichkeit nur im Ausnahmefall Gebrauch machen.

Am unteren Rand der Bibliotheksleiste werden in einem kleinen separaten Fensterbereich ein Vorschaubild sowie einige nützliche Informationen zu dem in der Baumansicht ausgewählten Eintrag angezeigt. Über einen Hyperlink lässt sich im Windows-Explorer das Dateiverzeichnis öffnen, das die zu dem betreffenden Typ gehörige Datei enthält.

### 4.2.3 Struktur- , 3D-, Text- und Dokumentationsansicht

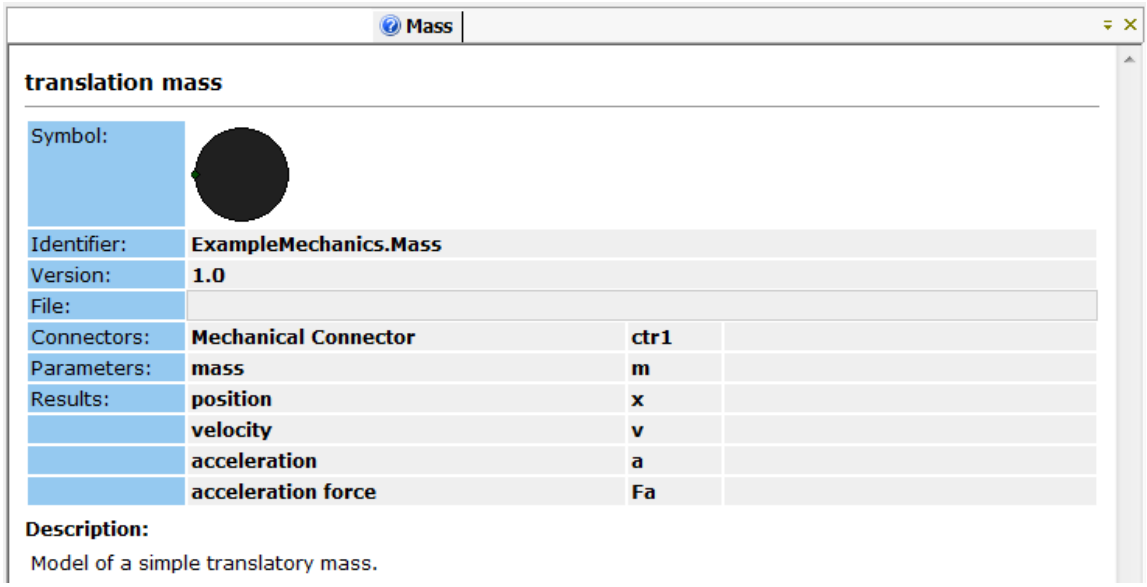
SimulationX unterstützt vier Arten von Dokumentenfenstern. Die **Strukturansicht** (3) dient der Visualisierung und Bearbeitung der logischen Struktur eines Modells. Mit der **3D-Ansicht** (4) können Elemente der Bibliothek MKS-Mechanik räumlich dargestellt und deren Lage im Raum interaktiv bearbeitet werden.
Für ein Modell können gleichzeitig mehrere Ansichten mit unterschiedlichen Einstellungen geöffnet werden. Mehr zur 3D-Ansicht erfahren Sie in **Kapitel 5.4.4**.

Die **Textansicht** dient der Darstellung und Eingabe des Modelica-Codes.

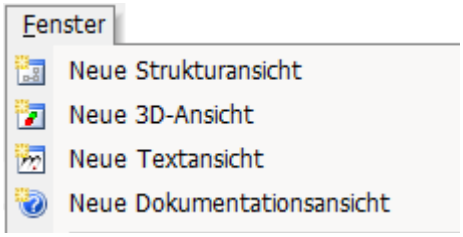
ExampleMechanics.Mass*

```
1   model Mass "translation mass"
2       MechConn ctr1 annotation(...);
5       Modelica.SIunits.Position x(
6           quantity="Mechanics.Translation.Displace",
7           displayUnit="m")"position";
8       Modelica.SIunits.Velocity v "velocity";
9       Modelica.SIunits.Acceleration a "acceleration";
10      Modelica.SIunits.Force Fa "acceleration force";
11      parameter Modelica.SIunits.Mass m=1 "mass";
12      equation
13          // enter your equations here
14          //
15          // interface equations to the connector
16          ctr1.x=x;
17          ctr1.F=Fa;
18          // derivates
19          v=der(x);
20          a=der(v);
21          // acceleration force
22          a*m=Fa;
23      public
24          annotation(...);
164 end Mass;
165
```

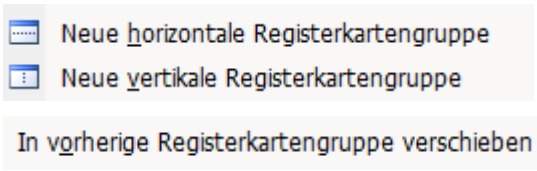
Mit der **Dokumentationsansicht** erhält man eine Darstellung, die der Ansicht der Online-Hilfe entspricht. Sie dient der Dokumentation des Modells.




Zusätzliche Ansichten erzeugen Sie über das Menü ***Fenster***.




Ein Klick mit der rechten Maustaste auf ein Register öffnet ein Popup-Menü mit folgenden Befehlen:




Erzeugen einer vertikalen oder horizontalen Anordnung

Wechsel in die vorherige Ansicht

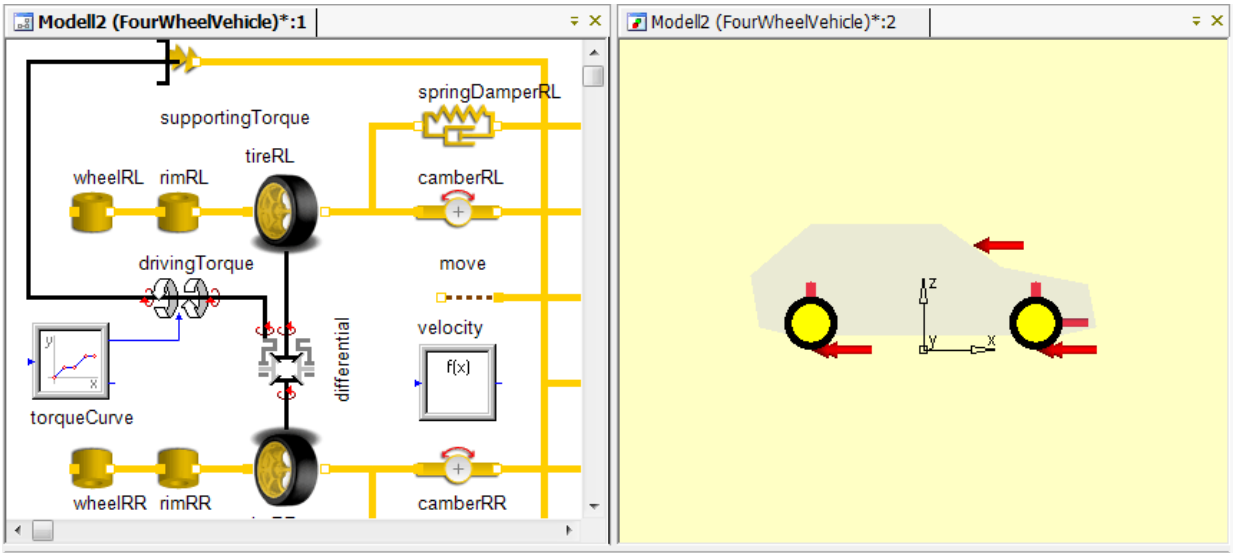



**Bild 4.3:** Vertikal angeordnete Strukturansichten

Für die Aufteilung in **Bild 4.4** wurde eine neue vertikale Registergruppe eingefügt. Alle vorgenommenen Einstellungen werden mit dem Modell gespeichert.

Im Eigenschaftendialog des Modells können im Register ***3D-Ansicht*** Einstellungen zur Darstellung (Farben, Raster u.a.) festgelegt werden. Diese können mit der Schaltfläche ***Als Standard speichern*** generell übernommen bzw. mit der Schaltfläche ***Rücksetzen auf Standardwerte*** zurückgesetzt werden.

## 4.2.4 Modellexplorer

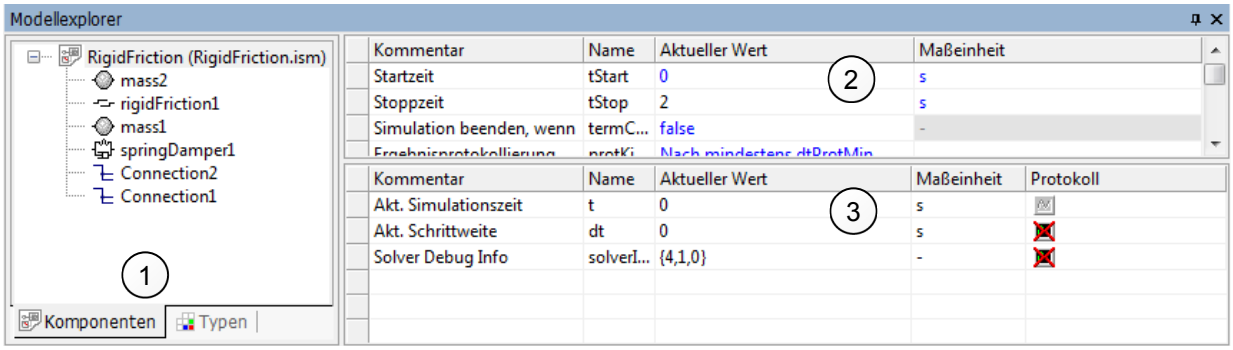



**Bild 4.4:** Modellexplorer: (1) Baumansicht, (2) Parametertabelle, (3) Ergebnistabelle

Die Baumansicht (1) des Modellexplorers (**Bild 4.5**) stellt alle im Modell verwendeten Komponenten und Typen dar. Für ein ausgewähltes Element oder eine Verbindung werden die Parameter und Ergebnisgrößen in Tabellenform aufgelistet, siehe (2) und (3). Sind keine Modellelemente markiert, werden die allgemeinen Parameter und Ergebnisvariablen des Modells angezeigt.

### 4.2.4.1 Bearbeiten von Komponenten-Eigenschaften

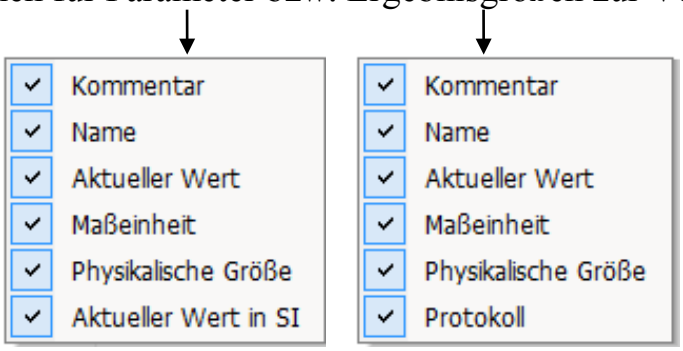
Markieren Sie in der Strukturansicht oder in der Baumansicht des Modellexplorers die Komponente (Element oder Verbindung), deren Eigenschaften Sie bearbeiten wollen. Eine Komponente, die Sie in einer Strukturansicht markieren, wird in den übrigen Strukturansichten sowie in der Baumansicht des Modellexplorers ebenfalls selektiert.
Für die jeweils gewählte Komponente werden auf der rechten Seite des Explorer-Fensters die zugehörigen Parameter und Ergebnisgrößen in Form zweier Tabellen angezeigt. Um einen Eintrag zu bearbeiten, klicken Sie in das betreffende Tabellenfeld. Sie können nun den Inhalt editieren oder einen Eintrag aus der zugehörigen Liste wählen.
Solange ein Eintrag bearbeitet wird, erfolgt keine Aktualisierung des zugehörigen Parameters. Erst mit Beendigung der Eingabe wird - nach entsprechender Gültigkeitsprüfung - der neue Wert an den Parameter übergeben.
Um die Eingabe zu beenden, gibt es folgende Möglichkeiten:

- durch Wechsel zu einer anderen Zeile innerhalb der Tabelle mit den Cursortasten ↑ und ↓ bzw. Mausklick auf das neue Feld

- Betätigen der *Return-Taste*
- Wechsel in ein anderes Fenster z. B. durch Klick auf die Strukturansicht

Änderungen können rückgängig gemacht werden (Menü ***Bearbeiten***).

Durch Anklicken der Kopfzeile der Parameter- oder der Ergebnisgrößentabelle mit der rechten Maustaste können Sie die darzustellenden Spalten durch Setzen des entsprechenden Hakens auswählen.
Folgende Spalten stehen für Parameter bzw. Ergebnisgrößen zur Verfügung:




Zur Sortierung der Tabelle klicken Sie mit der linken Maustaste auf den entsprechenden Spaltenkopf. Die Richtung der Sortierung ( - aufwärts; - abwärts) können Sie durch einen erneuten Klick ändern. Um zur standardmäßigen Reihenfolge zurückzukehren, klicken Sie auf den Kopf der Auswahlspalte (Spalte 1).

Um das Protokollattribut einer Ergebnisgröße ein- bzw. auszuschalten, klicken Sie auf das entsprechende Symbol in der Protokoll-Spalte.

Ergebnisgrößen, deren Protokollattribut eingeschaltet worden ist, können mittels Drag & Drop in ein bereits geöffnetes Ergebnisfenster oder in die Strukturansicht gezogen werden. Dazu wird das Protokollsymbol mit der linken Maustaste angeklickt und bei gedrückter Maustaste an die entsprechende Stelle bewegt. Beim Loslassen der Taste wird der Ergebnisverlauf je nach Ziel der Operation in einem bestehenden oder neuen Ergebnisfenster angezeigt.

| | Kommentar | Name | Aktueller Wert | Maßeinheit | Physik... | Protokoll |
|---|---|---|---|---|---|---|
| | Trägheitskraft | Fa | 0 | N | Kraft | |
| | Weg | x | 0 | m | Weg | |
| | Geschwindigkeit | v | 0 | m/s | Gesch... | |
| | Beschleunigung | a | 0 | $m/s^2$ | Beschl... | |
| | Zugeführte Leistung | Pk | 0 | kW | Leistu... | |

Der aktuelle Wert einer Ergebnisgröße kann nicht editiert werden. Die Maßeinheit, in der die Größe angezeigt werden soll, lässt sich jedoch einstellen.

#### 4.2.4.2 Hinzufügen von Parametern und Ergebnisgrößen

Um zum aktuellen Modell oder einem Element-Typ einen Parameter hinzuzufügen, führen Sie die folgenden Arbeitsschritte aus:

1. Wählen Sie in der Baumansicht des Modellexplorers unter **Typen** das Modell oder einen im Modell enthaltenen Typ aus.
2. Öffnen Sie in der entsprechenden Tabelle des Modellexplorers ein Kontextmenü (rechte Maustaste).
3. Wählen Sie den Befehl ***Neu***. Die angelegte Größe wird in der Tabelle angezeigt.
4. Name und Bezeichnung der neuen Größe können in der Tabelle direkt bearbeitet werden.

#### 4.2.4.3 Löschen von Parametern und Ergebnisgrößen

Um im aktuellen Modell oder einem Element-Typ einen Parameter zu löschen, führen Sie die folgenden Arbeitsschritte aus:

1. Wählen Sie in der Baumansicht des Modellexplorers das Modell oder einen im Modell gespeicherten Typen aus.
2. Markieren Sie in der entsprechenden Tabelle des Modellexplorers den Eintrag für die zu löschende Größe.
3. Führen Sie den Befehl ***Löschen*** aus. Die Größe wird aus der Tabelle und damit aus dem Modell entfernt.

Es können nur solche Parameter oder Ergebnisgrößen gelöscht werden, die vorher selbst hinzugefügt worden sind.

---

**Tipp:**
Für das Hinzufügen und Löschen von Parametern oder Ergebnisgrößen können Sie auch den TypeDesigner nutzen (s. **Kapitel 9 Erweiterte Modellierung**).

---

### 4.2.5 Ausgabebereich

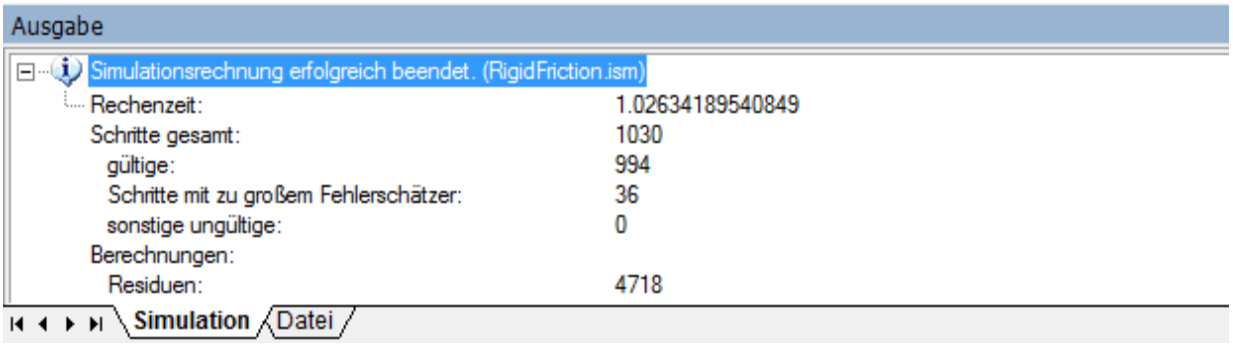
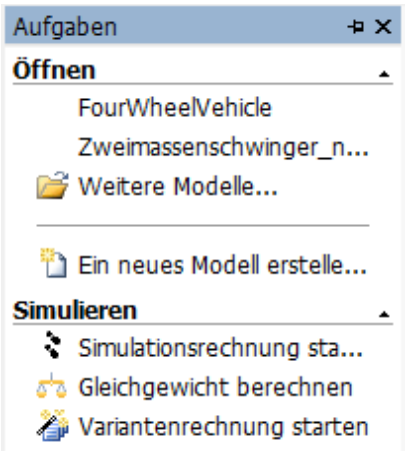



**Bild 4.5:** Ausgabebereich

Im **Ausgabebereich (6)** in Bild 4.1 werden Meldungen, Tracingausgaben, Warnungen und Fehlermitteilungen protokolliert. Diese Nachrichten sind verschiedenen Kategorien zugeordnet (z. B. Simulation, Datei). Der Inhalt des Ausgabebereichs kann gespeichert, im Textformat exportiert und ausgedruckt werden. Weitere Informationen zu Tracingausgaben sind in **6.1.1.4** enthalten.

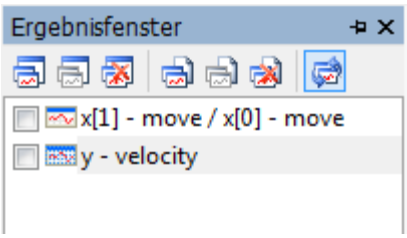
### 4.2.6 Aufgabenbereich

Im **Aufgabenbereich** (7) in Bild 4.1 werden die besonders häufig benutzten Befehle (Öffnen von Modellen, Start der Simulation) aufgelistet und können von dort direkt aufgerufen werden.




### 4.2.7 Ergebnisfenstermanager

Der **Ergebnisfenstermanager** (8) in Bild 4.1 erlaubt die zentrale Verwaltung der Ergebnisfenster aller geöffneten Modelle. Ergebnisfenster können mit einem Klick auf das jeweilige Symbol aus- bzw. eingeblendet werden oder über die Betätigung der entsprechenden Schaltfläche in der Symbolleiste geschlossen werden.




Die Schaltflächen haben folgende Bedeutung:

- Laden eines zuvor gespeicherten Ergebnisverlaufes
- alle Ergebnisfenster sichtbar
- alle Ergebnisfenster unsichtbar
- alle Ergebnisfenster löschen
- Ergebnisfenster des aktuellen Modells sichtbar
- Ergebnisfenster des aktuellen Modells unsichtbar
- Ergebnisfenster des aktuellen Modells löschen
- Ergebnisfenster synchronisieren

In der Liste der Ergebnisfenster können Sie durch Setzen/Löschen des Hakens einzelne Ergebnisfenster sichtbar/unsichtbar setzen.

SIMULATION X®

Powered by ITI

Modeling

# Modeling

# 5 Modellierung

## 5.1 Modellierungsmöglichkeiten in SimulationX

In SimulationX gibt es verschiedene Wege Modelle zu erstellen:

- Physikalisch-objektorientierte Modelle
- Signalorientierte Modelle
- Gleichungs- und Algorithmen-basierte Modelle

Dadurch kann der Anwender entscheiden, welches für Ihn der beste Ansatz für die Komponenten- und Systemmodellierung ist. Auch eine Kombination der verschiedenen Ansätze in einem Modell ist möglich, so dass für jede Komponente der bestmögliche Weg zur Modellierung genutzt werden kann.

Die Nutzung und die Anwendungsgebiete der einzelnen Modellierungsansätze sind im Folgenden beschrieben.

### 5.1.1 Physikalisch-objektorientierte Modellierung

Die physikalisch-objektorientierte Modellierung ist die natürlichste Weise der physikalischen Verhaltensbeschreibung in Simulationsmodellen. Sie ist auch als Netzwerkmodellierung bekannt. SimulationX nutzt diesen Modellansatz in allen seinen physikalischen Domänen und Bibliotheken.

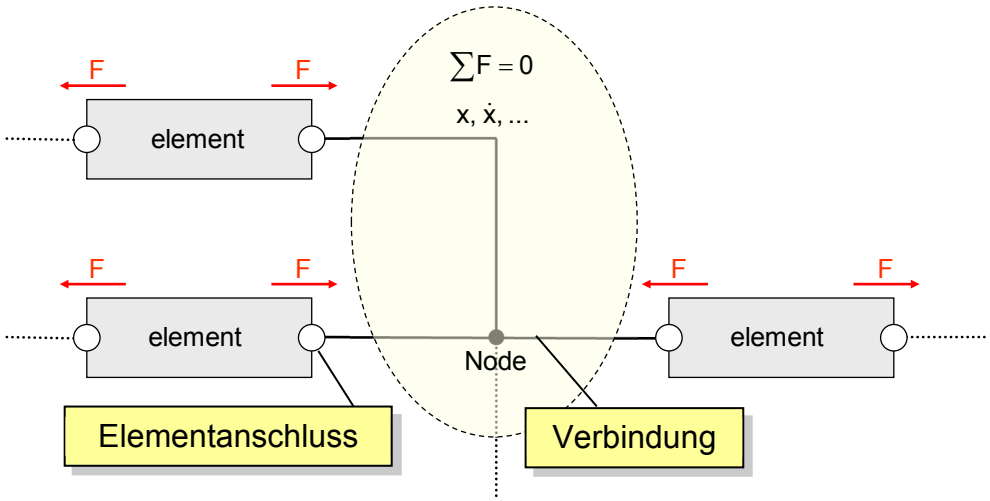



**Bild 5.1:** Physikalische Netzwerkmodellierung

Das Konzept der Netzwerkmodellierung ist in **Bild 5.1** dargestellt. Physikalische Netzwerkmodelle werden durch Elemente gebildet, die untereinander durch Verbindungen , auch Knoten genannt, zusammengefügt werden. In SimulationX sind Elemente Objekte, die in den Bibliotheken zu finden sind und bei Ihrer Zusammenschaltung Verbindungen erzeugen.

Physikalische Zusammenhänge in Netzwerkmodellen werden durch Ausdrücke von Potential- und Flussgrößen formuliert.

Eine Verbindung führt ein Satz von Potenzialgrößen, der für alle verbundenen Elementanschlüsse identisch ist. Solche mögliche Größen sind z.B. Wege, Geschwindigkeiten und Beschleunigungen in der Mechanik, Druck in den Fluid-Bibliotheken, Spannungen in der Elektronik oder Temperaturen in den thermischen Modellen.

Die Flussgrößen sind Größen, für die bestimmte Erhaltungsgleichungen erfüllt werden müssen. Zum Beispiel müssen Kräfte oder Drehmomente an den Verbindungen in der Mechanik sich zu Null aufsummieren. Das gleiche gilt für Ströme in der Elektronik und für Flussgrößen in anderen physikalischen Domänen.

Die Potenzialgrößen, wie auch die Erhaltungsgleichungen für die Flussgrößen die zu einer physikalischen Domäne gehören, werden in den Verbindungen definiert.

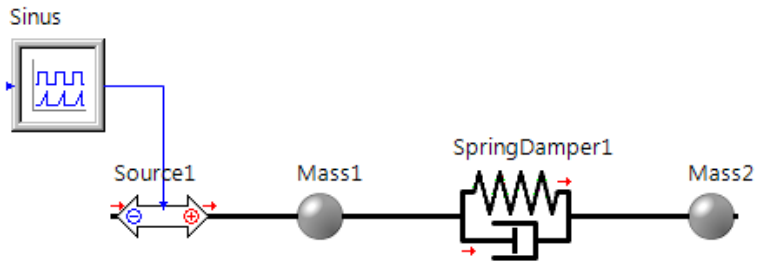



**Bild 5.2:** Physikalisches Netzwerkmodell der Linearen Mechanik

Elemente, wie die Objekte die man in den SimulationX-Bibliotheken findet oder die vom Anwender entwickelt wurden, definieren eine Beziehung zwischen den Potenzialgrößen an ihren Anschlüssen und den internen Flussgrößen. Eine Feder in der Mechanik beschreibt die innere Kraft `F` über die Wegdifferenz `dx` und die Steifigkeit `k` mittels der Gleichung `F=k*dx`.

### 5.1.2 Signalorientierte Modellierung

Die Modellierung von Signalstrukturen folgt einem von der physikalischen Modellierung abweichendem Konzept. Signalstrukturen, wie z.B. Regler, sind informationsverarbeitende Strukturen. Das bedeutet, dass die Elemente solcher Strukturen Ausgangsdaten aus Eingangsdaten erzeugen und diese weitergeben. Innerhalb der Modelle existiert ein klar definierter Informationsfluss (Kausalität).

Signalorientierte Modelle sind der passende Weg um Regelungssysteme abzubilden und können auch für Hilfsstrukturen zur Berechnung von Abhängigkeiten in den physikalischen Bibliotheken genutzt werden. In **Bild 5.3** ist eine signalorientierte Struktur dargestellt.

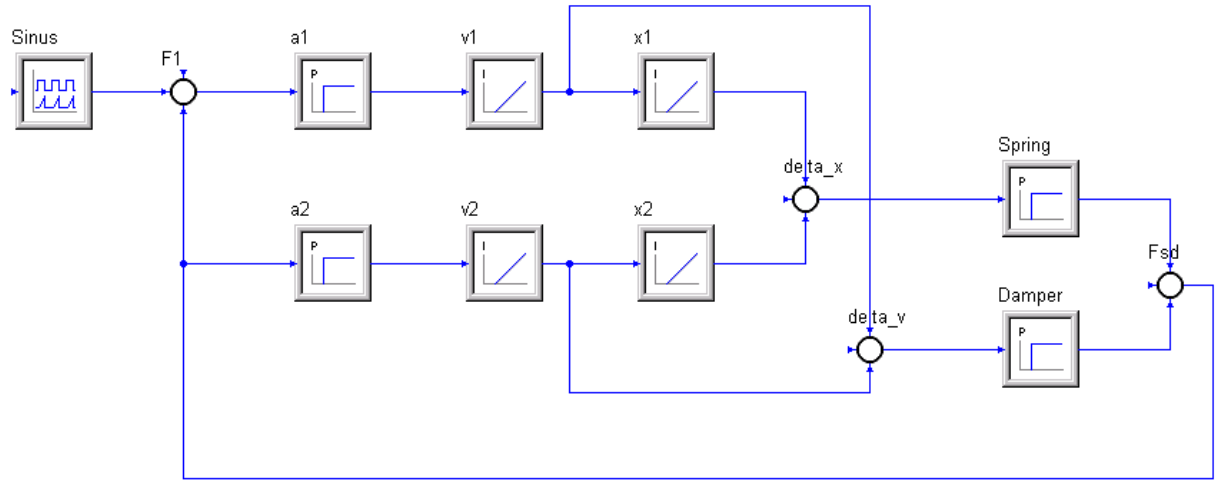



**Bild 5.3:** Signalorientiertes Modell

Für die Verbindung von physikalischen Modellen mit signalorientierten Modellen gibt es Sensoren und Aktuatoren für die verschiedenen physikalischen Domänen. Sensoren wandeln physikalische Größen in Signale um und Aktuatoren nutzen Signale und wandeln diese in entsprechende physikalische Aktionen (wie z.B. Kräfte oder Wegverschiebungen) um.

Die signalorientierte Modellierung kann auch für die Modellierung physikalischer Strukturen eingesetzt werden. Dies ist aber umständlich und führt zu schwer verständlichen Modellen. So beschreiben die Modelle in **Bild 5.2** und **Bild 5.3** den gleichen Sachverhalt. Das verdeutlicht den Vorteil der physikalischen Modellierung.

### 5.1.3 Modellierung mit Gleichungen und Algorithmen

Gleichungen und Algorithmen in SimulationX nutzen die Fähigkeiten der Modellierungssprache Modelica, die ein wesentlicher Bestandteil von SimulationX ist (siehe **Kapitel 8** und **10**).

Gleichungen ermöglichen die direkte textorientierte Eingabe physikalischer Zusammenhänge in algebraischen und Differenzialgleichungen, was für die Erweiterung von SimulationX und die Entwicklung anwenderspezifischer Elemente mit Funktionen außerhalb der SimulationX-Standardbibliotheken nützlich ist.

Algorithmen stellen Methoden für das Implementieren von programmgesteuerter Funktionalität, wie z.B. in Reglern, bereit.

Die Arbeit mit Algorithmen und Gleichungen ist nicht Bestandteil dieses Kapitels. Eine weitergehende Erläuterung ist in **Kapitel 8** enthalten.

# 5.2 Modellerstellung

## 5.2.1 Die SimulationX Programmoberfläche

Die Programmoberfläche von SimulationX ist in verschiedene Fenster und Bereiche unterteilt und enthält alle Werkzeuge zur Arbeit mit dem Simulationstool. Eine detaillierte Beschreibung finden Sie in **Kapitel 3**.

SimulationX-Modelle erzeugt man durch Ziehen von Elementen aus der Bibliotheksleiste in die ***Strukturansicht,*** die sich normalerweise in der Mitte des SimulationX-Fensters befindet. Die Bibliotheksleiste, die sich typischerweise im linken Teil des SimulationX-Fensters befindet, stellt die Sammlung von Element-Typen, die in SimulationX Modellen genutzt werden können, bereit.

Das Speichern von Modellen kann als

- SimulationX-Modell (*.ism),
- SimulationX-Projekt (*.isx) oder
- Modelica-Datei erfolgen.

Beim Speichern von Modellen können Sie zusätzlich folgende Optionen auswählen:

| | |
|---|---|
| SimulationX Modell (*.ism) | ☐ Verschlüsselt speichern<br>☐ Simulationsergebnisse komprimieren<br>☐ Modell ohne Simulationsergebnisse speichern<br>☐ Modell ohne Kommentare speichern<br>☐ Modell ohne CAD-Daten speichern |
| SimulationX Projekt (*.isx) | ☐ Modell ohne Kommentare speichern<br>☐ Modell ohne CAD-Daten speichern |
| Modelica Datei (*.mo) | |

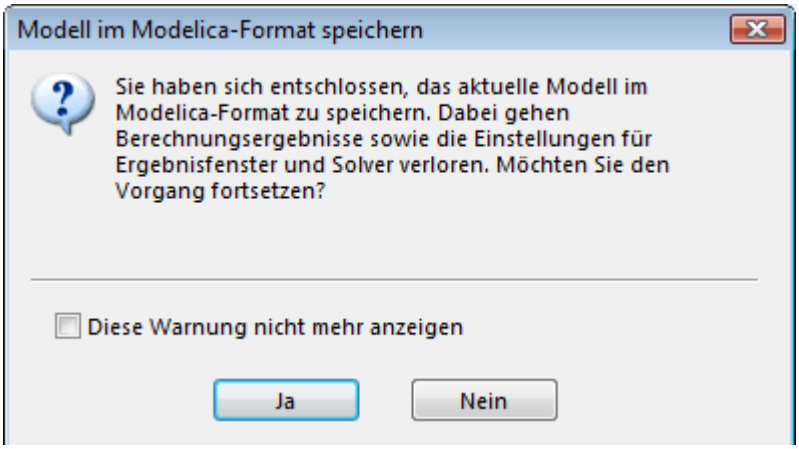
Wenn Sie Ihr Modell vor unberechtigtem Zugriff schützen wollen, speichern Sie dieses als verschlüsseltes SimulationX-Modell ab.

Das Dateiformat SimulationX-Projekt gestattet das beschleunigte Laden und Speichern von Modellen mit umfangreichen binären Daten, wie aufgezeichneten Simulationsergebnissen, Animationen und CAD-Daten sowie in die Strukturansicht eingefügte Grafiken. Ein solches Projekt ist wie ein Zip-Archiv aufgebaut und erlaubt den direkten Zugriff auf das Modell im Textformat und die eingebetteten Daten. Damit können aufgezeichnete Simulationsergebnisse aus einer Datei mittels ZIP-Programm extrahiert und in anderen Programmen ausgewertet bzw. weiterverarbeitet werden.

Die verschlüsselte Speicherung von Modellen wird derzeit nicht unterstützt!

Wird das Speichern von importierten CAD-Daten ausgeschlossen, so werden die Dreiecksnetze, welche die Form des Körpers beschreiben durch einfache Quader ersetzt. Die Massen- und Trägheitsdaten bleiben jedoch erhalten.

Beim Speichern als Modelica-Datei erhalten Sie folgenden Hinweis:




Alternativ kann ein SimulationX-Modell auch exportiert werden. Dies geschieht über das Menü ***Datei/Export***. Es stehen dieselben Auswahlen, wie beim Speichern zur Verfügung. Beim exportieren werden alle externen Referenzen aufgelöst, in das Modell integriert und gespeichert (wie Menü ***Berabeiten/Auflösen von externen Referenzen***). Die vormals externen Referenzen findet man dann im ***Modellexplorer*** unter ***Typen***.

## 5.2.2 Modellelemente und Bibliotheken

Elemente für SimulationX können von 3 verschiedenen Quellen kommen:

- SimulationX-Bibliotheken
- Modelica Standardbibliothek
- Bibliotheken von anwenderdefinierten Element-Typen.

All diese Elemente können Teil ein und desselben Modells sein.

### 5.2.2.1 SimulationX Bibliotheken

SimulationX stellt eine Sammlung von Bibliotheken (siehe **Bild 5.4**) für verschiedene physikalische Domänen und Applikationsfelder bereit. Die

Verfügbarkeit der Bibliotheken hängt von der Lizenz ab, die für die Installation von SimulationX erworben wurde.

Diese SimulationX-Bibliotheken werden von ITI entwickelt und gepflegt und basieren auf dem Know-Hows einer Vielzahl von Ingenieur-Projekten und Applikationen. Sie sind umfangreich validiert und getestet und die Handhabung der Elemente dieser Bibliotheken gewährt ein Maximum an Komfort und Einfachheit der Nutzung. Für diese Bibliotheken kann der Anwender von ITI volle inhaltliche und modellierungstechnische Unterstützung erhalten.

Die SimulationX-Bibliotheken basieren alle auf den Modellierungsansätzen nach **Abschnitt 5.1.**

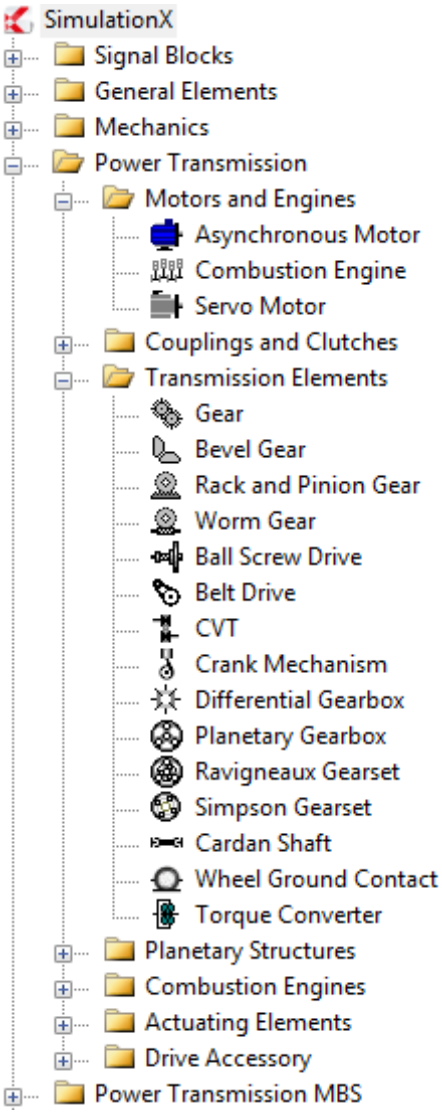



**Bild 5.4:** Teilansicht der SimulationX Bibliotheken

Die Bibliotheken können nicht verändert werden. Sie können aber als Ausgangspunkt für die kundenspezifische Modellobjekt-Entwicklung durch Zusammenfassen in Compounds und die Erweiterung der Funktionalität vorhandener Bibliothekselemente (siehe **Kapitel 8**) genutzt werden.

### 5.2.2.2 Modelica Standard-Bibliothek

SimulationX nutzt Modelica als eine Modellentwicklungs- und Programmierschnittstelle für anwenderspezifische Typen. Modelica ist ein produktunabhängiger Sprachstandard und es existieren in Modelica geschriebene, öffentlich verfügbare Modellbibliotheken. Diese können auch in SimulationX genutzt werden.

Die Nutzung der Modelica-Bibliotheken in SimulationX ist in **Kapitel 10** beschreiben.
Die Modelica-Bibliotheken werden nicht von ITI entwickelt und es gibt keinen technischen Support.

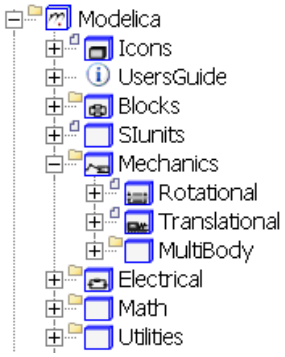



**Bild 5.5:** Teilansicht der Modelica Bibliotheken

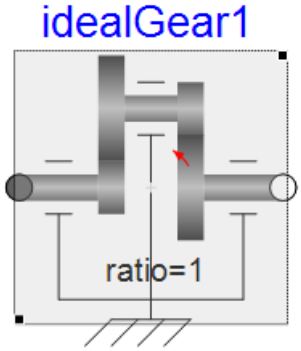
Im Vergleich zu den SimulationX-Bibliotheken gibt es in Modelica einige Einschränkungen:

in Modelica gibt es keine Connection-Objekte, was die Modelobjekt-Entwicklung komplexer macht. Die Verbindung mit SimulationX-Bibliotheken erfordert spezielle Interface-Elemente.

Modelica erlaubt keine variablen Werte oder Ausdrücke bei der Zuweisung zu Parametern, was ein Hauptfaktor für die Modellierungsflexibilität in SimulationX ist.

- **Größenänderung und Rotation von Modelica-Objekten:**

Um die Größe eines Elements zu verändern, positionieren Sie den Mauszeiger auf einen speziell dafür markierten Eckpunkt des Markierungsrahmens und ziehen bei gedrückter linker Maustaste das Elementsymbol auf die gewünschte Größe.

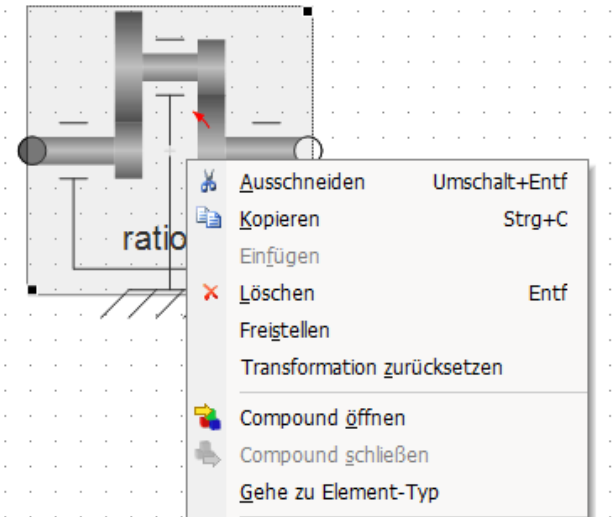


Wenn dabei gleichzeitig die *Shift-Taste* gedrückt gehalten wird, wird die ursprüngliche Proportion des Elements beibehalten.

Die Größe eines Elements lässt sich nur anpassen, wenn die folgenden Voraussetzungen erfüllt sind:

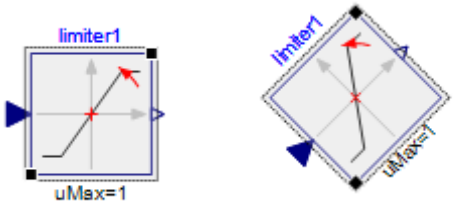
- Für das Symbol des Elementtyps werden Vektorgrafiken statt Bitmaps verwendet
- Anstelle von proprietären SimulationX-Anschlüssen werden ausschließlich Modelica-kompatible Connectoren verwendet.

Die genannten Voraussetzungen werden durch die Elemente der Modelica-Standard-Bibliothek sowie durch alle anderen Modelica-konformen Typen erfüllt.




Zum Wiederherstellen der Standardgröße wählen Sie im Kontextmenü des Elements den Befehl ***Transformation zurücksetzen***.

Diese Elemente lassen sich auch drehen. Dazu wird das Element angeklickt, mit dem Cursor an dem roten Pfeil angefasst und gedreht.




### 5.2.2.3 Nutzerdefinierte Element-Typen

SimulationX stellt dem Nutzer verschiedenen Möglichkeiten für die Entwicklung von kundenspezifischen Bibliotheken und Modellelementen zur Verfügung. Diese werden in der Bibliotheksleiste im Verzeichnis ***ExternalTypes*** oder innerhalb des Simulationsmodells als lokale Typen abgelegt.

Nuntzerdefinierte Element-Typen sind:

- Compounds von vorhandenen Typen
- Erweiterungen von vorhandenen Typen
- Neuentwicklungen auf der Basis von Gleichungen und Algorithmen

oder eine Kombination dieser.

Für die Entwicklung der kundenspezifischen Element-Typen können alle SimulationX und Modelica Fähigkeiten ausgenutzt und gemeinsam verwendet werden.

Für Details zur Entwicklung anwenderspezifischer Element-Typen siehe **Kapitel 8**.

## 5.2.3 Modellerstellung und Modifikation

Dieser Abschnitt beschreibt die Erstellung von Simulationsmodellen mit SimulationX. Er gibt die notwendigen Informationen für einen Start. Mehr spezialisierte Themen, wie die Handhabung von Kennlinien, die Modellierung mit der MKS-Mechanik und das Einfügen von zusätzlichen Elementen (Momentan-anzeigen, Textboxen, Steuerelemente) in Modelle werden separat in **Abschnitt** Fehler! Verweisquelle konnte nicht gefunden werden. und **5.4.5** beschrieben.

### 5.2.3.1 Aufbau einer Modellstruktur

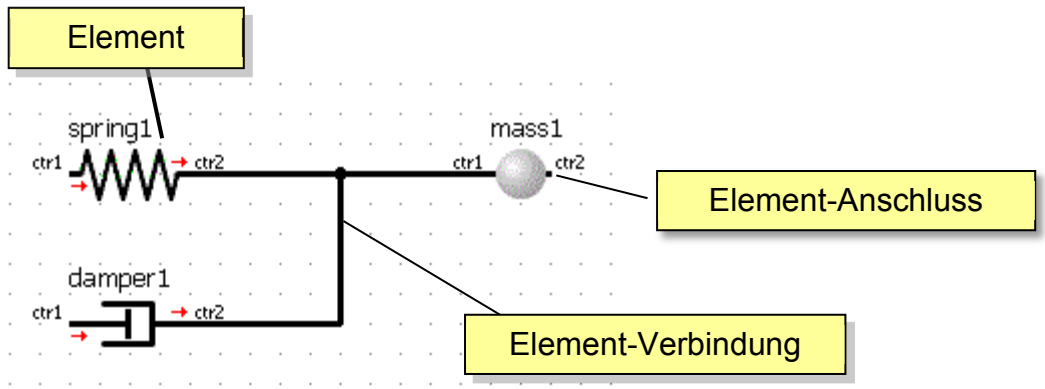



**Bild 5.6:** Komponenten

**Komponenten** eines Simulationsmodells (**Bild 5.6**) sind Elemente und Verbindungen. **Elemente** verfügen über Anschlusspunkte, die über **Verbindungen** miteinander verknüpft werden können (siehe unten). Verbindungen können beliebig verzweigt sein, d. h. Sie können mehr als zwei Anschlusspunkte miteinander verbinden.

Es gibt verschiedene Typen von Anschlüssen (z. B. Translatorische und Rotatorische Mechanik, Hydraulik, Elektrotechnik, Signalein- und -ausgang).
Nur Anschlüsse gleichen Typs lassen sich miteinander verbinden. Jeder Anschluss

besitzt - bezogen auf das jeweilige Element - einen eindeutigen Namen. Diese können über das Menü ***Ansicht/Anschlussbeschriftung*** oder die Schaltfläche eingeblendet werden.

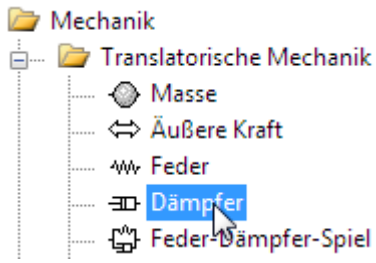
---

**Hinweis für ITI-SIM 3.x Anwender:**
Element-Typen werden nicht mehr nach Elementen und Knoten unterschieden. Die besondere Eigenschaft von Knoten, Träger der Zustandsgrößen und Erhaltungsgleichungen zu sein, entfällt. Zustandsgrößen werden nunmehr vorrangig in den Verbindungen realisiert.

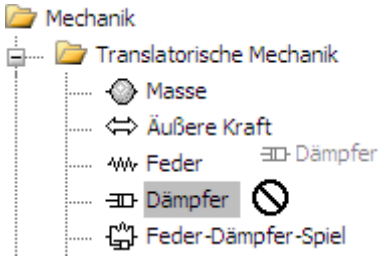
---

Um ein **neues Element** mittels *Drag & Drop* in der Strukturansicht zu platzieren, gehen Sie wie folgt vor:

1. Suchen Sie in der Baumansicht der Bibliotheksleiste den entsprechenden Element-Typ.

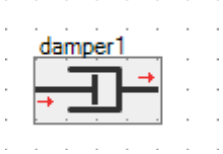



2. Klicken Sie mit der linken Maustaste auf den betreffenden Eintrag im Baum und halten Sie die Maustaste gedrückt.




3. Bewegen Sie nun den Mauszeiger mit gedrückter linker Taste zu der Position in der Strukturansicht, an der das neue Element eingefügt werden soll.

4. Beim Loslassen der Maustaste wird das Element an der gewünschten Stelle im Modell platziert.



Sie können sich das Positionieren von Elementen vereinfachen, indem Sie die folgenden Optionen und Befehle im Menü ***Elemente*** nutzen:

am Raster ausrichten

nach links drehen

nach rechts drehen

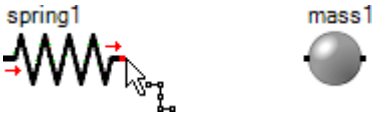
horizontal kippen

vertikal kippen

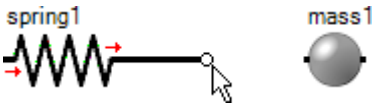
Um **mehrere Elemente** eines Typs in der Strukturansicht zu platzieren, wiederholen Sie Punkt 1-4 mehrfach oder nutzen die *Drag & Drop* Funktion (siehe **6.2.3.1**).

Um eine **Verbindung** zwischen Anschlusspunkten zweier Elemente zu ziehen, gehen Sie wie folgt vor:

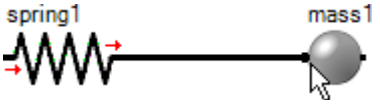
1. Bewegen Sie den Mauszeiger über den Anschluss, den Sie verbinden wollen. Der Mauszeiger verändert sich, der Anschluss färbt sich rot - Sie haben einen zulässigen Anschlusspunkt getroffen.




2. Bei gedrückter linker Maustaste ziehen Sie eine Verbindung zum Anschluss eines anderen Elements.



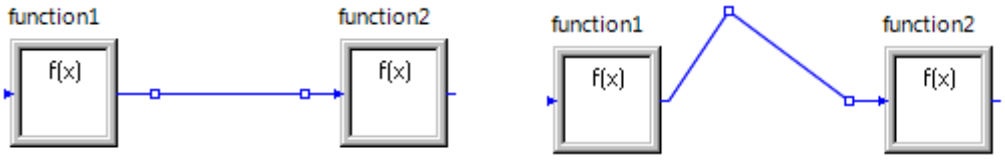

3. Maustaste loslassen - die Verbindung ist erstellt.



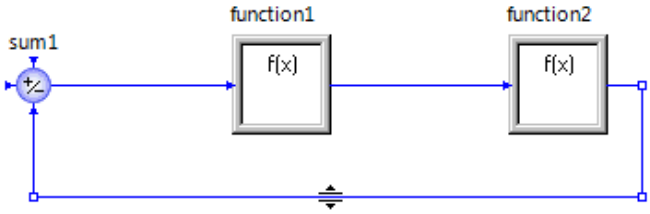

Sie können den Vorgang jederzeit abbrechen, indem Sie die Maustaste an einer freien Stelle der Strukturansicht loslassen oder die *Esc*-Taste betätigen.

**Hinweis für ITI-SIM 3.x Anwender:**
Für das Ziehen einer Verbindung ist es nicht mehr erforderlich, einen speziellen Bearbeitungsmodus einzustellen.

Der Weg einer Verbindung wird in SimulationX automatisch (Autorouting) festgelegt. Diesen zu verändern, ist leicht möglich. Dazu klicken sie mit der Maus auf die Verbindung. Anschließend werden die vorhandenen Stützstellen sichtbar.



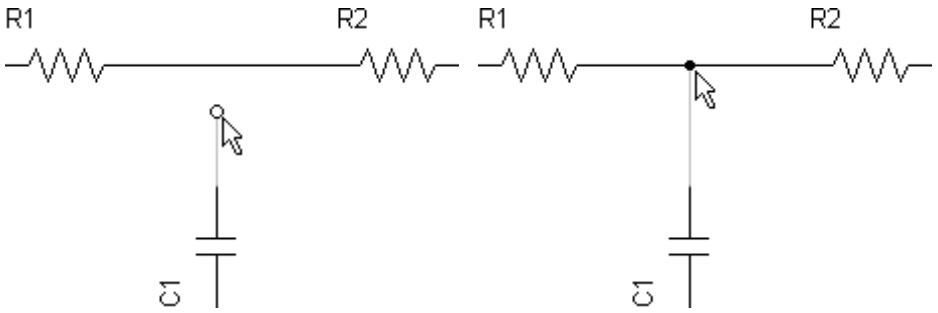

Diese können mit der Maus beliebig platziert werden, wodurch sich die Richtung der Verbindung ändert. Über das Kontextmenü der Verbindung können Punkte hinzugefügt/gelöscht werden. Auch kann das Autorouting abgeschaltet werden.




Beim Anklicken einer Verbindung erhält man durch den Kursor Informationen über die mögliche Richtung einer Verschiebung der Verbindung.

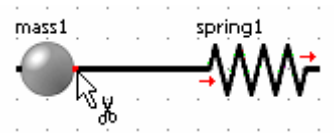
Es werden zwei grundlegende Arten von Verbindungen unterstützt:

- In Modelica sind Verbindungen nur zwischen zwei Connectoren möglich. Modelica-Verbindungen besitzen keinen eindeutigen Namen und verfügen weder über Parameter noch Ergebnisgrößen. Einer Verbindung kann jedoch mit einem optionalen Kommentar versehen werden.
- SimulationX-Verbindungen hingegen können beliebig verzweigt werden und können so mehr als zwei Element-Anschlüsse miteinander verbinden. Verbindungslinien können in beiden Richtungen auch zwischen einem freiem Anschlusspunkt und einer bereits bestehenden Verbindung gezogen werden. Jede SimulationX-Verbindung besitzt neben einem eindeutigen Namen in den meisten Fällen auch Parameter und Ergebnisgrößen.



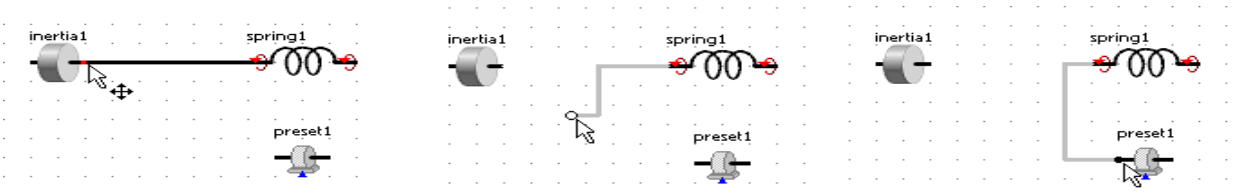
Verbindungen können **abgetrennt**, **umgeklemmt** oder **gelöscht** werden.

Um eine **Verbindung** an einem Anschluss **abzutrennen**, halten Sie die *Shift*-Taste gedrückt und gehen mit der Maus zu dem Anschluss, an dem die Verbindung getrennt werden soll.




Mit einem Klick auf die linke Maustaste wird die Verbindung gelöst. Über das Kontextmenü eines Elements können Sie mit dem Befehl ***Freistellen*** alle an diesem Element angeschlossenen Verbindungen trennen.

Um eine **Verbindung umzuklemmen**, gehen Sie mit der Maus zu einem bereits verbundenen Anschluss.
Bei gedrückter Maustaste können Sie die Verbindung aufnehmen und mit einem anderen Anschluss verbinden.



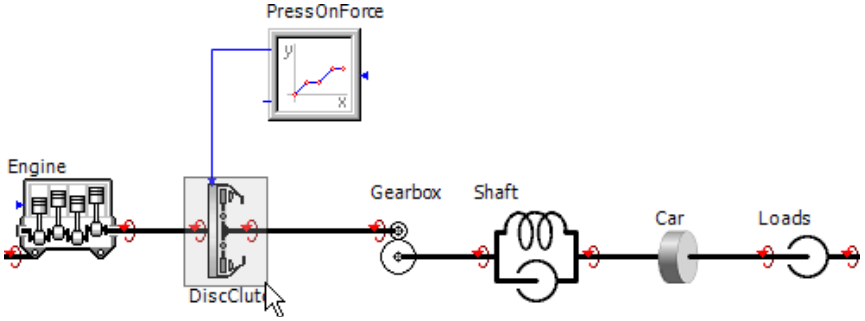

a) b) c)

Verbindungen können **dynamisch visualisiert** werden. Die Einstellungen erfolgen im Eigenschaftsdialog der Verbindungen (siehe **Kapitel 6.2**). Mit farbiger Hervorhebung können Sie die Unter- bzw. Überschreitung zulässiger Werte überwachen oder den aktuellen Zustand bestimmter Größen, wie z.B. Druck & Temperatur anschaulich darstellen. Eine so realisierte Animationssequenz lässt sich als Videodatei aufzeichnen.

### 5.2.3.2 Editieren von Modellen

- **Auswahl eines Elements**

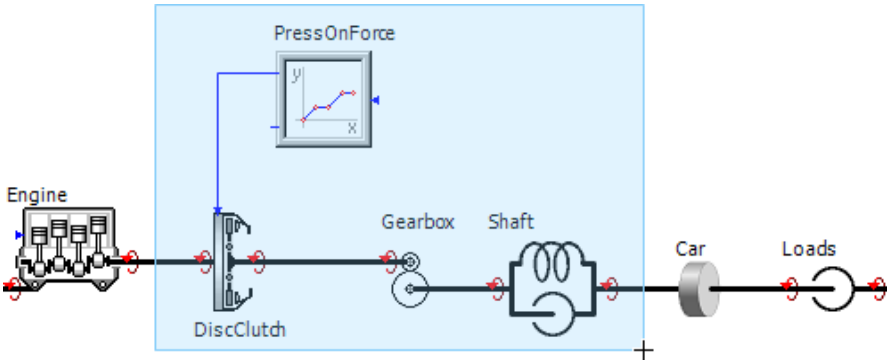
Um ein Element auszuwählen, klicken Sie es einfach an. Ein markiertes Element wird durch einen Rahmen optisch hervorgehoben. Einzelne Elemente können auch durch einen Klick auf den entsprechenden Eintrag im Modellexplorer ausgewählt werden (siehe **4.2.4.1**).



- **Auswahl mehrerer Elemente**

Um mehrere Elemente auszuwählen, gibt es zwei Möglichkeiten:

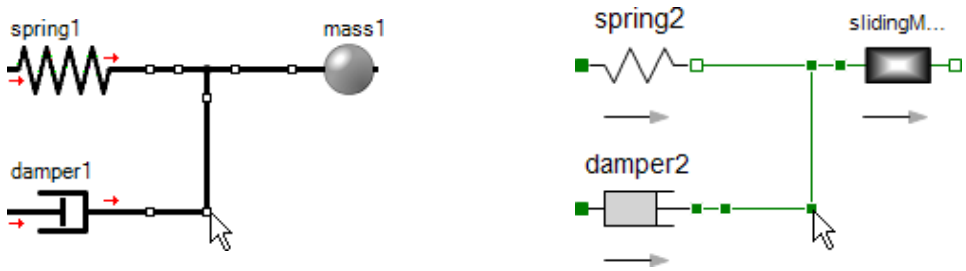
1. Ziehen Sie mit der Maus einen Rahmen um die Elemente, die Sie selektieren wollen.




2. Sie können ein Element zur aktuellen Auswahl hinzufügen, indem Sie es bei gedrückter *Shift*-Taste anklicken. Auf die gleiche Weise können Sie markierte Elemente aus der aktuellen Auswahl entfernen.

- **Auswahl einer Verbindung**

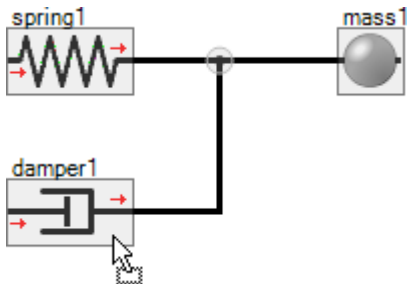
Um eine Verbindung auszuwählen, klicken Sie auf eine, der zur Verbindung gehörenden Linien. Eine markierte Verbindung erkennen Sie an den kleinen schwarzen Rechtecken. Alternativ dazu kann eine Verbindung auch durch einen Klick auf den entsprechenden Eintrag im Modellexplorer selektiert werden.
Eine Mehrfachauswahl von Verbindungen ist nicht möglich.




- **Verschieben von Elementen**

Ausgewählte Elemente können per *Drag & Drop* verschoben werden. Gehen Sie dazu wie folgt vor:

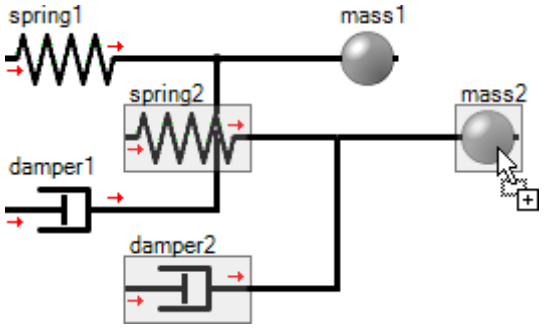
1. Klicken Sie mit der linken Maustaste auf ein aktuell ausgewähltes Element und halten Sie die Maustaste gedrückt.
2. Bei gedrückter Maustaste verschieben Sie die aktuelle Auswahl.
3. Lassen Sie die gewählten Elemente an der gewünschten Position fallen.




- **Kopieren von Elementen**

Neben dem Kopieren und Einfügen von Modellteilen über die Zwischenablage wird auch das Anlegen von Kopien mit *Drag & Drop* unterstützt. Gehen Sie dazu wie folgt vor:

1. Klicken Sie mit der linken Maustaste auf ein aktuell ausgewähltes Element und halten Sie die Maustaste gedrückt.
2. Halten Sie gleichzeitig die *Strg*-Taste gedrückt. Dadurch wird eine Kopie der aktuellen Auswahl erstellt, die Sie durch Bewegen der Maus an die gewünschte Position verschieben können.
3. Mit dem Loslassen der Maustaste wird der Kopiervorgang abgeschlossen.



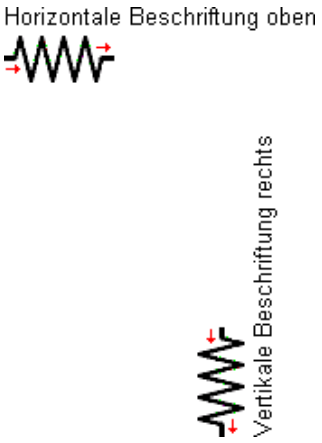

Eine begonnene *Drag & Drop*-Operation können Sie jederzeit abbrechen, indem Sie vor dem Loslassen der Maustaste die *Esc*-Taste betätigen.

### 5.2.3.3 Benennung von Elementen und Verbindungen

Die Beschriftung der Elemente und Verbindungen ist individuell den Erfordernissen anpassbar. Sie haben die Wahl zwischen den Einstellungen entsprechend **Tabelle 5.1**.

Beim Ändern der Namen von Modellkomponenten sollte bedacht werden, dass Verweise auf die betreffende Komponente innerhalb von Funktionsausdrücken oder Anweisungsfolgen möglicherweise nicht mehr aufgelöst werden können.

Auf der Registerkarte ***Allgemein*** des Eigenschaftsdialoges eines Elements oder Verbindung können Sie Namen und Kommentar der Modell-komponenten sowie die Lage und Ausrichtung des Beschriftungstextes von Elementen festlegen.




Die Art der Beschriftung aller Elemente ist global einstellbar (***Extras/Optionen/ Elemente***). Standardmäßig wird als Beschriftung der Name der Elemente ange-zeigt, dies ist jedoch für jedes Element abschaltbar.

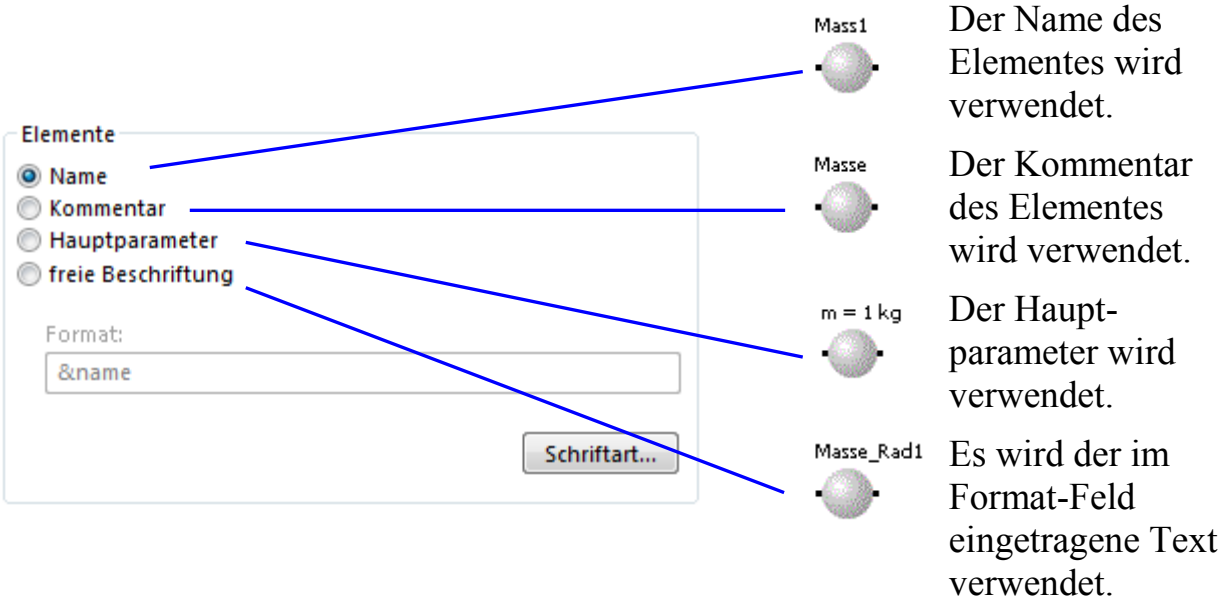



**Tabelle 5.1:** Beschriftung der Elemente

Weitere Optionen sind: Kommentar, Hauptparameter, freie Beschriftung.

Folgende Einträge können unter Formate genutzt werden:

| | |
|---|---|
| &name | Name des Elements |
| &comment | Kommentar des Elements |
| &mainparamvalue | Wert des Hauptparameter |
| &mainparamunit | Aktuelle Maßeinheit des Hauptparameters |

# 5.3 Parametrierung und Ergebnisgrößen

## 5.3.1 Eigenschaftsfenster

Eigenschaftsfenster dienen der Anzeige und Bearbeitung von Eigenschaften der Komponenten (Elemente, Verbindungen) eines Simulationsmodells.

Man öffnet diese Fenster mittels

- Doppelklick auf die gewünschte Komponente in der Strukturansicht oder im Modellexplorer,
- Menüleiste (Ansicht/Eigenschaften) oder Tastenkombination *Alt+Eingabe* oder
- Kontextmenü (Eigenschaften).

Zusammengehörige Eigenschaften werden in Registerkarten zusammengefasst. Mit den Tastenkombinationen *Strg+Tab* und *Strg+Tab+Umsch* kann zur nächsten bzw. vorhergehenden Registerkarte gewechselt werden.

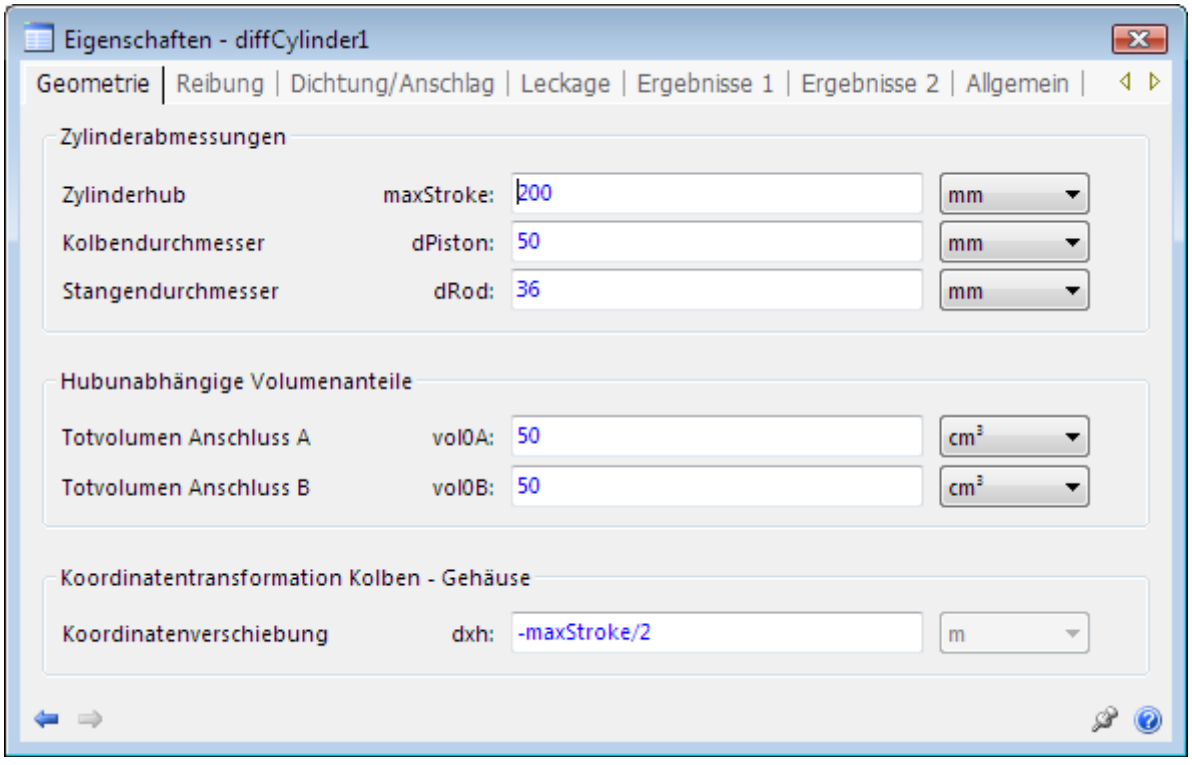



**Bild 5.7:** Eigenschaftsfenster

Die Fenstergröße ist variabel. Dabei wird abhängig von den Eigenschaften der jeweiligen Komponente eine bestimmte Mindestgröße vorgegeben. Position und Größe werden gespeichert, so dass beim erneuten Öffnen eines Eigenschaftsfensters dieses an gleicher Stelle mit gleicher Größe wie zuvor auf dem Bildschirm erscheint.
Standardwerte für Parameter und Einheiten werden blau dargestellt. Geänderte Eigenschaften einer Komponente werden sofort übernommen. Vorgenommene

Änderungen können mit der Schaltfläche rückgängig gemacht werden.
In der Symbolleiste am unteren Rand des Eigenschaftsfensters finden sie folgende Schaltflächen:

Durch einen Klick auf die Schaltfläche ***Hilfe*** wird die entsprechende Unterstützung der Online-Hilfe aktiviert.

Ein Klick auf die Schaltfläche ***Geöffnet halten*** (angepinnt) verhindert das automatische Schließen des Fensters bei der Auswahl einer anderen Komponente. Das Eigenschaftsfenster bleibt auch bei Änderung der aktuellen Auswahl geöffnet. Es werden immer die Eigenschaften der jeweils ausgewählten Komponenten angezeigt. Dieser Darstellungsmodus gilt so lange, bis das Fenster geschlossen oder die Schaltfläche erneut betätigt wird.

Ein Klick auf die Schaltfläche ***Zurück*** öffnet den Eigenschaftsdialog für die zuletzt bearbeitete Komponente.

Wenn Sie wieder zu der Komponente zurückkehren möchten, die vor dem Klicken auf die Schaltfläche ***Zurück*** bearbeitet wurde, klicken Sie auf die Schaltfläche ***Vorwärts***.

Das Fenster kann durch folgende Aktionen geschlossen werden:

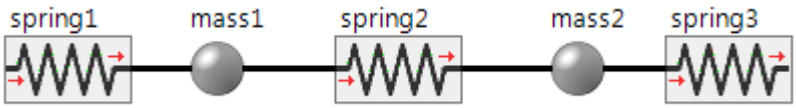
- Betätigung von *RETURN*, *ESCAPE* oder *Alt+F4*
- Klick auf die Schließen-Schaltfläche in der Titelzeile
- Verlust des Eingabefokus (z.B. durch Auswahl eines anderen Elements in einer Modellansicht)

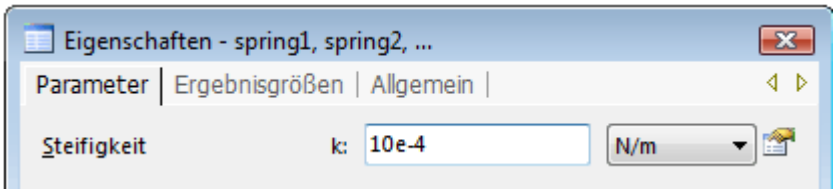
Solange sich das Fenster im angepinnten Zustand befindet, können die Tasten RETURN und ESCAPE nicht zum Schließen des Fensters verwendet werden.
Die Tasten haben hier folgende Funktionen:

- *RETURN* Übernahme der aktuellen Änderung
- *ESCAPE* Rücknahme der aktuellen Änderung.

Wird eine Fehlernachricht eingeblendet kann diese mit *ESCAPE* geschlossen werden. Zur Rücknahme der Änderung ist erneut die *ESCAPE* zu betätigen.
Das Schließen des Fensters ist nicht möglich solange eine fehlerhafte Eingabe vorliegt.

Das Eigenschaftsfenster unterstützt die **Mehrfachauswahl** von Komponenten.
In diesem Fall werden nur die Eigenschaften angezeigt, die in allen zur aktuellen Auswahl gehörenden Komponenten vorhanden sind. Ändern Sie eine solche Eigenschaft, wirkt sich das auf alle gewählten Komponenten aus. Die folgenden Bilder demonstrieren die gleichzeitige Änderung der Steifigkeit dreier Federn.




Über die Schaltfläche (alternativ über Modellexplorer und rechte Maustaste in eine Zeile, anschließend ***Bearbeiten der Attribute...*** wählen) gelangen Sie zu den Attributen von Parametern und Ergebnisgrößen. Hier können Sie zusätzliche Einstellungen zur Steuerung der Globalen Symbolischen Analyse (GSA) sowie des Solvers vornehmen.

Einstellungen sollten nur von erfahrenen Anwendern vorgenommen werden.

| Attribut | Bedeutung |
|---|---|
| *min, max* | Extrema des Parameters / der Variable (Ergebnisgröße) (die der Wert annehmen darf) |
| *start* | Startwert (Anfangswert);<br>wenn *fixed == true* beginnt die Berechnung mit diesem Wert |
| *fixed* | *true*: Solver wird gezwungen den Startwert zu benutzen<br>*false*: Solver darf Startwert gegebenenfalls ändern |
| *notFixed* | Zustand darf sich in Anfangswertberechnung nach einer Eventiteration ändern, auch wenn physikalische Gründe dagegen sprechen (z.B. Weg- oder Geschwindigkeitssprung);<br>- ist nur bei Zustandsgrößen relevant |
| *nominal* | Skalierungsfaktor für den Solver, falls Parameter / Variable Zustandsgröße ist (Nominalwert, z.B. Hydraulik: Nominalwert des Drucks = 1 bar, Wert ist anwendungsspezifisch und i.d.R. vom Anwender ignorierbar) |
| *stateSelect* | Spezifikation, ob eine Variable (bevorzugt) ein Zustand werden soll oder nicht (*never, avoid, default, prefer, always*) |
| *minNotReached,* | *true*: der Wert des *min*-Attributes darf nicht erreicht werden (z.B. *T.min = 0 'K'* (Temperatur) darf nie erreicht werden) |
| *maxNotReached* | *true*: der Wert des *max*-Attributes darf nicht erreicht werden |

| | |
|---|---|
| *absTol, relTol,* | Toleranzen von Zuständen für den Solver (falls nichts angegeben wird, werden die Toleranzen aus der Simulationssteuerung übernommen) |
| *discontChange* | Zustand darf sich während der Integration auch ohne Unstetigkeit sprungartig ändern |
| *quantity* | Klasse der physikalischen Einheit (vom Anwender in Modellelementen von *Externen Typen* änderbar; ansonsten in Modellkomponenten von *Internen Typen* nur änderbar, wenn *Basics.Unitless* eingestellt ist) |
| *comment* | Kommentar zum Parameter / zur Variable (vom Anwender stets änderbar) |
| *value* | |

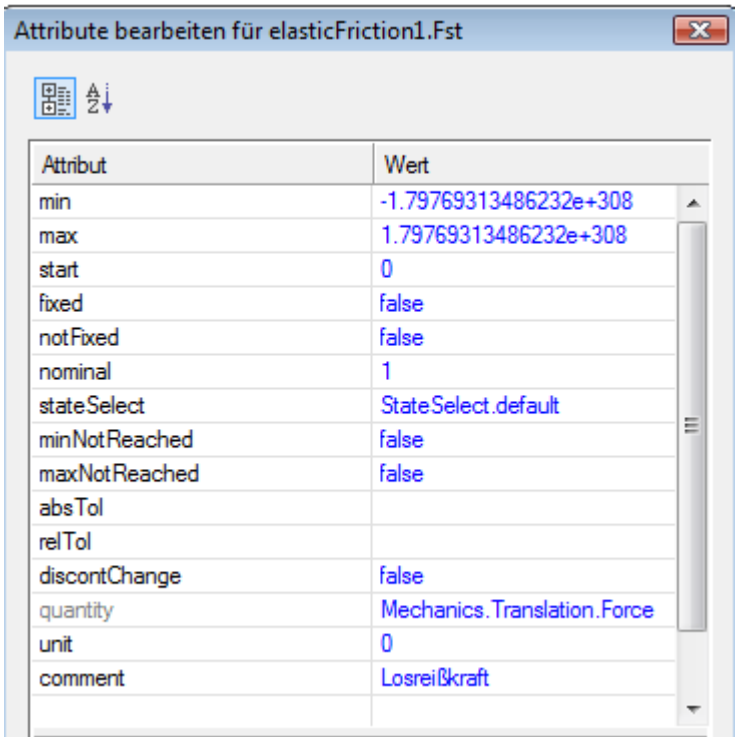


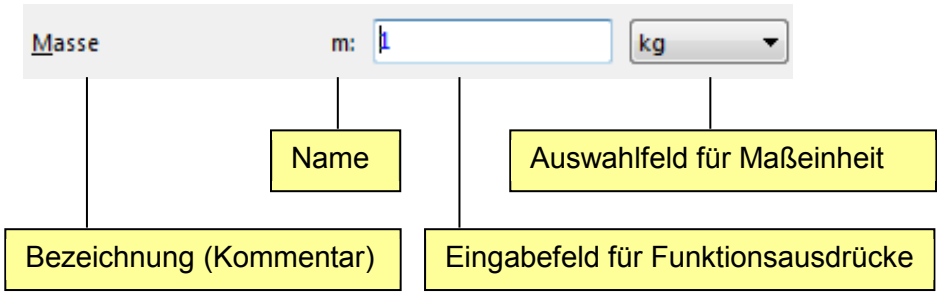

**Bild 5.8:** Beispiel für die Attribute eines Parameters

## 5.3.2 Parameter

Um Modellkomponenten zu parametrieren, verwendet SimulationX spezielle Steuerelemente. Die Beschriftung der Steuerelemente besteht aus einem innerhalb der jeweiligen Komponente eindeutigen Namen und einem Kommentar. Der Name kann in Funktionsausdrücken und Algorithmen verwendet werden. Während der Kommentar einer Größe sprachabhängig ist, bleibt deren Name in jeder Sprachversion von SimulationX gleich. Dies ermöglicht den Austausch von Modellen zwischen unterschiedlichen Sprachversionen, wobei die Bezeichnungen in der Originalsprache angezeigt werden.

Der Kommentar und gegebenenfalls die physikalische Größe lassen sich ebenfalls über den Attributedialog ändern (siehe Abschnitt **6.3.1**).

### 5.3.2.1 Parametereingabe




Wurde an einem Parameter etwas geändert, die Änderung wurde jedoch noch nicht übernommen, wird dieser Zustand wird durch einen angehängten * in der Titelzeile des Fensters angezeigt.

In SimulationX können Parameter nicht nur mit einfachen Zahlenwerten belegt werden. Vielmehr ist im Eingabefeld für Funktionsausdrücke auch folgendes anwendbar:

- **Mathematische Ausdrücke**
  Bis auf wenige Ausnahmen kann jeder Parametereintrag eine mathematische Funktion sein. Der Umfang der möglichen Notationen wird in **Kapitel 10.4.2** und **8.2.7** aufgezeigt.

  Beispiele für mathematische Funktionen

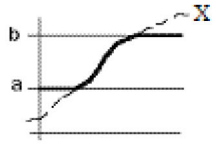
  | | |
  |---|---|
  | - Summe, Differenz | `5.23*mass1.x-50+function1.y` |
  | - Produkt, Quotient | `0.01*gravity*CarMass.m/2` |
  | - relative Abweichung | `2*(ctr1.x-ctr2.x)/(ctr1.x+ ctr2.x)` |
  | - gewichteter Mittelwert | `(a*x1+b*x2)/(a+b)` |
  | - natürlicher Logarithmus | `a*ln(x+b)` oder `a*log(x+b)` |
  | - Potenzfunktion | `0.31*CarMass.v^2/2` |
  | - Trigonometrische Funktion | `230*cos(2*pi*50*time)` |
  | - Exponentialfunktion | `1-exp(-time/10'ms')` |
  | - Kennlinien | `curve1.curve(ctr1.x)` |
  | - Wurzel | `0.01*sqrt(k)` |
  | - Betrag | `abs(x)` |

- **Logische Bedingungen**
  Jeder Parametereintrag kann logische Bedingungen enthalten. Siehe auch **Kapitel 10.4.2, 10.4.6** und **10.4.9.** Durch die Verknüpfung logischer Ausdrücke lassen sich auch komplexe Zusammenhänge abbilden.

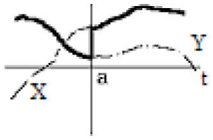
Beispiele für logische Operatoren

- Schaltfunktion `if(time<0.2) then 0 else 1`

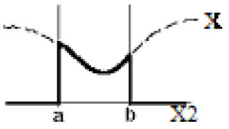
- Begrenzer `if x>b then b elseif x<a then a else x`




- Fallgrenze `if(time>a)then X else Y`




- Bereichsauswahl `if(X2>a)and(X2<b)then X else 0`

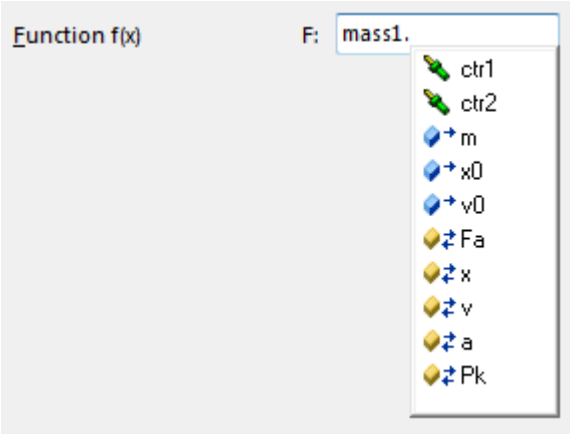



- unterschiedliche Steife für Zug und Druck

```
if (ctr1.x>ctr2.x) then k1 else k2
```

- **Variable**
  Bei der Parametrierung können alle im Modell bekannten Variablen, auch Ergebnisvariablen genutzt werden.

  Wird der Name eines Elements gefolgt von einem Punkt eingegeben, so zeigt die Autovervollständigung eine entsprechende Auswahl an.




Weiterführende Ausführungen zu den Namensräumen und Bezeichnern, sowie den nutzbaren Funktionen sind im **Kapitel 10.4** enthalten.

---

**Tipp:**
Ein Funktionsausdruck, der aufgrund seiner Länge nicht vollständig im Eingabefeld zu sehen ist, wird als Tooltipp vollständig dargestellt. Bewegen Sie dazu den Mauszeiger auf das Eingabefeld.

---

Bei der Eingabe von Ausdrücken wird das Auswahlfeld für die Maßeinheit deaktiviert (Feld wird grau). Das Ergebnis von Ausdrücken wird immer als SI-Wert interpretiert, solange im Ausdruck selbst nicht die gewünschte Maßeinheit explizit angegeben wurde.
Weiterführende Hinweise zu Maßeinheiten finden Sie im Abschnitt **5.3.2.2**.

Umschaltung der Einheiten mit Umrechnung

Steht im Eingabefeld eine einfache Zahl, erfolgt bei der Auswahl einer anderen Maßeinheit eine Umrechnung. Der Wert des Parameters, der in die Simulationsrechnung eingeht, ändert sich dadurch nicht.

**Bild 5.9:** Umschalten der Einheiten mit Umrechnung

Sie können diese Umrechnung unterdrücken, indem Sie während der Auswahl der neuen Maßeinheit die *Shift*-Taste gedrückt halten. Der Wert des Parameters, der in die Simulationsrechnung eingeht, ändert sich entsprechend der eingestellten Einheit.

**Bild 5.10:** Umschalten der Einheiten ohne Umrechnung

### 5.3.2.2 Physikalische Einheiten

Jeder Ausdruck kann mit einer Zeichenkette (in einfachen Anführungsstrichen), die eine physikalische Einheit beschreibt, erweitert werden.

Es gibt zwei Formen:

1. z. B. 1 'km': Der Wert wird als in der Einheit gegeben interpretiert und in die SI-Einheit umgerechnet, d. h. 1 'km' == 1000
2. z. B. 1 '->km': Der Wert wird als SI-Wert interpretiert und in die gegebene Einheit umgerechnet, d. h. 1 '->km' == 0.001.

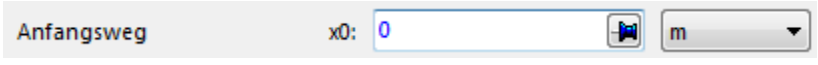
Beide Formen können auch gemeinsam benutzt werden, z. B. 1 'km^2->ha' == 100. Die Existenz der eingegeben Einheit wird mit Hilfe einer Datenbank überprüft, dabei werden alternative Schreibweisen für eine Einheit zugelassen, z. B. 1 'km²' == 1 'km^2' == 1 'km2'; 1 'N/m2' == 1 'Nm-2'
Zusammengesetzte Werte amerikanischer Maßeinheiten können direkt eingegeben werden, z. B. 6 'ft' 6 'in' == 1.9812 'm'.

### 5.3.2.3 Konstanten

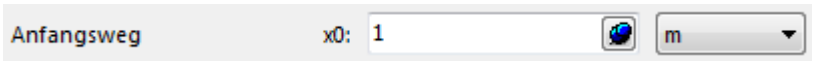
| | |
|---|---|
| `pi` | 2*arccos(0) == 3.14159265358979... |
| `e` | exp(1) == 2.71828182845905... |

## 5.3.3 Anfangswerte

Wenn es sich bei einem Parameter um einen Anfangswert handelt, so erscheint das Steuerelement für Parameter in abgewandelter Form:



Der Parameter enthält einen Pin. Wenn der Pin lose (frei) ist (s.o.) dann ist der Startwert als Hinweis für den Solver zu verstehen. Er beginnt mit diesem Wert die Suche (Iteration) nach konsistenten Anfangswerten. Der Wert für den Start der Simulation kann in diesem Fall vom eingegebenen Anfangswert abweichen.



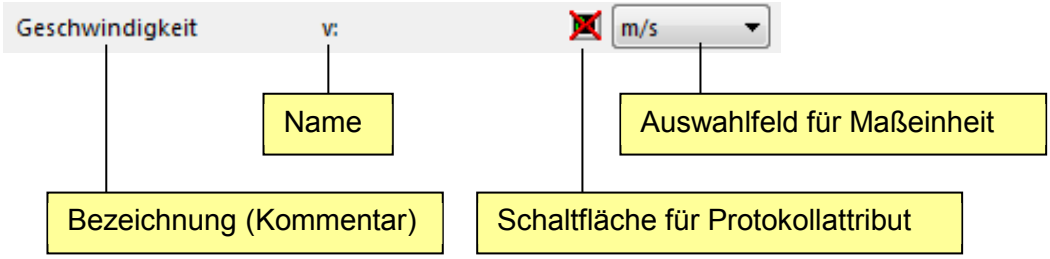

Bei festem Pin (fixiert) beginnt die Simulation exakt mit dem eingegebenen Anfangswert. Sind sich widersprechende Anfangswerte fixiert, erhält man eine entsprechende Fehlermeldung (siehe **Kapitel 11.2**).
Wenn kein Wert eingetragen ist, wird er immer als frei interpretiert.

In das Eingabefeld für Startwerte können **nur konstante Ausdrücke** eingegeben werden.

### 5.3.4 Ergebnisgrößen

Jedes Element eines Modells errechnet in Abhängigkeit von seiner Funktionalität und Komplexität eine Anzahl von Ergebnisgrößen. Zu jedem Ausgabezeitschritt werden alle Ergebnisgrößen mit aktivem Protokollattribut aller Elemente des Modells bereitgestellt.

- **Steuerelement für Ergebnisgrößen**




Das Protokollattribut einer Ergebnisgröße kann mit einem Klick auf die zugehörige Schaltfläche ein- bzw. ausgeschaltet werden.
Der Verlauf einer protokollierten Größe kann in Ergebnisfenstern dargestellt oder in einer Datei gespeichert werden. Die gewählte Maßeinheit gilt als Voreinstellung für die Anzeige. Sie hat für die Simulationsrechnung keine Bedeutung.

### 5.3.5 Globale Parameter

Diese Parameter sind global im Modell definiert und mit Standardwerten belegt.

| Name | Kommentar | Standardwert |
|---|---|---|
| `gravity` | Gravitationsbeschleunigung | 9.80665 m/s² |
| `gravity3D` | Gravitationsbeschleunigungsvektor (3D) | {0.0,0.0,gravity} m/s² |
| `pAtm` | Atmosphärendruck | 1.01325 bar |

| | | |
|---|---|---|
| **TAtm** | Atmosphärentemperatur | 20 °C |
| **iSim** | Nummer des Simulationslaufs (wird vor jeder Simulation erhöht) | 0 |
| **nSim** | Name der Simulation (wird bei Variantenrechnung verwendet) | leer |

# 5.4 Weiterführende Themen

## 5.4.1 Beschreibung nichtlinearer Eigenschaften über Kraftkennlinie

Oft sind die nichtlinearen Eigenschaften von Bauteilen oder Komponenten durch Kraft- bzw. Momentenkennlinien beschrieben. Beispiele dafür sind Moment-Verformungskennlinien von Kupplungen mit nichtlinearen Steifigkeitseigenschaften oder Schlupfkennlinien bei Kontaktproblemen.
Veränderliche Kräfte/Momente lassen sich am besten mit den Elementen der Mechanik *Äußere Kraft* bzw. *Äußeres Moment* modellieren.

### 5.4.1.1 Beschreibung über Formeln

Formelmäßig beschriebene Zusammenhänge können direkt in den Parameterdialog eingetragen werden. Ist das Moment einer Kupplung mit der Formel

$$M_{iKupplung} = k_0 \cdot (\varphi + a \cdot \varphi^3) \qquad \textbf{(5.1)}$$

gegeben, steht im Parameterdialog des *Äußeren Momentes* beispielsweise folgender Eintrag:

```
k0 * ((self.ctr1.phi - self.ctr2.phi) + a * (self.ctr1.phi - self.ctr2.phi)^3)
```
**(5.2)**

oder alternativ:

```
k0 * (self.dphi + a * self.dphi^3)
```
**(5.3)**

**Bild 5.11:** *Äußeres Moment* mit Formel für Momentenkennlinie parametriert

Dabei sind *k0* und *a* nutzerdefinierte Parameter, die zuvor im Parameterdialog des Modells angelegt wurden. Über die Bezeichner `self.ctr1.phi` und `self.ctr2.phi` kann in der Bibliothek Mechanik vom Element aus auf die Verformung der angeschlossenen Trägheiten zugegriffen werden. Da φ die Relativ- oder Differenzverdrehung der beiden Kupplungshälften bezeichnet, muss dafür im Parametereintrag `ctr1.phi-ctr2.phi` geschrieben werden.

Alternativ kann auch auf die Ergebnisgröße des Elements referenziert, also dphi geschrieben werden.

Das so parametrierte Element *Äußeres Moment* wird zwischen zwei rotatorische Trägheiten geschaltet (**Bild 5.12**).

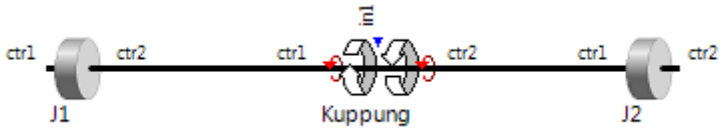



**Bild 5.12:** Modellstruktur für Kupplung mit nichtlinearer, über Formel beschriebener Momentenkennlinie

Die Drehträgheiten sind jeweils mit der Trägheit einer Kupplungshälfte parametriert. Außerdem werden an den Trägheiten die Verformungsgrößen (Winkel, Winkelgeschwindigkeit und Winkelbeschleunigung) der Kupplungshälften berechnet. Siehe auch Beispieldatei *Mechanics/ NonlinearCoupling.ism* Beschreibung über Kennlinie.
Gegeben sei beispielsweise folgende Kennlinie einer mehrstufigen Kupplung.

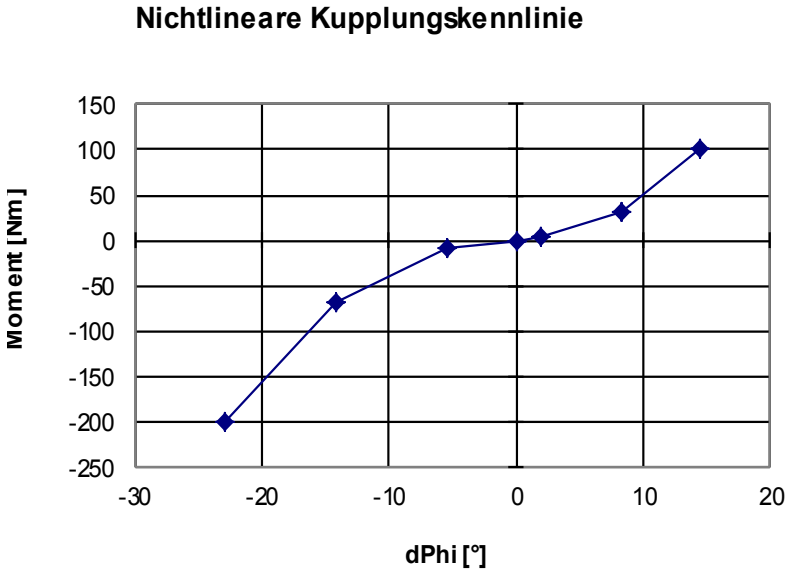



**Bild 5.13:** Punktweise gegebene Kennlinie

Punktweise gegebene Zusammenhänge lassen sich in SimulationX am besten mit dem Element *Kennlinie* (Bibliothek *Signalquellen)* beschreiben, indem man da die Punkte einträgt oder vom Datenträger lädt (**Bild 5.14)**.

Das Moment ist in Abhängigkeit von dem Verformungs- bzw. Differenzwinkel der Kupplungshälften $\Delta\varphi$ beschrieben. Demzufolge muss für das Objekt *Kennlinie* $\Delta\varphi$ die Bezugsgröße (x-Achse) sein. Entsprechend wird im Dialog nicht die Simulationszeit sondern der Signaleingang x als Bezugsgröße gewählt (**Bild 5.14**). Außerdem wird $\Delta\varphi$ auf den Eingang des Signalgliedes geführt (**Bild 5.15)**. Zuvor muss $\Delta\varphi$ bereitgestellt werden. Dazu werden die Winkel beider Kupplungshälften

über Aufnehmer abgegriffen. Die Berechnung der Winkeldifferenz kann an einem Summationspunkt erfolgen.

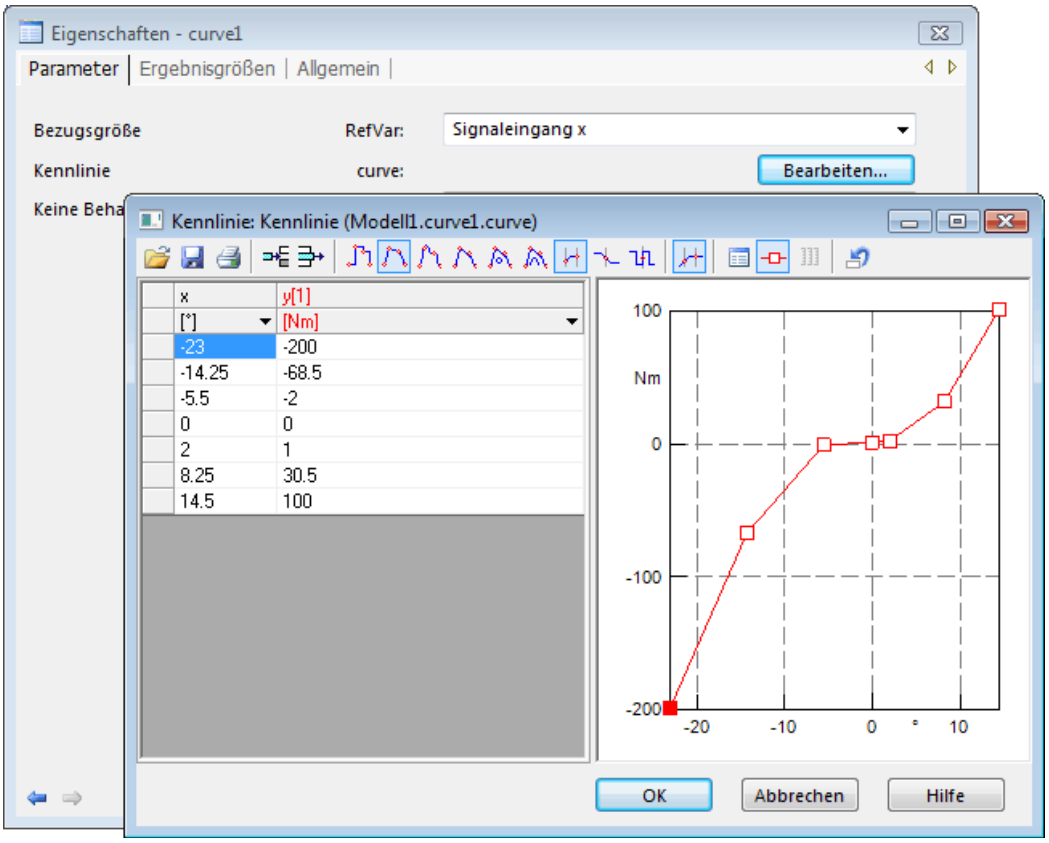



**Bild 5.14:** Abbildung punktweise gegebener Zusammenhänge mit dem Objekt *Allgemeine Kennlinie*

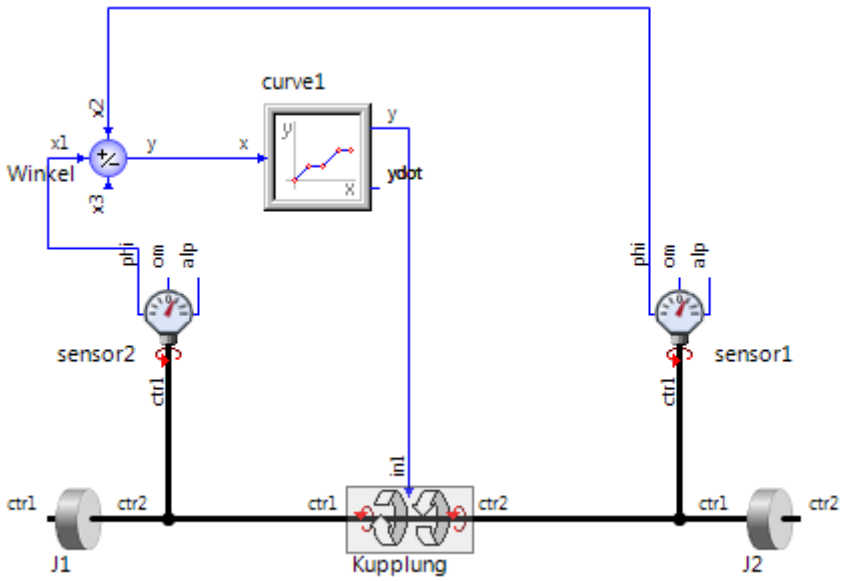



**Bild 5.15:** Struktur eines Kupplungsmodells mit punktweise gegebener Kennlinie

Die Ausgangsgröße des Elements *Allgemeine Kennlinie* (Kennlinie in **Bild 5.14**) ist das Kupplungsmoment. Es wird per *Äußeres Moment* (Kupplung) zwischen die Kupplungshälften gebracht. Das *Äußere Moment* ist mit dem Signaleingang `in1` parametriert, der sich auf die am Signaleingang `in1` anliegende Größe bezieht. Die Kupplungshälften sind mit Drehträgheiten modelliert (J1 und J2), welche mit der Trägheit der Kupplungshälften parametriert sind. An den Drehträgheiten werden auch die Bewegungsgrößen der Kupplung berechnet.

An J1 wird ein Anfangswert von 2 U/s vorgegeben.

Führt man eine Simulationsrechnung aus und stellt das Moment über dem Differenzwinkel dar, erhält man die eingegebene Kennlinie für den während der Rechnung durchlaufenen Bereich (**Bild 5.16**).

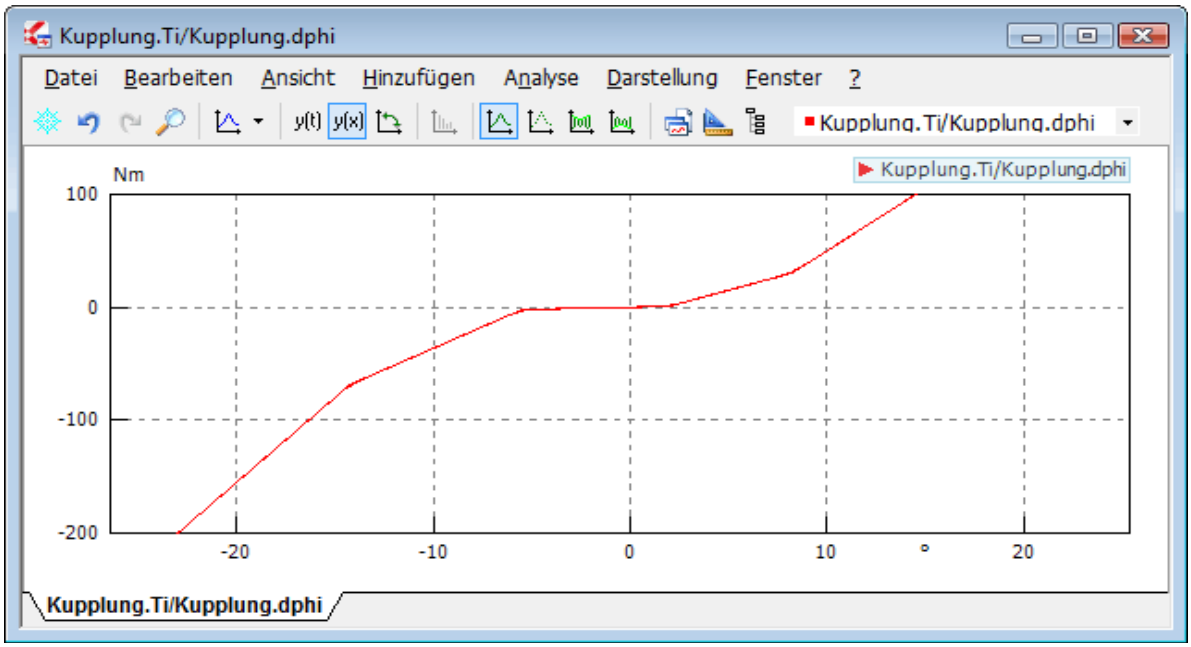



**Bild 5.16:** Simulationsergebnis Kupplungskennlinie

## 5.4.2 Veränderliche Elementparameter

### 5.4.2.1 Parametrierung der Elemente mit veränderlichen Parametern - Parameter sind effektive Größen

Im allgemeinen Fall, d.h. wenn die Elemente nicht speziell auf veränderliche Parameter zugeschnitten sind, geht der in SimulationX verwendete Lösungsalgorithmus von konstanten Parametern aus.

Ein Beispiel hierfür sind Federelemente. Im Federelement wird die Kraft $F$ wie folgt berechnet:

$$F = k \cdot (x_1 - x_2) = k \cdot x\,, \qquad \textbf{(5.4)}$$

bzw. das Moment M für eine Drehfeder:

$$M = k \cdot (\varphi_1 - \varphi_2) = k \cdot \varphi \,. \qquad \textbf{(5.5)}$$

Stellt man **(5.5)** nach der Steifigkeit k um, erhält man

$$k = \frac{M}{\varphi}, \qquad \textbf{(5.6)}$$

was exakt für konstante Steifigkeiten gilt.
Will man im Federelement mit veränderlichen Steifigkeiten arbeiten, muss man also die sogenannte effektive Steife $k_{eff}$ einsetzen, da sie gleichfalls als Quotient aus Moment (bzw. Kraft) und Verformung (φ bzw. x) definiert ist:

$$k_{eff}(\varphi) = \frac{M(\varphi)}{\varphi}. \qquad \textbf{(5.7)}$$

Die effektive Steifigkeit entspricht damit dem Anstieg einer Geraden, die durch den Arbeitspunkt und den Nullpunkt der M-φ-Kennlinie geht (**Bild 5.17)**.
Dabei kann $k_{eff}$ durchaus eine Funktion sein, muss also nicht konstant sein. Für den mit **(5.1)** gegebenen Zusammenhang für das Moment in der Kupplung würde sich beispielsweise folgende effektive Steifigkeit ergeben:

$$\begin{aligned} k_{eff} &= \frac{M_{iKupplung}}{\varphi} = \frac{k_0 \cdot \left(\varphi + a \cdot \varphi^3\right)}{\varphi} \\ k_{eff} &= k_0 \cdot a\left(1 + a \cdot \varphi^2\right) \end{aligned} \,. \qquad \textbf{(5.8)}$$

Parametriert man die Steife eines Federelementes mit dem Ausdruck $k_0 \cdot a\left(1 + a \cdot \varphi^2\right)$, ist der nichtlineare Zusammenhang exakt beschrieben. Eine Integration über den Parameter ist nicht erforderlich. Anfangswerte für den Start der Simulation (Integrationskonstanten, z.B. **(5.10)**) müssen nicht bekannt sein.
Im folgenden Abschnitt wird darauf eingegangen, wie verfahren werden kann, wenn die Steifigkeit als Ableitung gegeben ist.

### 5.4.2.2 Integration über veränderliche Parameter-Parameter sind als Ableitungen gegeben

Wenn veränderliche Parameter als Ableitung gegeben sind, muss eine Integration über diese Parameter erfolgen, was wieder am Beispiel einer nichtlinearen Feder demonstriert werden soll.

Allgemein ist die Federsteifigkeit k mit folgendem Zusammenhang definiert:

$$k = \frac{dM}{d\varphi}, \qquad \textbf{(5.9)}$$

wobei k die Ableitung des Momentes M nach dem Winkel φ ist. Die Steifigkeit entspricht damit dem Anstieg der Tangente an den Arbeitspunkt der M-φ-Kennlinie (**Bild 5.17**).

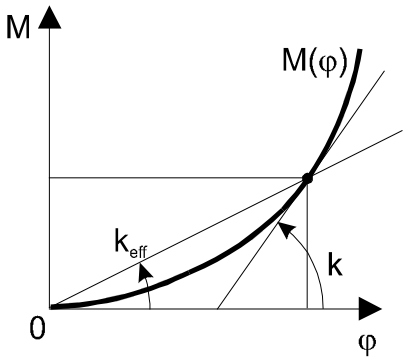



**Bild 5.17:** Steifigkeit k und effektive Steifigkeit $k_{eff}$ für nichtlineare Moment-Verformungs-Kennlinien M(φ)

Eine Umstellung von **(5.9)** ergibt die Beziehung für das Moment:

$$M = \int k d\varphi + M_0 . \qquad \textbf{(5.10)}$$

$M_0$ ist dabei die Integrationskonstante.

Man berechnet also im Modell das Moment per Integration. Dafür stehen entsprechende Signalglieder zur Verfügung.

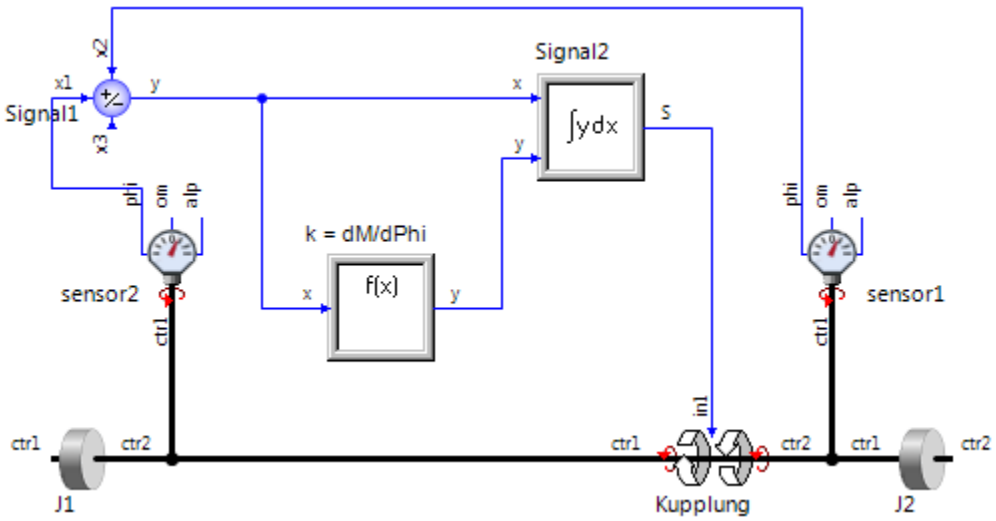



**Bild 5.18:** Modellstruktur für Kupplung mit nichtlinearer Steife – Integration der Steife über dem Winkel

Will man obige Integrationsformel direkt umsetzen, muss man **über den Differenzwinkel φ integrieren**. Wie man diesen bekommt, wurde bereits unter **5.4.1.2** beschrieben und ist in **Bild 5.18** dargestellt. Die Integration erfolgt im Signalglied *Integral y über x* der Bibliothek *Spezielle Signalglieder*. Das am

Ausgang anliegende Moment wird wieder per *Äußeres Moment* (Kupplung) zwischen die Drehträgheiten der Kupplungshälften (J1 und J2) gebracht.

Im Allgemeinen wird jedoch bei der Simulation über der Zeit integriert. Man kann das Integral für das Moment auch dafür aufschreiben. Das Moment als **Integral über der Zeit (5.13)** erhält man aus **(5.10)** , indem man **(5.12)** einsetzt.

$$M = \int (k \cdot \omega) dt + M_0 \text{, mit} \qquad \textbf{(5.11)}$$

$$d\varphi = \omega dt \text{,} \qquad \textbf{(5.12)}$$

wobei ω die Geschwindigkeit ist. Das Simulationsmodell sieht dann beispielsweise wie in **Bild 5.19** dargestellt aus.

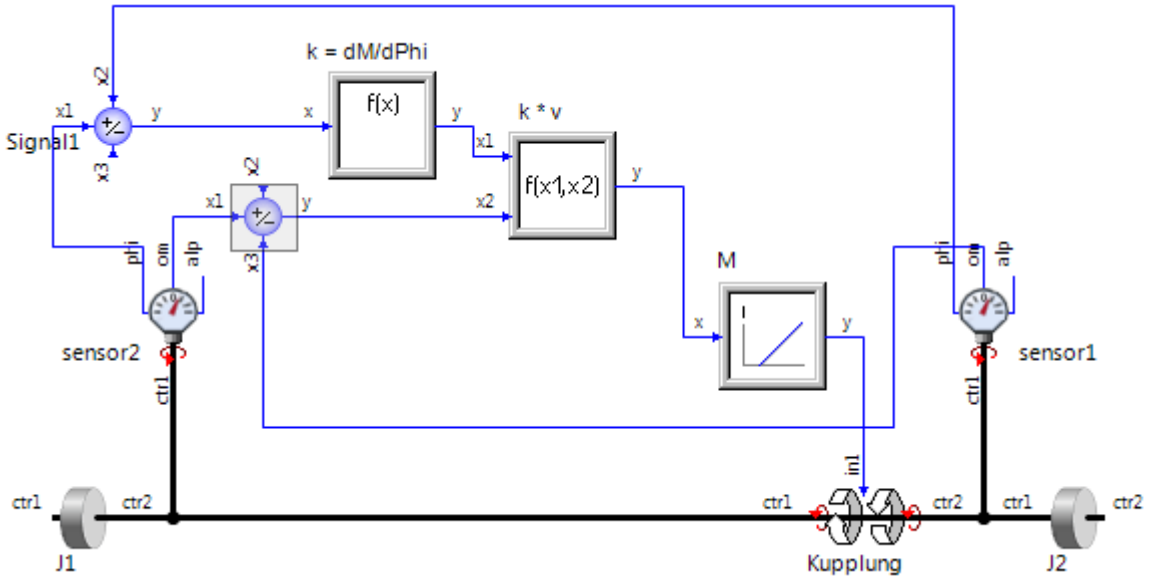



**Bild 5.19:** Modellstruktur für Kupplung mit nichtlinearer Steife – Integration der Steife über der Zeit

Für die Integration über der Zeit wird das *I-Übertragungsglied* der Bibliothek *Lineare Signalglieder* benötigt.

Anmerkungen:

- Bei der Integration muss das innere Moment der Feder für den Simulationsstartzeitpunkt bekannt sein und eingegeben werden (was bei Verwendung der effektiven Steifigkeit nicht erforderlich ist, vgl. **(5.7)**). Oft beginnt die Simulation jedoch im unverformten Zustand, so dass das innere Moment zu Beginn der Simulation und somit die Integrationskonstante Null sind.

- Das *Äußere Moment* (Kupplung in **Bild 5.18** und **Bild 5.19**) ist mit der am Eingang anliegenden Signalgröße `in1` parametriert.

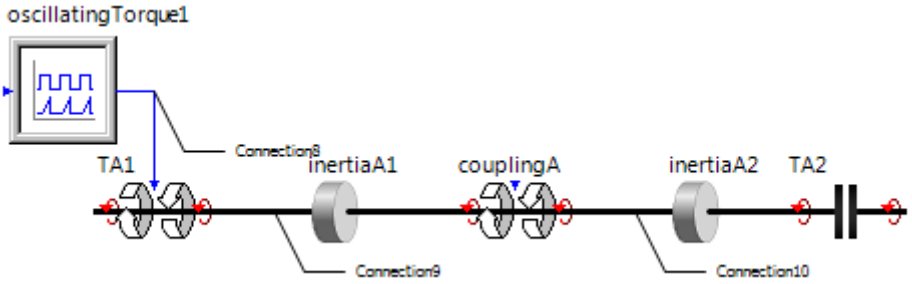
– Wenn sich der als Ableitung gegebene Zusammenhang geschlossen integrieren lässt, kann die Steife *k* beispielsweise nach **(5.13)** in eine effektive Steife $k_{eff}$ umgerechnet werden.

$$k_{eff} = \frac{\int k\, d\varphi}{\varphi} \quad \text{bzw.} \qquad k_{eff} = \frac{\int k\, dx}{x} \tag{5.13}$$

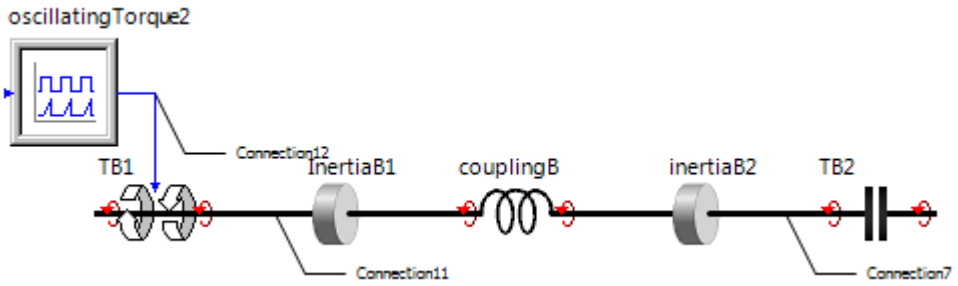
In der Beispielsammlung von SimulationX im Modell *Mechanics/ NonlinearCoupling.ism* sind für den gegebenen nichtlinearen Zusammenhang $M_{iKupplung} = k_0 \cdot (\varphi + a \cdot \varphi^3)$ drei Modellierungsmöglichkeiten zusammengestellt.

Gegeben: Mi = k0 * (phi + a * phi^3)

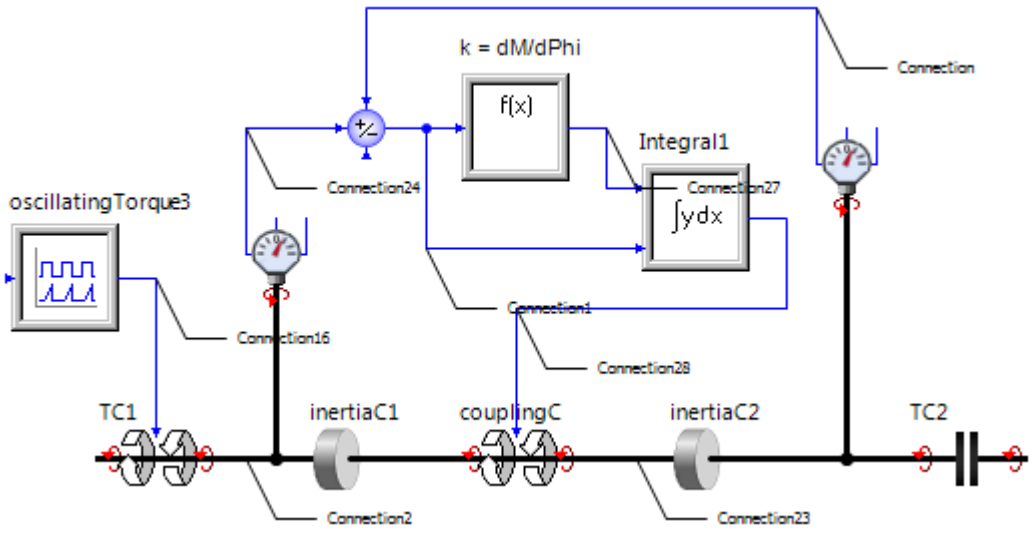
***Lösung 1:*** Das Kupplungsmoment ist abhängig von der Drehwinkeldifferenz
-> Verwendung eines Kraftelementes mit Formel




***Lösung 2:*** Die effektive Kupplungssteifigkeit ist bekannt. Sie kann in ein Feder-Element eingetragen werden.




***Lösung 3:*** Wird das innere Moment nach dem Winkel phi abgeleitet, erhält man die tatsächliche Steifigkeit. In diesem Fall muss integriert werden und es bieten sich Lösungen gemäß **Bild 5.18** bzw. **Bild 5.19** an.




Sowohl die Ergebnisverläufe (**Bild 5.20**) als auch das dynamische Verhalten aller drei Modellierungen sind identisch. Somit sind all diese Modellierungen bezüglich der Ergebnisse gleichwertig. Der Anwender kann, je nach Neigung oder Anwendung, seine Modellierungsvariante auswählen.

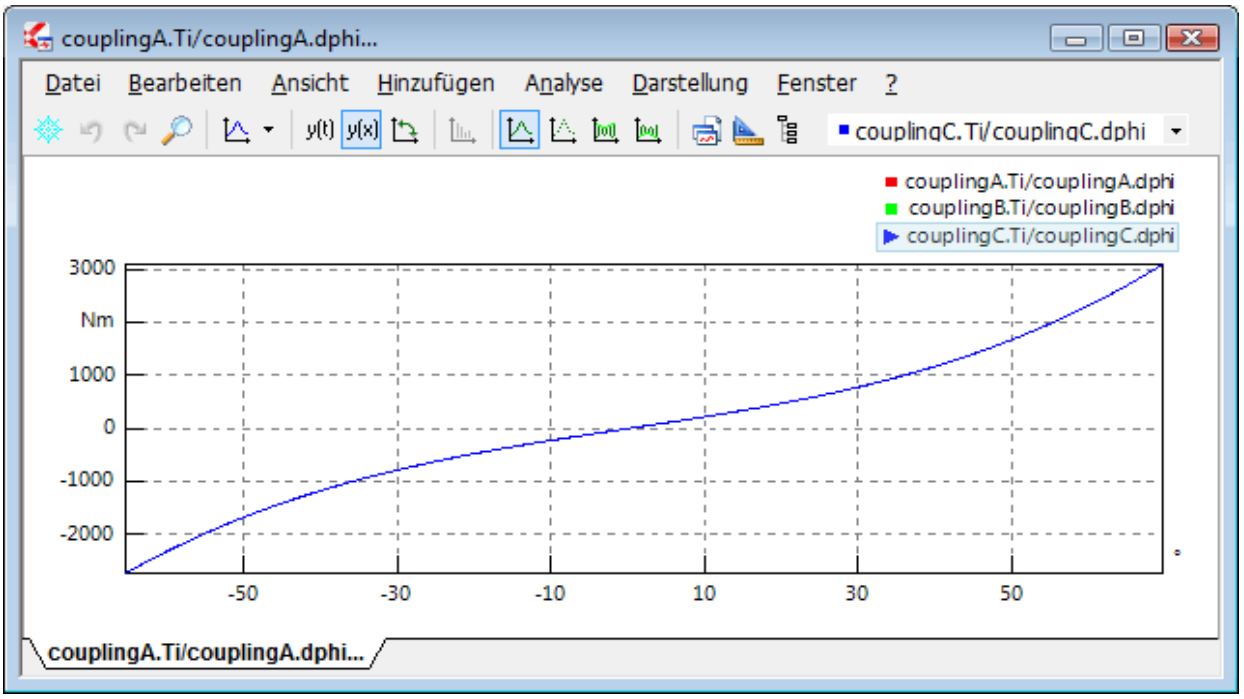



**Bild 5.20:** Identische Ergebnisse verschiedener Varianten der Modellierung nichtlinearer Zusammenhänge im Element

Die hier vorgestellten Lösungen sind beispielhaft. Selbstverständlich ergeben sich weitere Lösungsmöglichkeiten unter Verwendung des Type-Designers oder unter Nutzung des Funktionsumfanges von Modelica.

## 5.4.3 Arbeiten mit Kennlinien und Kennfelder

Mehrere Elemente verschiedener Domänen (z. B. Signalquellen, Proportional-Wege-Ventile...) besitzen die Option zur Parametrierung mittels Kennlinien, Kennliniensätzen, Kurvenscharen und Kennfeldern.

### 5.4.3.1 Kennlinie

Kennlinien gestatten die punktweise Beschreibung von Verläufen und Abhängigkeiten. Sie bieten umfangreiche Möglichkeiten zur Parametrierung, die nachfolgend im Detail beschrieben sind.

- ***Kennliniendialog:***

Mit der Schaltfläche ***Bearbeiten...*** öffnen Sie einen Kennliniendialog, der die nähere Beschreibung des zu erzeugenden Signals erlaubt.

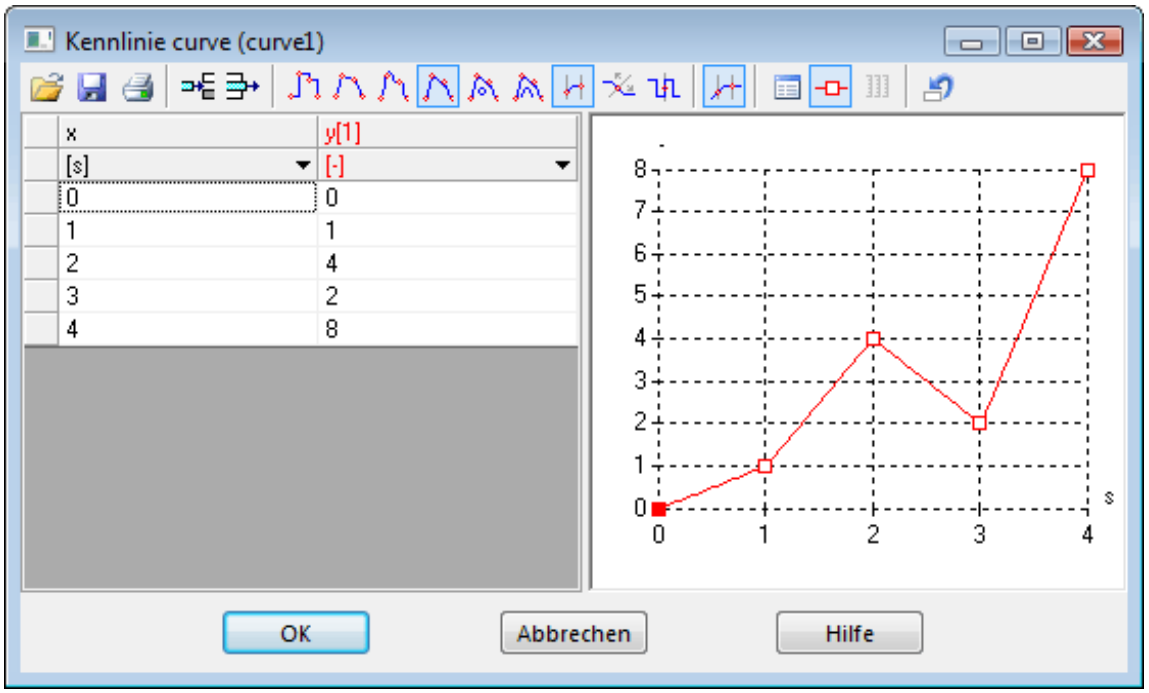



**Bild 5.21:** Kennliniendialog

- ***Kennlinieneingabe, Datenimport und -export:***

Für die Eingabe der Kennlinie und Kennliniensätze existieren mehrere Möglichkeiten, Datenpunkte zu erzeugen bzw. zu verändern:

***Eintragen/Ändern*** von Wertepaaren in der Tabelle

Doppelklick in die grafische Darstellung zur Erzeugung von Punkten und Verschieben der Punkte durch Ziehen mit der Maus

***Laden*** einer Kennlinie aus einer Datei (unterstützte Formate: Text (*.txt, *.csv), ITI-SimX-Format (*.rsf), IEEE-Binärformat (*.bin), ITI-Binärformat (*.rfb), ITI-ASCII-Format (*.rfu))

Kennlinien und Kennliniensätze können in den genannten Formaten gespeichert und auch gedruckt werden (siehe auch **Kap. 9.1**).
Für das Editieren der Kennlinie in der Tabelle stehen folgende Tastenbelegungen zur Verfügung:

| | |
|---|---|
| *Tab*-Taste | bewegt Cursor zur nächsten Zelle |
| *Shift+Tab* | bewegt Cursor zur vorhergehenden Zelle |
| *ESC*-Taste | bricht aktuellen Editiervorgang ab und schließt den Kennliniendialog |
| *Enter*-Taste | beendet aktuellen Editiervorgang |
| *Cursortasten auf/ab* | bewegen den Cursor innerhalb einer Spalte nach oben/unten |
| *Cursortasten links/rechts* | bewegen den Cursor innerhalb einer Zeile nach links/rechts |

Tabellenzeilen können mit den beiden Schaltflächen zum ***Einfügen*** und ***Löschen*** im **Toolbar** hinzugefügt oder entfernt werden.

Eine Zeile wird durch einen Mausklick auf die jeweils erste Spalte markiert (Zeile erhält eine Umrahmung). Bewegt man sich mit der *Tab-Taste* über die letzte Spalte der Tabelle hinaus, wird automatisch eine neue Zeile angefügt. Der aktuelle Kennlinien-Verlauf wird in der Vorschau grafisch dargestellt, wobei der in der Tabelle markierte Punkt in der Grafik ebenfalls markiert ist.

| | |
|---|---|
| *Shift*-Taste | Maus lässt sich nur in Y-Richtung bewegen, X-Wert bleibt konstant |
| *Strg*-Taste | Maus lässt sich nur in X-Richtung bewegen, Y-Wert bleibt konstant |

Der in der Vorschau jeweils bearbeitete Punkt ist in der Tabelle ebenfalls markiert.

- ***Interpolation bzw. Approximation zwischen den Stützstellen:***

Die Einstellung des jeweiligen Modus erfolgt durch Betätigen des entsprechenden Knopfes auf dem **Toolbar**.

Die einstellbaren Modi wirken folgendermaßen:

Treppenfunktion:

Der Funktionswert bleibt vom aktuellen Datenpunkt bis zum nächsten konstant. Bei unterschiedlichen Funktionswerten hat der Funktionswert an den Stützstellen einen Sprung. Alle Ableitungen sind konstant null und somit stetig.

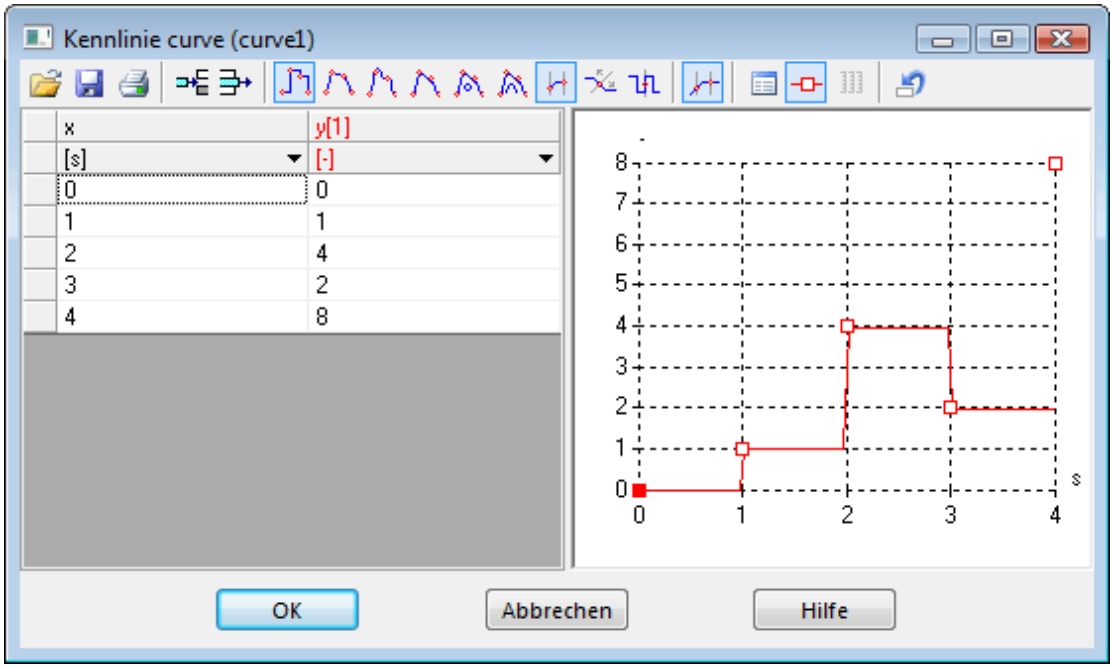



**Bild 5.22:** Kennlinie mit Treppenfunktion

Lineare Interpolation:

Die eingegebenen Stützstellen werden mit Geraden verbunden.

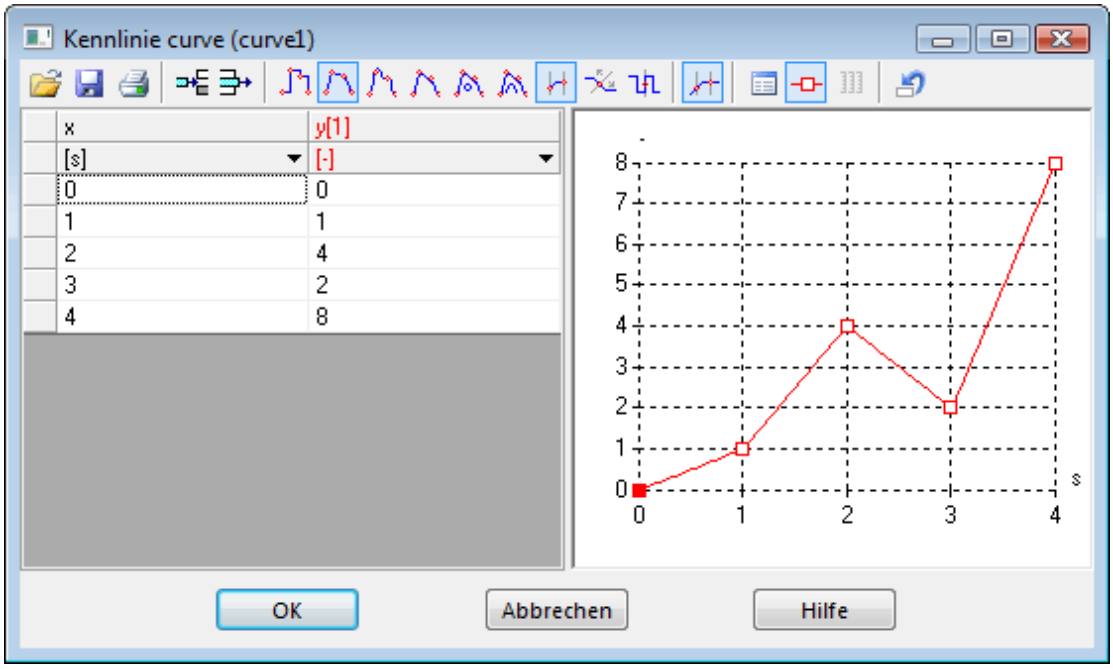



**Bild 5.23:** Kennlinie mit linearer Interpolation

Spline-Interpolation:

Die eingegebenen Stützstellen werden durch Splines (Polynome 3. Grades) verbunden. Damit ergibt sich ein glatter Verlauf der Kennlinie / Kurve (1. und 2. Ableitungen sind stetig, 3. Ableitung hat Sprünge in den Stützstellen).
Die erste Ableitung (bei Wegvorgabe die Geschwindigkeit) ist stückweise

quadratisch. Die zweite Ableitung (bei Wegvorgaben die Beschleunigung) ist stückweise linear.

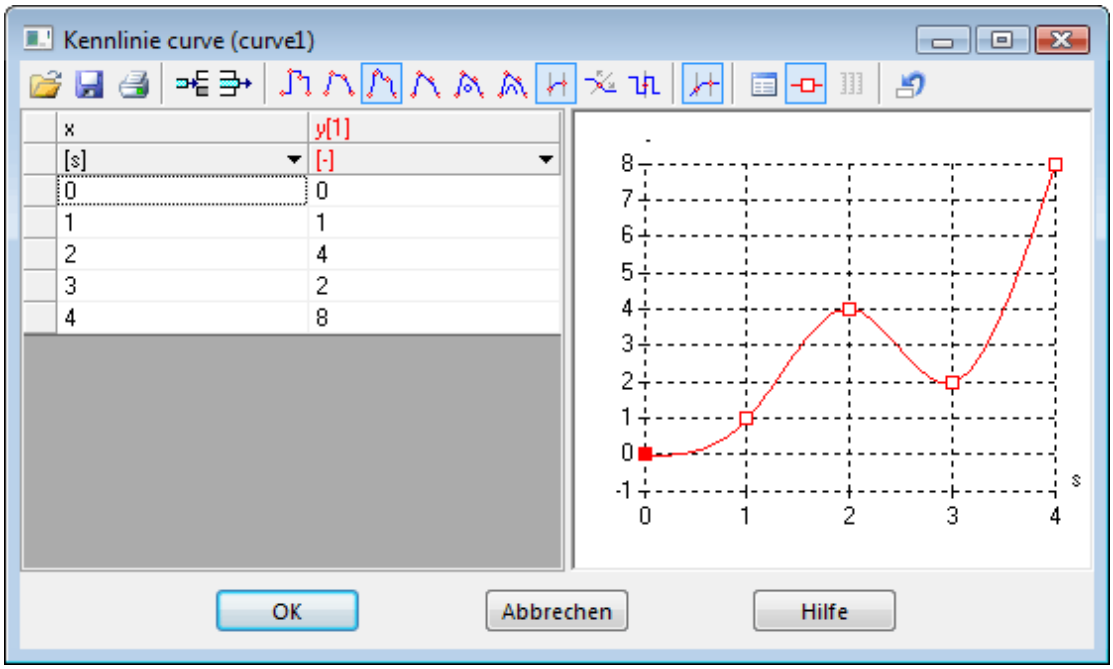



**Bild 5.24:** Kennlinie mit Spline-Interpolation

Hyperbolische Approximation:

Die Kurve wird an die Stützstellen mit einem hyperbolischen Verlauf angenähert. Die zulässige Abweichung wird dabei durch den Parameter Approximations-Toleranz bestimmt.
Die Hyperbelstücke gehen stetig und stetig differenzierbar in die benachbarten Hyperbelstücke oder in die Verbindungsgeraden über. Der Übergang erfolgt links und rechts mit dem gleichen Abstand von der Stützstelle auf der x-Achse. Dieser Abstand ist die Hälfte der kleineren Intervalllänge der beiden Intervalle, die sich in der Stützstelle berühren. Beim längeren Intervall geht somit das Hyperbelstück in die Verbindungsgerade über, bevor das nächste Hyperbelstück erreicht wird. In den Randintervallen wird von der Randstützstelle bis mindestens zum Intervall-mittelpunkt bei nicht periodischer Fortsetzung ebenfalls ein Stück der Ver-bindungsgeraden verwendet.

Die ***Approximations-Toleranz*** (im abgebildeten Kurvenbeispiel ist ein Wert von 1000 eingestellt) ist ein Maß für den Abstand des Scheitelpunktes des jeweiligen Hyperbelstücks vom Datenpunkt. Vereinfacht ausgedrückt: Je größer die Zahl ist, desto größer ist der Abstand. Allerdings gibt es einen Maximalabstand, um die genannten Bedingungen erfüllen zu können. Deshalb führt eine weitere Vergrößerung nicht zu einer Änderung der Kurve.
Wenn Stücke der Verbindungsgeraden verwendet werden, ist die erste Ableitung (bei Wegvorgabe die Geschwindigkeit) konstant und die zweite Ableitung (bei Wegvorgaben die Beschleunigung) konstant null. In den Hyperbelstücken sind diese Ableitungen ungleich null und stetig. Der Übergang in der ersten Ableitung (bei Wegvorgabe die Geschwindigkeit) erfolgt stetig. Dagegen hat die zweite

Ableitung (bei Wegvorgaben die Beschleunigung) an den Übergangsstellen zwischen den Hyperbelstücken und zwischen den Geradenstücken und den Hyperbelstücken einen Sprung.
Genau dann, wenn die Summe der benachbarten Anstiege einer Stützstelle null ist, liegen die Hauptachsen der zugehörigen Hyperbel parallel zu den x- und y-Achsen. Der Scheitelpunkt des Hyperbelastes liegt genau in diesem Fall senkrecht über oder unter der Stützstelle.

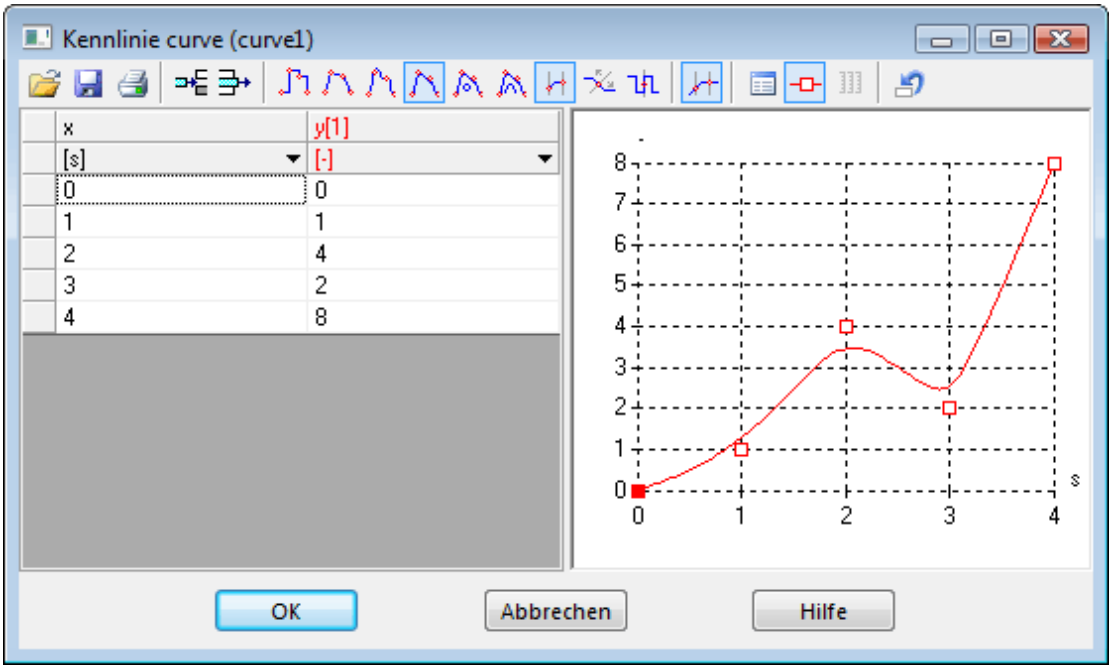



**Bild 5.25:** Kennlinie mit Hyperbolischer Approximation

## Approximation durch Kreisbögen:

Die Kurve wird an die Stützstellen durch eine Aneinanderreihung von Kreisbögen angenähert. Die zulässige Abweichung wird dabei durch den Parameter Approximations-Toleranz bestimmt.
Die Kreisbogenstücke gehen stetig und stetig differenzierbar in die benachbarten Kreisbogenstücke oder in die Verbindungsgeraden über. Der Übergang erfolgt links und rechts mit dem gleichen Abstand vom Datenpunkt. Dieser Abstand ergibt sich aus der Approximations-Toleranz und den Abständen der beiden dem aktuellen Datenpunkt benachbarten Datenpunkte vom aktuellen Datenpunkt. Er ist höchstens die Hälfte des kleineren der beiden Abstände. In den Randintervallen wird von der Randstützstelle bis mindestens zum Intervallmittelpunkt bei nicht periodischer Fortsetzung auf jeden Fall ein Stück der Verbindungsgeraden verwendet.

Die ***Approximations-Toleranz*** (im abgebildeten Kurvenbeispiel ist ein Wert von 1000 eingestellt) ist der senkrechte Abstand des Datenpunktes vom Kreisbogen-stück. Der Abstand des Datenpunktes vom Mittelpunkt des zugehörigen Kreises ist somit die Summe aus Kreisradius und der Approximations-Toleranz. Allerdings gibt es einen Maximalabstand, damit die Berührungspunkte des Kreises nicht weiter als die halbe Streckenlänge vom nächsten Datenpunkt entfernt liegen.

Deshalb führt eine weitere Vergrößerung nicht zu einer Änderung der Kurve.
Wenn Stücke der Verbindungsgeraden verwendet werden, ist die erste Ableitung (bei Wegvorgabe die Geschwindigkeit) konstant und die zweite Ableitung (bei Wegvorgaben die Beschleunigung) konstant null. In den Kreisbogenstücken sind diese Ableitungen ungleich null und stetig. Der Übergang in der ersten Ableitung (bei Wegvorgabe die Geschwindigkeit) erfolgt stetig. Dagegen hat die zweite Ableitung (bei Wegvorgaben die Beschleunigung) an den Übergangsstellen zwischen den Kreisbogenstücken und zwischen den Geradenstücken und den Kreisbogenstücken einen Sprung.
Genau dann, wenn die Summe der benachbarten Anstiege einer Stützstelle null ist, liegt der Kreismittelpunkt senkrecht über oder unter der Stützstelle. Genau in diesem Fall haben die Übergangsstellen genau den gleichen Abstand von der Stützstelle (auf der x-Achse).

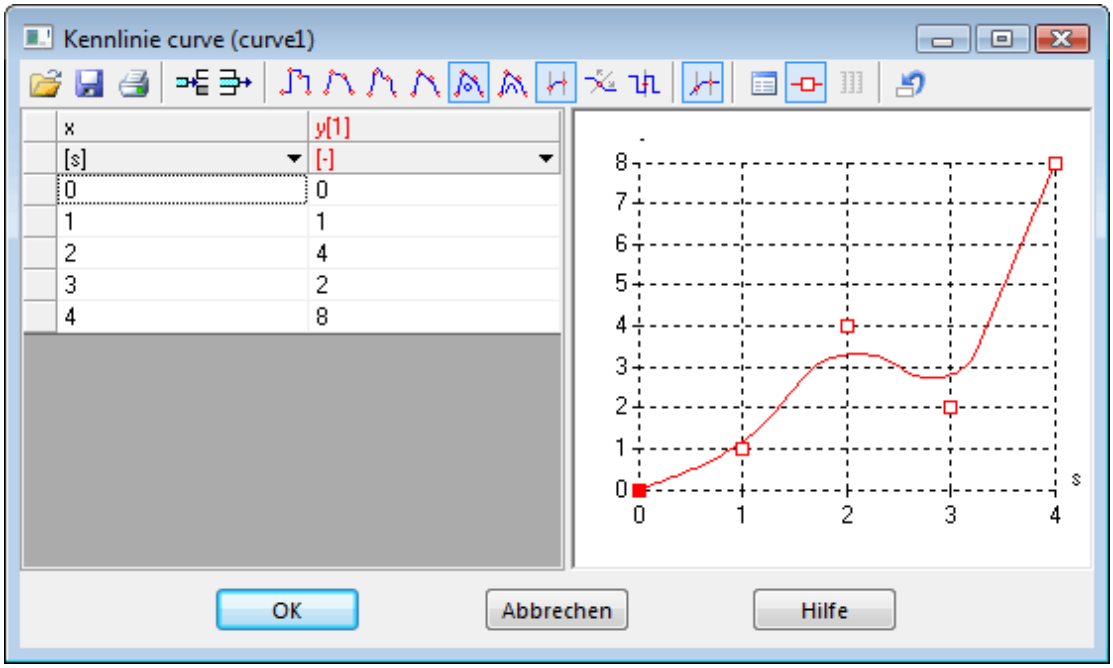



**Bild 5.26:** Kennlinie mit Approximation durch Kreisbögen

Quadratische Approximation:

Die Kurve wird an die Stützstellen durch eine Aneinanderreihung von quadratischen Polynomen angenähert. Die zulässige Abweichung wird dabei durch den Parameter Approximations-Toleranz bestimmt. Die Parabelstücke gehen stetig und stetig differenzierbar in die benachbarten Parabelstücke oder in die Verbindungsgeraden über. Der Übergang erfolgt links und rechts mit dem gleichen Abstand von der Stützstelle auf der x-Achse. Dieser Abstand ist höchstens die Hälfte der kleineren Intervalllänge der beiden Intervalle, die sich in der Stützstelle berühren. Beim längeren Intervall geht somit das Parabelstück in die Verbindungsgerade über, bevor das nächste Parabelstück erreicht wird. In den Randintervallen wird von der Randstützstelle bis mindestens zum Intervallmittelpunkt bei nicht periodischer Fortsetzung ebenfalls ein Stück der Verbindungsgeraden verwendet.

Die ***Approximations-Toleranz*** (in dem abgebildeten Kurvenbeispiel ist ein Wert von 1000 eingestellt) ist der Abstand der Übergangsstelle, der Berührungspunkte der Parabel und der Verbindungsgeraden, von der Stützstelle auf der x-Achse. Allerdings gibt es einen Maximalabstand, damit die Berührungspunkte der Parabel nicht weiter als die halbe Streckenlänge vom nächsten Datenpunkt entfernt liegen. Deshalb führt eine weitere Vergrößerung nicht zu einer Änderung der Kurve.
Wenn Stücke der Verbindungsgeraden verwendet werden, ist die erste Ableitung (bei Wegvorgabe die Geschwindigkeit) konstant und die zweite Ableitung (bei Wegvorgaben die Beschleunigung) konstant null. In den Parabelstücken ist die erste Ableitung linear und die zweite Ableitung konstant. Der Übergang in der ersten Ableitung (bei Wegvorgabe die Geschwindigkeit) erfolgt stetig. Dagegen hat die zweite Ableitung (bei Wegvorgaben die Beschleunigung) an den Übergangsstellen zwischen den Parabelstücken und zwischen den Geradenstücken und den Parabelstücken einen Sprung.
Genau dann, wenn die Summe der benachbarten Anstiege einer Stützstelle null ist, liegt der Scheitelpunkt und der Brennpunkt der Parabel senkrecht über oder unter der Stützstelle.

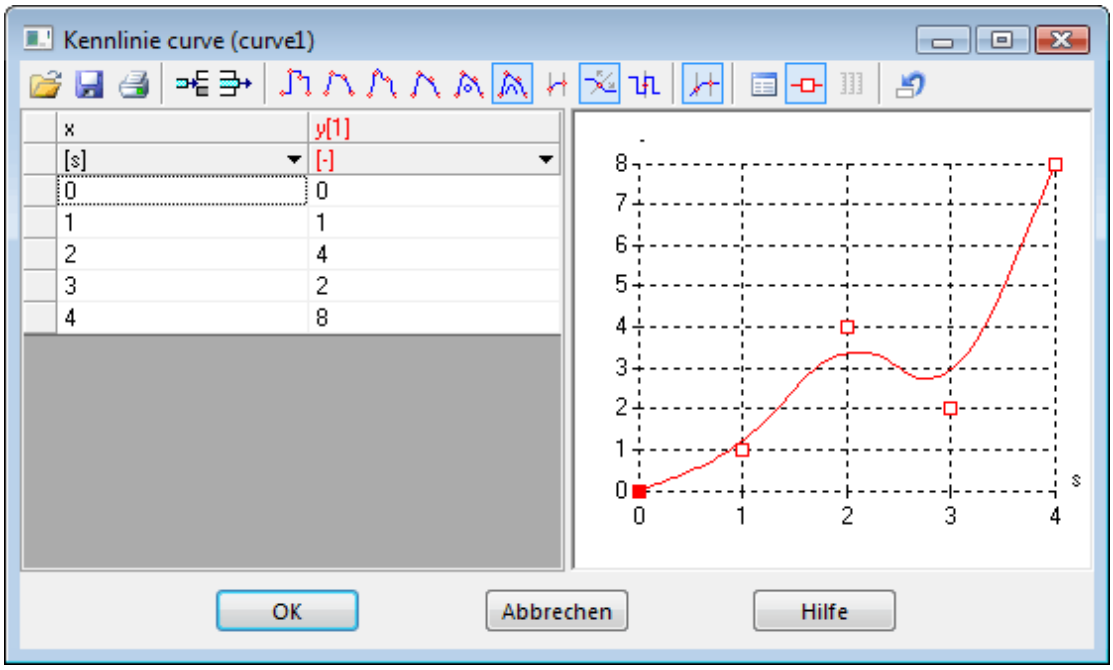



**Bild 5.27:** Kennlinie mit Quadratischer Approximation

- ***Fortsetzung der Kurve außerhalb des spezifizierten Intervalls (Definitionsbereichs):***

Die Kennlinie bietet eine Reihe von Möglichkeiten, die beschriebene Kurve über den angegebenen Bereich hinaus fortzusetzen. Dabei stehen alternativ die Auswahlmöglichkeiten *Keine Sonderbehandlung für Werte außerhalb des eingegebenen Bereichs*, *Fortsetzung durch Spiegelung* und *Periodische Fortsetzung*. Für die ersten beiden Möglichkeiten kann zusätzlich *Lineare Extrapolation* eingeschaltet werden.

Lineare Extrapolation: Diese Schaltfläche beeinflusst das Verhalten in den Darstellungsarten *Keine Sonderbehandlung für Werte außerhalb des eingegebenen Bereichs* und *Fortsetzung durch Spiegelung*. Ist die lineare Extrapolation eingeschaltet, so wird die Kurve über den angegebenen Bereich hinaus mit dem Anstieg im jeweiligen Grenzpunkt linear fortgesetzt. Bei ausgeschalteter linearer Extrapolation bricht die Simulation beim Überschreiten des spezifizierten Kennlinienbereichs mit einer Fehlermeldung ab. Die Einstellung ist bei der Kurve mit *periodischer Fortsetzung* unwirksam. Nachfolgende Abbildung illustriert die lineare Extrapolation:

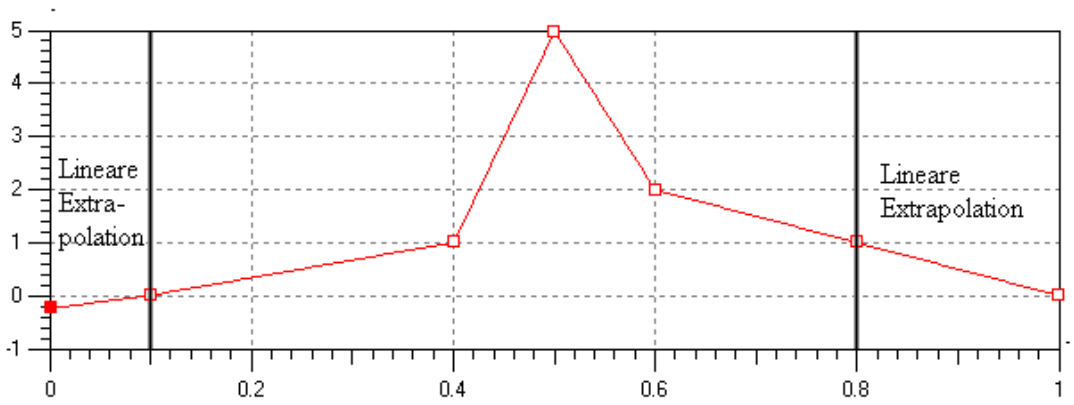



**Bild 5.28:** Lineare Extrapolation

Keine Sonderbehandlung für Werte außerhalb des eingegebenen Bereichs: Es erfolgt weder eine Spiegelung noch eine periodische Fortsetzung der Kennlinie. Jedoch wird bei eingeschalteter linearer Extrapolation die Kurve an den Rändern mit dem Anstieg im Grenzpunkt linear fortgesetzt.

Fortsetzung durch Spiegelung: Die Kurve wird bezüglich des linken Grenzpunktes gespiegelt (beide Koordinatenrichtungen). Außerhalb des definierten Bereichs ergibt sich das Verhalten gemäß der Einstellung für die Interpolation. Die Fortsetzung durch Spiegelung ist in nachfolgender Abbildung zusammen mit linearer Extrapolation demonstriert:

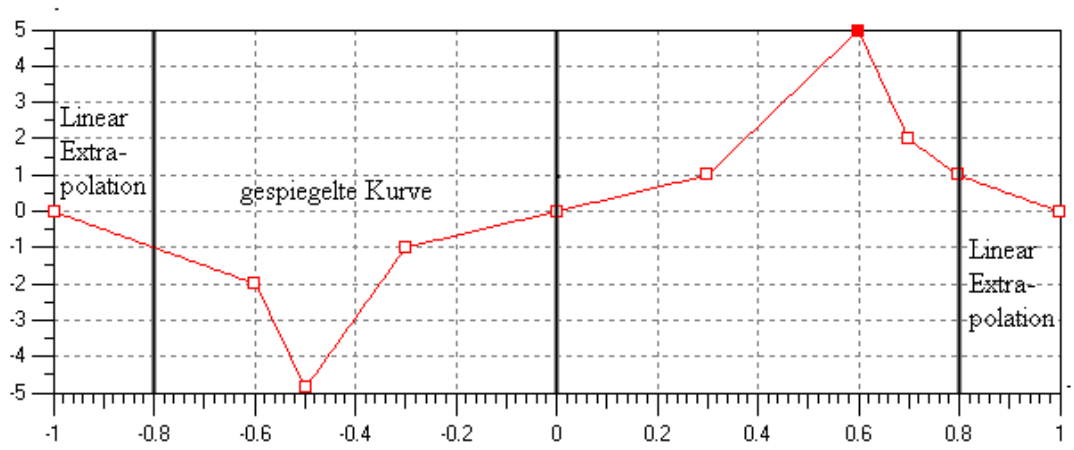



**Bild 5.29:** Fortsetzung der Kurve durch Spiegelung

Periodische Fortsetzung: Außerhalb des spezifizierten Bereichs wird die Kurve periodisch fortgesetzt. Der rechte Randpunkt wird dabei so angepasst, dass sich eine stetige Fortsetzung ergibt - er nimmt den Funktionswert des linken Randpunktes an. Bei periodischer Fortsetzung kann keine Einstellung für die *lineare Extrapolation* vorgenommen werden. Es ergibt sich beispielhaft der nachfolgend gezeigte Verlauf:

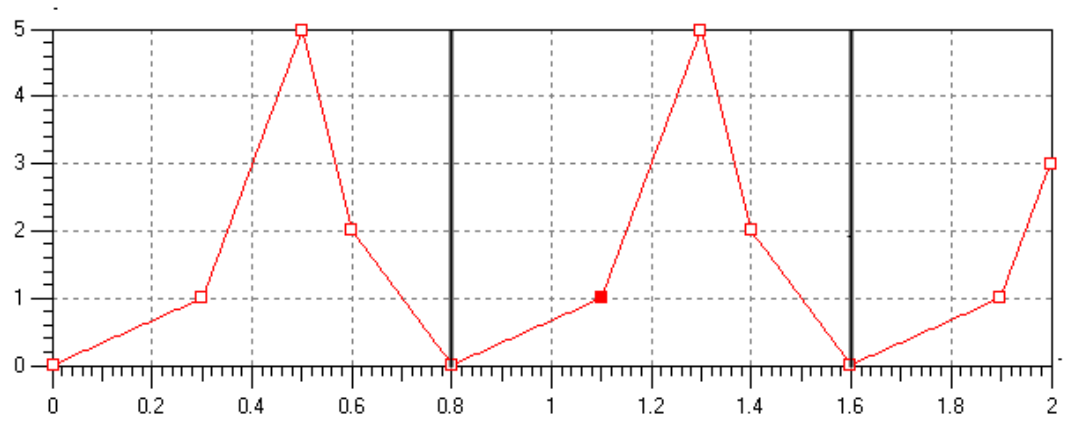



**Bild 5.30:** Periodische Fortsetzung der Kurve

- ***Eigenschaftsdialog für Kennlinien:***

Der Eigenschaftsdialog für die Kennlinien wird durch Betätigung der entsprechenden Schaltfläche auf dem Toolbar des Kennliniendialogs (Kennlinien, Kennliniensätzen, Kurvenscharen und Hysterese-Kurven) geöffnet.

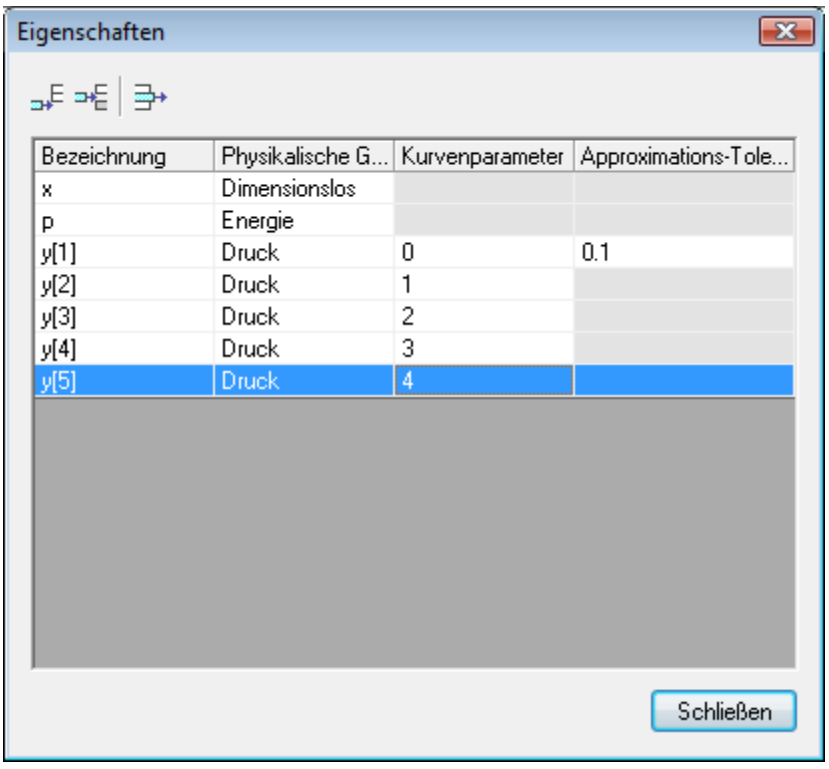



**Bild 5.31:** Eigenschaftsdialog für Kennlinien

Im Eigenschaftsdialog werden in einer Tabelle alle verfügbaren Achsen aufgeführt. Zuerst wird die X-Achse angezeigt, gefolgt von einer oder mehreren Y-Achsen (abhängig von der Kennlinienart).

Funktionsübersicht im Eigenschaftsdialog:

| Funktion | Kennlinie | Kenn-liniensatz | Kurvenschar | Hysterese-Kurve |
|---|---|---|---|---|
| Ändern der Bezeichnung einer Achse: | ja | ja | ja | ja |
| Einfügen einer Kurve vor der ausgewählten ( ): | nein | ja | ja | nein |
| Einfügen einer Kurve am Ende der Tabelle ( ): | nein | ja | ja | nein |
| Löschen der ausgewählen Kurve ( ): | nein | ja | ja | ja |
| Auswahl physikalischer Größen: | ja, für X- und Y-Achse | für X- und für alle Y-Achsen | Für Kurven-parameter, X- und für alle Y-Achsen gemeinsam | für X- und für alle Y-Achsen gemeinsam |
| Eingabe von Kurven-parametern: | nein | nein | ja | ja |
| Eingabe der Approximations-Toleranz: | ja, wenn Approxi-mation eingestelllt | ja, wenn Approxi-mation eingestelllt | ja, wenn Approxi-mation eingestelllt | ja, wenn Approxi-mation eingestelllt |

***Ändern der Bezeichnung für eine Achse***

Die Bezeichnung einer Achse ändert man, indem die entsprechende Zelle der Tabelle doppelt angeklickt wird. Anschließend ist die Windows-Editierfunktion für die Änderung verfügbar. Standardmäßig werden die Achsen mit y[i] bezeichnet, wobei i für die fortlaufende Nummerierung der Y-Achsen steht (alle Y-Kurven benötigen diesen zusätzlichen Parameter). In der Standardeinstellung werden die Achsen in der Reihenfolge ihrer Erzeugung nummeriert.

***Einfügen einer Kurve vor der ausgewählten Zeile***

Um eine neue Kurve vor einer bestimmten Zeile einzufügen, wird diese durch Anklicken eines ihrer Felder ausgewählt. Nach dem Drücken von erfolgt ein neuer Eintrag in der Tabelle, wobei die fortlaufende Nummerierung beibehalten wird. Diese Funktion ist für Kennliniensätze und Kurvenscharen verfügbar.

### *Einfügen einer Kurve am Tabellenende*

Durch Drücken von wird am Ende der Tabelle eine neue Zeile eingefügt und entsprechend nummeriert 'y[i]'.
Diese Funktion ist für Kennliniensätze und Kurvenscharen verfügbar.

### *Löschen einer Kurve*

Um eine Kurve aus der Tabelle zu Löschen wird diese angeklickt und anschließend betätigt. Die folgenden Kurven werden automatisch neu nummeriert.

### *Rücksetzen auf Standardwerte*

Die Werte der gesamten Kennlinie werden auf Standardwerte zurückgesetzt.

### *Auswahl physikalischer Größen*

Ein Anklicken eines Eintrages in der Spalte "Physikalische Größen" öffnet ein Auswahlmenü, welches die Festlegung einer physikalischen Größe für eine Achse der entsprechenden Kurve erlaubt. Das Einstellen der Einheiten für die Eingabe der Werte erfolgt im Kennliniendialog.
Beachten Sie, dass für Kurvenscharen und Hysteresekurven nur eine gemeinsame physikalische Größe für alle Y-Achsen gewählt werden kann!

### *Setzen von Kurvenparametern*

Mittels eines zusätzlichen Parameters, welcher typischerweise über einen Signaleingang bereitgestellt wird, kann die Auswahl der einzelnen Kurven bei Kurvenscharen erfolgen. Für Zwischenwerte der Kurvenparameter erfolgt eine lineare Interpolation zwischen den benachbarten Kurven.

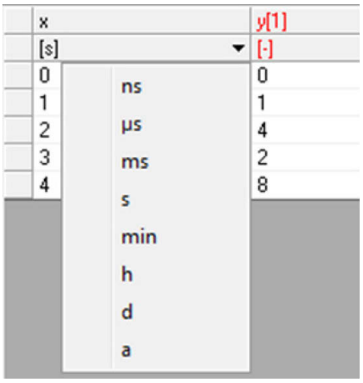
---

**Hinweis:** Es ist immer darauf zu achten, dass die Werte in der Reihenfolge der Kurven streng monoton anwachsen, ansonsten kann der Eigenschaftsdialog nicht geschlossen werden!

---

Der **Eigenschaftsdialog** für die Kennliniendarstellung wird durch Betätigung der entsprechenden Schaltfläche auf dem **Toolbar** des Kennliniendialogs geöffnet:

### *Auswahl physikalischer Einheiten:*




Für die x- und y-Achse können Maßeinheiten gemäß der im **Eigenschaftsdialog für *Kennlinien*** ausgewählten physikalischen Größe festgelegt werden.

Dabei erfolgt eine automatische Umrechnung der eingegebenen Werte in die jeweils ausgewählte Einheit. Die Auswahl erfolgt im Kennliniendialog durch einen Klick mit der linken Maustaste auf die jeweilige Spalte in der Titelzeile der Wertetabelle, worauf sich ein entsprechendes Auswahlmenü mit den zur Verfügung stehenden Einheiten öffnet (siehe Abbildung).
Soll eine Umrechnung der Zahlenwerte in die neue Einheit vermieden werden, so muss bei der Einheiten-Auswahl die *Shift*-Taste gedrückt gehalten werden.
Die ausgewählte Einheit für die Eingangsgröße bestimmt ausschließlich die Umrechnung in der Wertetabelle und der graphischen Darstellung. Das Eingangssignal selbst muss in der entsprechenden SI-Basiseinheit bereitgestellt werden (dies geschieht automatisch bei Verwendung von einheitenbehafteten Signalen).
Die Einheit für die Ausgangsgröße wird für die Ergebnisdarstellung übernommen. Das Ausgangsignal selbst wird in der jeweiligen SI-Basiseinheit bereitgestellt.

### 5.4.3.2 Kennliniensatz

Der Kennliniensatz erzeugt aus einem vorgegebenen Eingangssignal einen Vektor von Ausgangssignalen, wovon jedes einzeln parametrierbar ist.
Die Parametrierung ist weitgehend identisch zur Parametrierung von Kennlinien. Folgende Unterschiede bestehen:

Der Eigenschaftsdialog erlaubt das Hinzufügen und Löschen von Kurven.

Die Auswahl von physikalischen Größen und Einheiten ist der Verwendung mehrerer Kurven angepasst.

Interpolation und Approximation sind an die Verwendung mehrerer Kurven angepasst.

- ***Kennliniensatz-Dialog***

Sind mehrere Kennlinien vorhanden, öffnet sich im Dialog eine Tabelle (1. Spalte repräsentiert die X-Werte, 2. bis n-te Spalte repräsentieren die Y-Werte der einzelnen Kennlinien). Die Tabelle kann, wie unter **5.4.3.1** beschrieben, bearbeitet werden.

Im Toolbar gibt es zusätzlich ein Element

*Alle Achsen anzeigen* (separat für jede Y-Achse)

### 5.4.3.3 Kurvenschar

Eine Kurvenschar ermöglicht die Modellierung eines Kennfeldes durch Vorgabe einer Reihe von Kurven z=f(x), deren Auswahl durch ein Eingangssignal y gesteuert wird. Eine vergleichbare Funktionalität ist auch über 2D-Kennfelder nutzbar.

Die Parametrierung entspricht im Wesentlichen der **Kennlinie** und es soll nur auf die folgenden Besonderheiten eingegangen werden:

Kurvenscharen werden immer als Funktion des Eingangssignals x berechnet und das Eingangssignal y liefert den Parameter zur Auswahl der Kurve aus der Kurvenschar.

Der Eigenschaftsdialog erlaubt das Hinzufügen und Löschen von Kurven.

Die Auswahl von physikalischen Größen und Einheiten ist der Verwendung mehrerer Kurven angepasst.

Interpolation und Approximation sind an die Verwendung mehrerer Kurven angepasst. Für jede einzelne Kurve kann die Interpolationsart separat festgelegt werden. Dazu wählt man einen zu der jeweiligen Kurve gehörenden Tabelleneintrag und wählt die gewünschte Interpolationsart durch die entsprechende Schaltfläche. **Entlang der y-Achse wird zwischen den einzelnen Kurven immer linear interpoliert.**

**Hinweis:** Es ist immer darauf zu achten, dass die Werte für den Kurvenparameter in der Reihenfolge der Kurven streng monoton anwachsen.

### 5.4.3.4 2D-Kennfeld

Mittels des 2D-Kennfeldes besteht die Möglichkeit, eine Parametrierung mit 2D-Feldern (2-parametrige Kurven) vorzunehmen.

- ***Kennfelddialog:***

Der Kennfelddialog ermöglicht das komfortable Einstellen und Editieren von:

- Stützstellen (als Tabelle oder grafisch interaktiv)
- Interpolation/Approximation
- Maßeinheiten

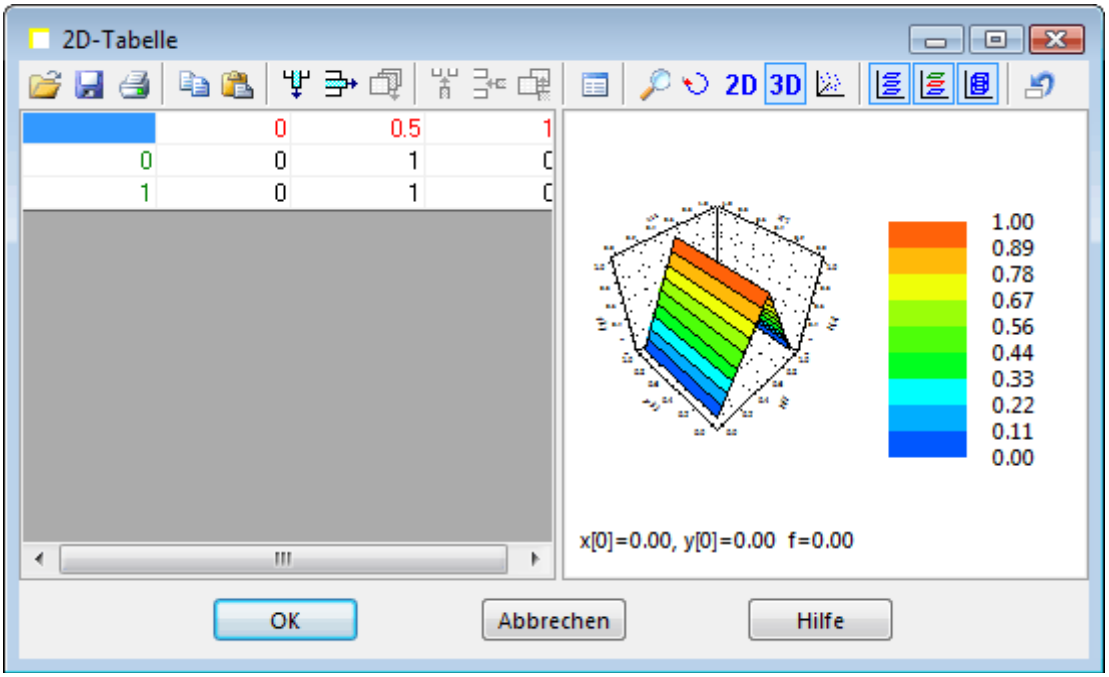



**Bild 5.32:** 2D-Kennfelddialog

- ***Toolbar:***

Der ***Toolbar*** im oberen Bereich des Kennfelddialogs bietet Zugriff auf folgende Funktionen:

*Laden* von Kennfelddatensätzen

*Speichern* des eingegebenen Datensatzes

*Drucken* des Kennfeldes

Aktuelle Auswahl *kopieren*

*Einfügen* in aktuelle Auswahl

Aktuelle Spalte *löschen*

Aktuelle Zeile *löschen*

Neue Spalte an aktueller Position *einfügen*

Neue Zeile an aktueller Position *einfügen*

*Eigenschaftsdialog* der Kennlinie

Rücksetzen auf Standardwerte

*Zoomfunktion* für die grafische Darstellung

*Rotationsfunktion* für die grafische Darstellung

*Zweidimensionale grafische Darstellung* der Kurve (Draufsicht, Konturdiagramm)

*Dreidimensionale grafische Darstellung*

*Punktweise Darstellung* der Stützstellen

*Höhenlinien anzeigen*

*Zonenbereich anzeigen*

*Gitterstruktur anzeigen*

- ***Kennfeldeingabe, Datenimport:***

Für die Eingabe des Kennfeldes existieren mehrere Möglichkeiten, Datenpunkte zu erzeugen bzw. zu verändern:

*Eintragen/Ändern* von Wertepaaren in der Tabelle

*Laden* eines Kennfeldes aus einer Datei (unterstützte Formate: Text (*.txt), Microsoft Excel-Arbeitsmappe(*.xls).

Die Anzeige der Kennfelder erfolgt grafisch und in Tabellenform. In der Tabelle beschreiben die Einträge in der 1. Zeile (rot) die Datenpunkte der x-Achse, die Einträge der 1. Spalte (grün) die Position der Datenpunkte bezüglich der y-Achse.

In den verbleibenden Zellen werden die zu den durch Zeile und Spalte festgelegten x- bzw. y-Koordinaten gehörigen Funktionswerte eingetragen.

In der grafischen Darstellung sind die zur jeweiligen Mausposition gehörigen x-, y- und Funktionswerte sichtbar.

Für das Editieren des Kennfeldes in der Tabelle stehen folgende Tastenbelegungen zur Verfügung:

| | |
|---|---|
| *Tab*-Taste | bewegt den Cursor zur nächsten Zelle |
| *Shift+Tab* | bewegt den Cursor zur vorherigen Zelle |
| *Cursortasten auf/ab* | bewegen den Cursor innerhalb einer Spalte nach oben und unten |
| *Cursortasten links/rechts* | bewegen den Cursor innerhalb einer Zeile nach links und rechts |

Zum Markieren von einzelnen Feldern der Tabelle können die in Windows üblichen Markierfunktionen angewandt werden, nachdem ein Tabellenfeld per Mausklick markiert wurde:

| | |
|---|---|
| *Linke Maustaste+Shift* | Markieren des rechteckigen Tabellenbereiches zwischen dem zuletzt selektierten Tabellenfeld und dem angeklickten Feld |
| *Linke Maustaste+Strg* | Hinzufügen des angeklickten Feldes zur aktuellen Auswahl |
| *Cursortasten+Shift* | Markieren des bei der Cursorbewegung überstrichenen rechteckigen Tabellenbereiches |

- ***Datenexport:***

Kennfelder können als Text (*.txt) und Microsoft Excel-Arbeitsmappe (*.xls) gespeichert( ) werden und auch gedruckt( ) werden.

- ***Kopieren und Einfügen von Tabellenfeldern:***

Über die Schaltfläche können markierte Tabellenfelder in die Zwischenablage kopiert werden. Nach Markieren einer neuen Position in der Tabelle werden durch Betätigen der Schaltfläche die zwischengespeicherten Werte an der markierten Stelle eingefügt. Dabei erhält das selektierte Feld den linken oberen Wert des in die Ablage kopierten Bereiches.

- ***Anpassen der Tabellengröße:***

Die Größe der Tabelle und damit die Zahl der Punkte in der Kurve kann über die Schaltflächen (*aktuelle Spalte löschen*) und (*aktuelle Zeile löschen*)

verringert werden. Dabei wird jeweils die Zeile bzw. Spalte des aktuell selektierten Tabellenfeldes gelöscht. In gleicher Weise erlauben die Schaltflächen (*Spalte einfügen*) und (*Zeile einfügen*) eine Erweiterung der Tabelle. Die neuen Spalten bzw. Zeilen werden dabei links bzw. oberhalb des aktuell markierten Tabellenfeldes eingefügt. Die neuen x- bzw. y-Werte werden so gewählt, dass der neue Wert genau zwischen seinen beiden Nachbarwerten liegt. Bei Einfügen von Zeilen oberhalb der ersten Zeile bzw. links der ersten Spalte wir der Wert so eingestellt, dass die markierte Zeile/Spalte einen Wert in der Mitte zwischen der neuen Zeile/Spalte und ihrer Nachbar-Zeile/Spalte annimmt.

- ***Anpassen der grafischen Darstellung:***

Für das Anpassen der grafischen Darstellung steht eine Reihe von Schaltflächen im Toolbar zur Verfügung:

*Zoomfunktion*: Die Zoomfunktion erlaubt die Vergrößerung eines Bereiches der Kurvendarstellung. Dazu wird die Schaltfläche gedrückt und danach mit der Maus ein rechteckiger Bereich der Darstellung selektiert. Diese Operation kann durch erneutes Betätigen der Schaltfläche und einen Klick mit der rechten Maustaste auf die Darstellung rückgängig gemacht werden. Ein Doppelklick auf die Grafik stellt die ursprüngliche Ansicht vor allen Zoom- oder Rotationsschritten wieder her.

*Rotation der grafischen Darstellung*: Mittels dieser Schaltfläche kann die Kurvenansicht frei in alle Richtungen rotiert werden. Dafür wird die Schaltfläche betätigt und anschließend mit gedrückter linker Maustaste (über der grafischen Darstellung) der Betrachtungswinkel verändert. Die Operation wird durch erneutes Betätigen der Schaltfläche und einen Klick mit der rechten Maustaste auf die Grafik rückgängig gemacht. Ein Doppelklick auf die Grafik stellt die ursprüngliche Ansicht vor allen Zoom- oder Rotations-Schritten wieder her.

*2D-Ansicht*: Diese Darstellungsart zeigt die Kurve in der Draufsicht (Konturdiagramm). Je nach Auswahl werden Datenpunkte bzw. Höhenlinien, Zonenbereiche und/oder ein Gitternetz angezeigt.

*3D-Ansicht*: Diese Darstellungsart zeigt die Kurve in perspektivischer Ansicht. Je nach Auswahl werden Datenpunkte bzw. Höhenlinien, Zonenbereiche und/oder ein Gitternetz angezeigt.

*Punktweise Darstellung der Stützstellen*: In dieser Darstellung werden nur die in der Tabelle gegebenen Stützstellen der Kurve angezeigt. Es erfolgt keine Anzeige von Höhenlinien, Zonenbereichen und Gitternetz.

*Höhenlinien anzeigen*: Diese Schaltfläche aktiviert/deaktiviert die Anzeige von Höhenlinien in der grafischen Darstellung. Bei der punktweisen Darstellung der Stützstellen Punkte steht diese Funktion nicht zur Verfügung.

*Zonenbereiche anzeigen*: Diese Schaltfläche aktiviert/deaktiviert die farbige Markierung der Höhenzonen in der grafischen Darstellung. Eine Höhenzone entspricht dabei genau dem Bereich zwischen 2 Höhenlinien. Bei der punktweisen Darstellung der Stützstellen Punkte steht diese Funktion nicht zur Verfügung.

*Gitterstruktur anzeigen*: Diese Darstellung aktiviert/deaktiviert die Verbindung der eingegebenen Stützstellen mit einem Gitternetz. Bei der punktweisen Darstellung der Stützstellen Punkte steht diese Funktion nicht zur Verfügung.

- ***Eigenschaftsdialog für 2D-Kennfelder*:**

Der über die entsprechende Schaltfläche im Kennfelddialog aktivierbare Eigenschaftsdialog erlaubt die Einstellung von

- Kommentartext,
- physikalischer Größe,
- Maßeinheit und
- Interpolations- bzw. Approximationsart (bei letzterem mit Approximationstoleranz)

sowohl für die Eingangsgrößen im Kennfeld als auch für den Funktionswert. Im Eigenschaftsdialog steht für jede der Größen eine separate Registrierkarte zur Verfügung.

Kommentartext:

Der Kommentartext dient der Beschriftung der einzelnen Achsen in der grafischen Darstellung des Kennfeldes. Er kann z. B. Formelzeichen oder kurze beschreibende Texte enthalten.

Auswahl physikalischer Größen und Einheiten:

Den Eingangsgrößen und dem Funktionswert können über das Auswahlmenü auf der zugehörigen Registerkarte physikalische Größen zugewiesen werden. In der Folge sind die jeweiligen Werte mit Einheiten behaftet und durch die Auswahlbox für die Einheiten kann eine passende Einstellung vorgenommen werden. Die Einheiten beeinflussen die Zahlenwerte in der tabellarischen Vorgabe der Stützstellen sowie in der grafischen Darstellung des Kennfeldes. Ein Wechsel der Einheit bedingt eine Umrechnung in der Tabelle und der Grafik. Darüber hinaus gilt die gewählte physikalische Größe und Einheit beim Wertebereich ebenso für die Darstellung der Ergebnisse.

Es ist jedoch zu beachten, dass die Signalwerte an den Signaleingängen immer in der SI-Basiseinheit der betreffenden Größe bereitgestellt werden müssen. Die aktuell ausgewählte Einheit für die Stützstelleneingabe beeinflusst lediglich die Darstellung des Kennfeldes, intern wird in der SI-Basiseinheit gerechnet. Ebenso beeinflusst die gewählte Einheit für die Werte des Kennfeldes nur die Kennfeld-

und Ergebnisdarstellung. Am Signalausgang wird der ermittelte Wert in der entsprechenden SI-Basiseinheit bereitgestellt.

Interpolation bzw. Approximation zwischen den Stützstellen:

Interpolation bzw. Approximation zwischen den Stützstellen können für alle Eingangsgrößen separat eingestellt werden. Es stehen die Möglichkeiten, wie Sie unter **5.4.3.1** bei Kennlinien aufgeführt sind, mit folgenden Einschränkungen:

- Lineare Interpolation (bezüglich aller Eingangsgrößen),
- Treppenfunktion (bezüglich aller Eingangsgrößen),
- Spline-Interpolation (nur bezüglich x-Achse),
- Hyperbolische Approximation (nur bezüglich x-Achse),
- Approximation durch Kreisbögen (nur bezüglich x-Achse),
- Quadratische Approximation (nur bezüglich x-Achse) zur Verfügung.

Für die Approximationsarten ist die Vorgabe einer Approximationstoleranz möglich, die ein Maß für die erlaubten Abweichungen der Approximation gegenüber den Stützstellen darstellt.

### 5.4.3.5 3D-Kennfeld

Mittels des 3D-Kennfeldes besteht die Möglichkeit, eine Parametrierung mit 3D-Feldern (3-parametrige Kurven) vorzunehmen.

- ***Kennfelddialog:***

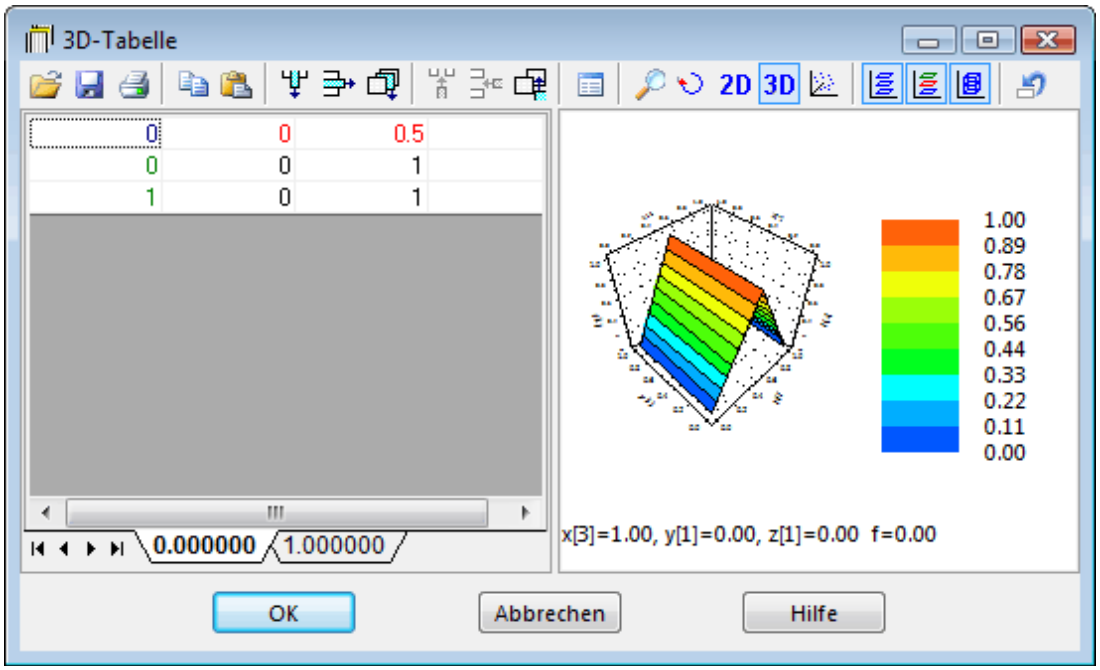



**Bild 5.33:** 3D-Kennfelddialog

Der Kennfelddialog ermöglicht das komfortable Einstellen und Editieren von:

- Stützstellen (als Tabelle),

- Interpolation/Approximation,
- Maßeinheiten.

- ***Toolbar:***

Der **Toolbar** im oberen Bereich des Kennfelddialogs bietet Zugriff auf alle im **2D-Kennfeld** (***Toolbar*** ) vorhandenen Funktionen, erweitert um:

*Aktuelle Tabelle löschen*

*Neue Tabelle an aktueller Position einfügen*

- ***Kennfeldeingabe, Datenimport und –export:***

Die Möglichkeiten und die Vorgehensweisen zur Kennfeldein- und –ausgabe entsprechen denen des **2D-Kennfeldes** (***Kennfeldeingabe, Datenimport***) und sind dort im Detail beschrieben.

Für das 3D-Kennfeld besteht darüber hinaus die Möglichkeit, die Stützstellen bezüglich der Eingangsgröße z anhand verschiedener Tabellen zu spezifizieren. Die Auswahl der jeweiligen Tabelle erfolgt durch Auswahl der zugehörigen Registerkarte am unteren Rand des Tabellenfensters. Der entsprechende z-Wert ist in der linken oberen Zelle der Tabelle angegeben und kann dort editiert werden.

Kopieren und Einfügen von Tabellenfeldern:
Die Funktionalität entspricht der des **2D-Kennfeld** und ist dort im Detail beschrieben.

Anpassen der Tabellengröße:
Neben den im **2D-Kennfeld** vorhandenen Editierfunktionen stehen zwei weitere Schaltflächen zur Verfügung, mit denen die Zahl der Stützstellen entlang der z-Achse des Kennfeldes beeinflusst wird. löscht die aktuell angezeigte Tabelle und damit alle Stützstellen für den entsprechenden z-Wert. fügt eine neue Tabelle an der aktuellen Position ein. Für das Einfügen und die Auswahl des neuen z-Wertes gelten die bereits für das ***Kopieren und Einfügen von Tabellenfeldern*** im **2D-Kennfeld** erläuterten Regeln.

Anpassen der grafischen Darstellung

Für das Anpassen der grafischen Darstellung (Zoom, Rotation, 2D/3D-Darstellung, Stützstellen, Höhenlinien, Zonenbereiche, Gitterstruktur) gilt das im **2D-Kennfeld** (***Anpassen der grafischen Darstellung***) genannte.

- ***Eigenschaftsdialog für 3D-Kennfelder*:**

Analog zum **Eigenschaftsdialog für *Kennlinien*** erlaubt der Eigenschafts-dialog die Festlegung von Kommentartexten, physikalischen Größen und Maß-einheiten, sowie der Interpolation bzw. Approximation bei der Berechnung des Ausgangssignals.

## 5.4.4 3D-Ansicht

Elemente der Mehrkörpermechanik können in einer oder mehreren 3D-Ansichten visualisiert werden. Hier lassen sich der Simulationsaufbau, die Orientierung und Ausdehnung der MKS-Elemente, als auch der Simulationsverlauf anschaulich darstellen. Über den Menüpunkt ***Fenster/Neue 3D-Ansicht*** lässt sich eine neue 3D-Ansicht für das aktuelle Modell öffnen. Neben der Visualisierung dient die 3D-Ansicht auch als interaktive Oberfläche für die Modellierung von MKS-Modellen.

### 5.4.4.1 Elemente der 3D-Ansicht

In der 3D-Ansicht sind neben den eigentlichen Elementen weitere Objekte (siehe **Bild 5.34**) sichtbar, welche einerseits der besseren Orientierung dienen, andererseits aber auch für die Interaktion verwendet werden.

Ein schwarzes Koordinatensystem (1) in der Mitte des Bildschirms visualisiert den Ursprung des aktuellen Modells. Anhand eines farbigen Koordinatensystems (2) links unten in der 3D-Ansicht lässt sich die Kameraorientierung ablesen. Letzteres befindet sich immer an derselben Stelle in der 3D-Ansicht, unabhängig von der Kameraposition.

MBS-Konnektoren sind an ihrem Koordinatensystem sowie Symbol erkennbar. Die Symbole entsprechen der Darstellung in der Strukturansicht, Eingangs-Konnektoren (3) werden durch ein weißes Quadrat mit gelbem Rand, Ausgangs-Konnektoren (4) durch ein gelbes Quadrat dargestellt.

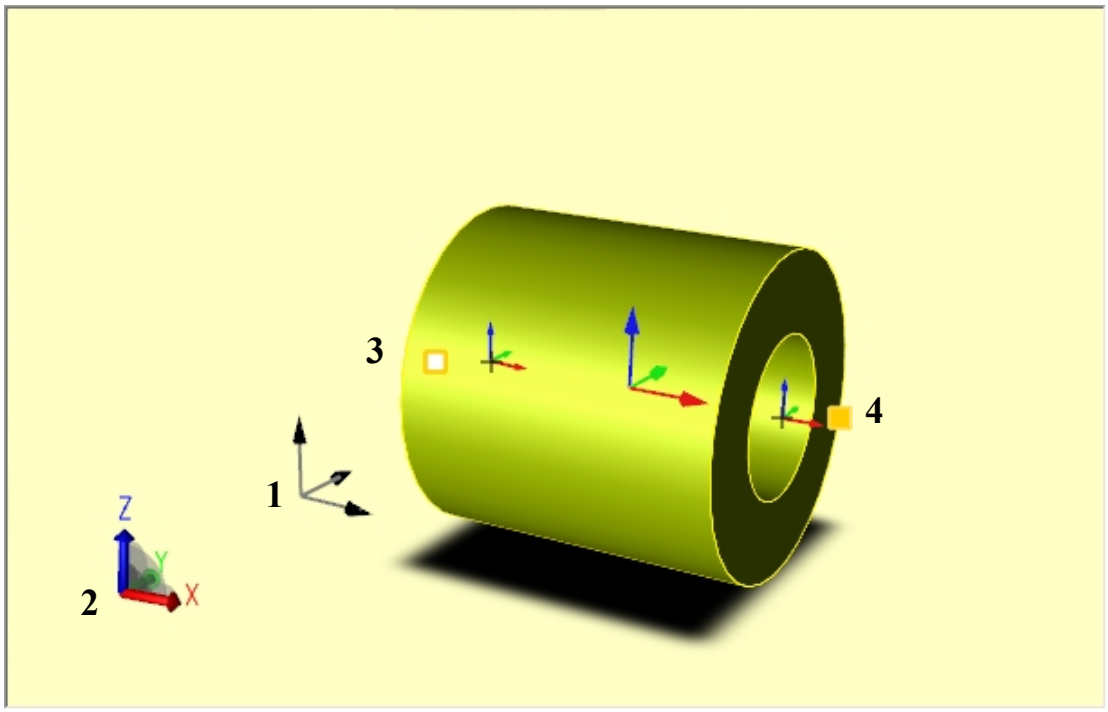
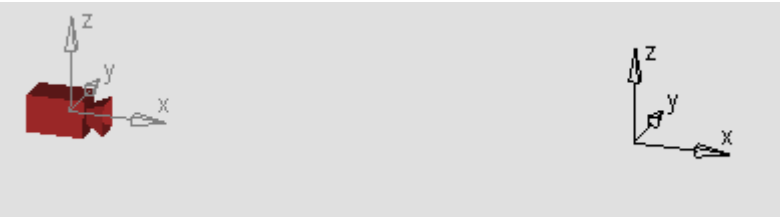



**Bild 5.34:** 3D Ansicht eines Modells

**Hinweis:** Die meisten interaktiven Funktionen stehen nur in der Modellierungsphase zur Verfügung. Daher sind die Konnektorsymbole nicht sichtbar, wenn sich das Modell im berechneten Zustand befindet.

### 5.4.4.2 Kameramanipulation

Die Kamera der 3D-Ansicht ist definiert durch eine Position und einen Punkt, auf welchen die Kamera ausgerichtet ist (Zielpunkt). Die Linie zwischen Kameraposition und Kamerazielpunkt wird als Sichtvektor bezeichnet.




Die Kamerasteuerung wird mit der Maus vorgenommen:

| | |
|---|---|
| *Mittlere Maustaste* | Dreht die Kamera um den Zielpunkt |
| *Shift* + *mittlere Maustaste* | Verschiebt die Position (und den Zielpunkt) |
| *Strg* + *mittlere Maustaste* | Verändert den Abstand zwischen Kameraposition und Zielpunkt |
| Taste *C* oder | Der Zielpunkt wird in das Zentrum des aktuell ausgewählten Elements gelegt. Die Anwendung dieser Funktion auf ein bereits ausgerichtetes Element verändert den Abstand der Kamera derart, dass das Element im gesamten Bildausschnitt zu sehen ist. |
| | Vorderansicht |
| | Rückansicht |
| | Linke Ansicht |
| | Rechte Ansicht |
| | Draufsicht |
| | Untersicht |
| | Isometrische Ansicht |

Diese Ansichten verändern weder das Ziel noch den Abstand zwischen Position und Zielpunkt, rotieren die Kamera lediglich so, dass der aktuelle Bildausschnitt aus der gewählten Ansichtsrichtung dargestellt wird.

Neben den verschiedenen Ansichten stehen in der Werkzeugleiste weitere Funktionen zur Verfügung:

Kamera verschieben

Kamera drehen

Kamera entlang einer Achse rotieren

X-Achse

Y-Achse

Z-Achse

Die Achse, um die die Kamera mit diesen Werkzeugen gedreht wird, entspricht der jeweiligen aktuellen Ansichtsachse, verläuft allerdings um den gegenwärtigen Zielpunkt.

gesamtes Modell Anzeigen : Ausrichtung der Kamera, um das gesamte Modell anzuzeigen.

Zoom : Auswählen des gewünschten Bildausschnitts durch Aufziehen eines Rechtecks.

---

**Hinweis:** Die meisten Kamerafunktionen stehen nicht zur Verfügung, wenn eine Modellkamera für die aktuelle Ansicht ausgewählt ist. Hier sind die Kameraeigenschaften abhängig von den Modellparametern.

---

### 5.4.4.3 Einstellungen

Für die Darstellung in der 3D-Ansicht können verschiedene Einstellungen vorgenommen werden. Unterschieden wird dabei zwischen **globalen** Einstellungen, welche für alle Elemente innerhalb einer Ansicht gelten, und **lokale** Einstellungen, welche nur für ein Element, aber in allen Ansichten gelten. Änderungen können im Eigenschaftsdialog der Elemente bzw. der jeweiligen Ansicht vorgenommen werden.

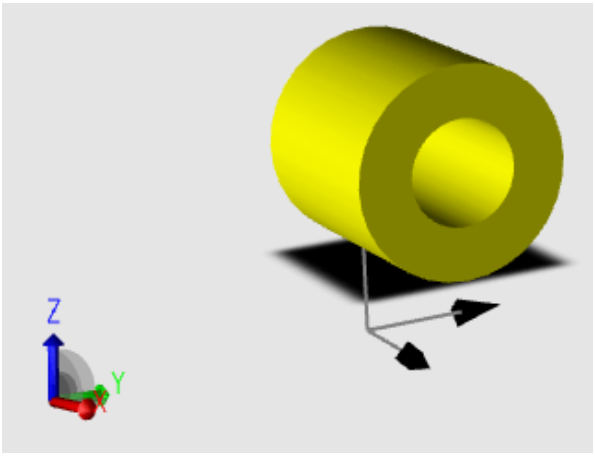
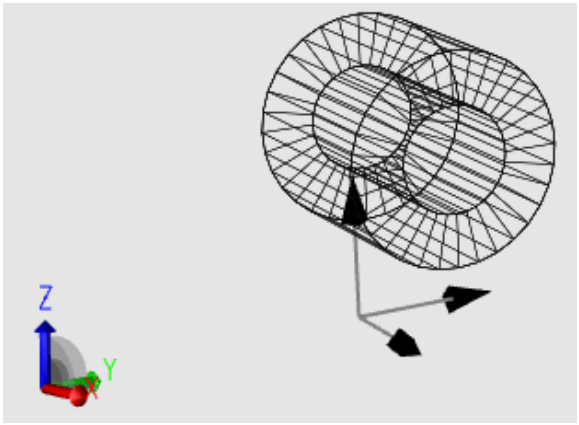
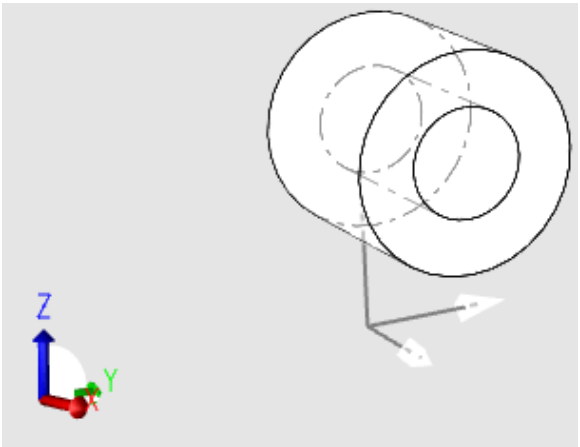
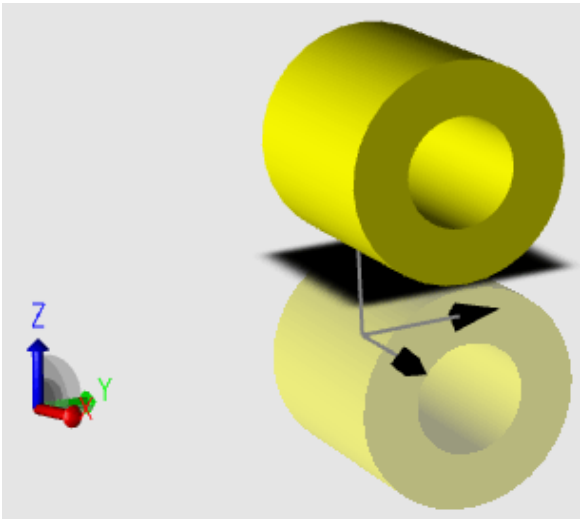
- **Globale Einstellungen**

Die globalen Einstellungen sind genau einer Ansicht zugeordnet, unabhängig davon welches Modell dargestellt ist, und wie viel Ansichten für dieses Modell existieren. Dadurch kann ein Modell gleichzeitig mit unterschiedlichen Einstellungen dargestellt werden.

Die aktuelle Darstellung der Elemente innerhalb einer Ansicht richtet sich nach dem eingestellten Modus, welcher in der Werkzeugleiste angezeigt wird und geändert werden kann. Standardmäßig stehen dem Anwender vier verschiedene Modi zur Verfügung. Es können allerdings auch eigene definiert und verwendet werden.

Die Eigenschaften des jeweiligen Ansichtsmodus werden im Eigenschaftsdialog der Ansicht angezeigt. Die einzelnen Einstellungen können beliebig verändert werden, wobei sich diese Änderungen nur auf die jeweilige Ansicht auswirken. Mithilfe des Buttons *Einstellungen Speichern* werden die vorgenommenen Änderungen für den aktuellen Modus übernommen, und sind damit auch in anderen Ansichten und Modellen verfügbar.

**Hinweis:** Die vordefinierten Modi können nicht überschrieben werden. Veränderungen werden in einem neuen Modus abgelegt, welcher anschließend allen Ansichten als Auswahl zur Verfügung steht.

| **Voreinstellung** | |
|---|---|
| Auswahl des aktuellen Ansichtsmodus<br>Zur Auswahl stehen: | |
| *Standardansicht*<br>  | *Drahtgitter*<br>  |
| *Technische Ansicht*<br>  | *Realistisch*<br>  |

| **Darstellung** | | |
|---|---|---|
| **Kanten** | | |
| Modus | Darstellungsmodus für die Kanten | |
| | *Aus* | Keine Kanten zeigen |
| | *Drahtgitter* | Alle Kanten anzeigen |
| | *Silhouette* | Aussenkanten und harte Kanten anzeigen |
| Verdeckte Kanten | Steuert die Behandlung verdeckter Kanten | |
| | *Ausblenden* | deaktiviert diese Kanten, auch wenn die Flächen aktuell nicht sichtbar sind |
| | *Überblenden* | zeichnet diese Kanten über die sichtbaren Flächen. Nur aktiv wenn Kanten angezeigt werden, deaktiviert anspruchsvolle Darstellungstechniken (siehe Schatten und Reflexion). |
| Farbe | Farbeinstellung der dargestellten Kanten | |
| Lokale Farbe ignorieren | Hier lässt sich einstellen, ob vorhandene Farbeinstellungen in den Elementen ignoriert werden sollen oder nicht. | |
| | *True* | vorhandene Farbeinstellungen in den Elementen werden ignoriert |
| | *False* | die eingestellte Farbe wird nur auf solche Elemente angewendet, welche keine lokalen Kantenfarben definiert haben |
| Beleuchtet | Aktiviert die Beleuchtungsberechnung für dargestellte Kanten (*true/false*). | |
| **Flächen** | | |
| Sichtbar | Steuert die Sichtbarkeit der Elementflächen (*true/false*). | |
| Farbe | Farbeinstellungen der dargestellten Flächen. | |
| Lokale Farbe ignorieren | wie bei Kanten | |
| Beleuchtungs-modell | Auswahl verschiedener Beleuchtungsmodelle | |
| | *Gouraud* | Standard |
| | *Phong Algorithmus* | ist aufwändiger, liefert aber auch bessere Ergebnisse. |
| | *Gooch* | |
| | *Flat* | deaktiviert die Interpolation |

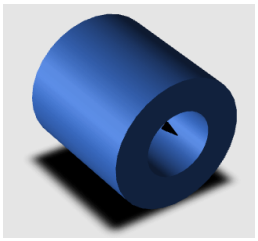
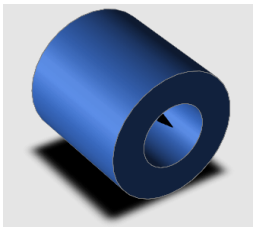
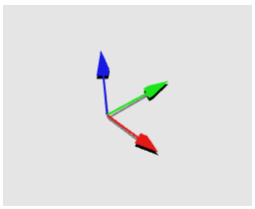
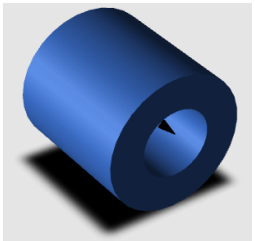
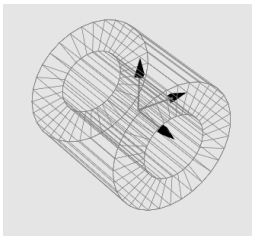
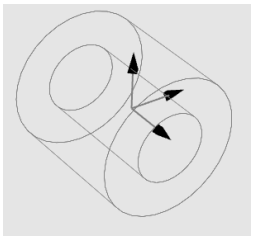
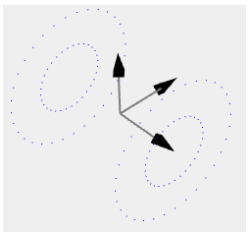
| | | |
|---|---|---|
| | *Aus* | komplette Beleuchtungsberechnung deaktiviert |
| **Allgemein** | | |
| Antialiasing | Aktiviert die Kantenglättung für Flächen, Kanten und Text. Nicht verfügbar bei Verwendung von OpenGL (*true/false*). | |
| Einfache Schatten | Aktiviert eine Schattenebene, auf welche die Schatten der einzelnen Elemente sichtbar sind. Deaktiviert bei aktiver Einstellung verdeckte Kanten (*true/false*). | |
| Reflextion | Aktiviert eine Reflexionsebene, auf der die Reflexionen der einzelnen Elemente sichtbar sind. Deaktiviert bei aktiver Einstellung verdeckte Kanten (*true/false*). | |
| Komplexe Schatten | Aktiviert den Schattenwurf zwischen Elementen. Nur verfügbar bei Verwendung von DirectX (*true/false*). | |
| **Qualität** | | |
| Einfacher Schatten | Qualitätseinstellung für die Schattenebene. (*niedrig/mittel/hoch*) | |
| Transparenz | Qualitätseinstellung für die Darstellung transparenter Elemente. (*aus/niedrig/mittel/hoch*) | |
| Komplexer Schatten | Qualitätseinstellung für die Elementschatten (*niedrig/mittel/hoch*) | |
| Hidden Line Algorithmus | Qualitätseinstellung für die Behandlung verdeckter Linien, siehe Kanten ->verdeckte Kanten. | |

**Tabelle 5.2:** Globale Einstellungen der Ansicht

- **Lokale Einstellungen**

Die lokalen Einstellungen gelten ausschließlich für das betreffende Element inklusive enthaltener Elemente. Allerdings wirken sich diese auf alle Ansichten aus, in denen das Element dargestellt wird.

Generell werden für die Darstellung der Elemente die Eigenschaften der jeweiligen Ansicht, sowie ansichtsrelevante Modellparameter verwendet. In den lokalen Einstellungen ist es möglich, diese zu überschreiben. Änderungen in den lokalen Einstellungen sind dabei immer Modell unabhängig, d.h. hier können auch Änderungen vorgenommen werden, wenn sich das Modell im berechneten Zustand befindet.

| **Anzeige** | | |
|---|---|---|
| Anzeige | Enthält mehrere auswählbare Anzeigeeinstellungen für das Element. Diese Einstellung überschreibt die globalen Einstellungen für die Sichtbarkeit von Flächen und Kanten. | |
| |  | *globale Einstellungen übernehmen (Standard)* |
| |  | *Flächen & Kanten* |
| |  | *Unsichtbar* |
| |  | *Solid* |
| |  | *Drahtgitter* |
| |  | *Objektkanten* |
| |  | *Punkte* |

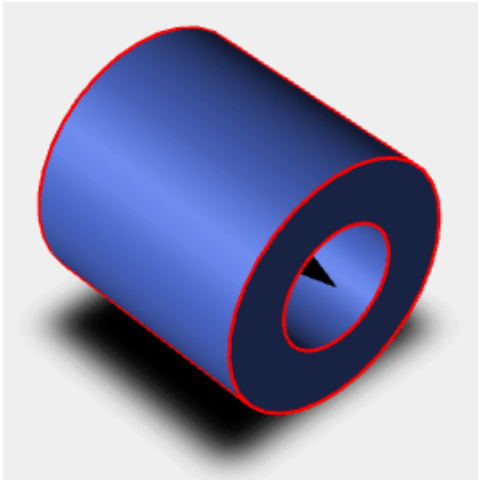
| | |
|---|---|
| Sichtbarkeit Körper-Koordinatensystem | Steuert die Sichtbarkeit des Koordinatensystems des Körpers, welches standardmäßig nur für selektierte Elemente angezeigt wird |
| | *Koordinatensystem immer anzeigen* |
| | *Koordinatensystem nie anzeigen* |
| **Material** | |
| Verwendete Modellparameter | Steuert, ob für die Farbe des Elements der Wert des Modellparameters verwendet werden soll. |
| Farbe | Farbe der Flächen des Objekts |

**Tabelle 5.3:** Lokaler Einstellungen der Ansicht

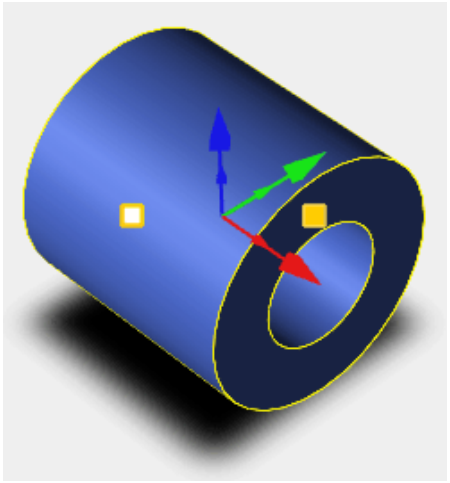
### 5.4.4.4 Arbeiten mit der 3D-Ansicht

- **Auswahl von Elementen**

Die Auswahl von Elementen in der 3D-Ansicht erfolgt mithilfe der Maus. Um den Vorgang zu erleichtern werden Elemente beim Überfahren mit dem Mauszeiger hervorgehoben, sodass erkennbar ist welches Element ausgewählt wird. Selektierte Elemente sind mit einem gelben Rahmen gekennzeichnet, außerdem ist ihr lokales Koordinatensystem sichtbar.



mit Mauszeiger überfahren



ausgewähltes Element

**Bild 5.35:** Auswahl eines Elements

Falls vorhanden, werden auch dessen Konnektoren durch jeweils ein Koordinatensystem, sowie Symbol dargestellt.
Die aktuelle Selektion lässt durch Auswählen weiterer Elemente bei gedrückter *Shift* Taste erweitern. Es können auch mehrere Elemente auf einmal ausgewählt werden, dazu wird die *linke Maustaste* gedrückt gehalten, um einen Auswahl Rahmen aufzuziehen.
Bei einfacher Selektion werden nicht nur das ausgewählte Element, sondern alle

Elemente intern registriert, welche sich bei Betätigung der Maustaste unterhalb des Mauszeigers befanden. Solange keine neue Selektion vorgenommen wird, kann mit der *Tabulator* Taste zwischen den registrierten Elementen weitergeschalten werden.
Eine Selektion bleibt solange bestehen bis ein neues Element ausgewählt, oder die Selektion durch klicken auf einen leeren Bereich der 3D Ansicht aufgehoben wird.

- **Elemente erstellen**

MBS Elemente können innerhalb der 3D-Ansicht erstellt werden. In einfachsten Fall wird das Element aus der Bibliotheksleiste an die gewünschte Position in die 3D-Ansicht gelegt.
Für die Standardkörper steht auch ein Konstruktionsvorgang zur Verfügung, der gestartet werden kann indem bei Ablegen des Elements in die 3D-Ansicht die *Strg* Taste gedrückt wird. Während dieses Vorgangs können die jeweiligen Dimensionen des Elements interaktiv durch die Bewegung der Maus definiert, und mit Betätigung der linken Maustaste bestätigt werden. Als Konstruktionshilfe ist für jeden eindimensionalen Parameter eine Konstruktionsachse vorgegeben, entlang welcher der Wert bei der Bewegung der Maus abgelesen wird. Es ist außerdem möglich mit der Taste *D* zwischen kontinuierlicher und diskreter Konstruktion umzuschalten. Der Konstruktionsvorgang ist beendet, sobald alle Parameter definiert sind, oder der Vorgang mit der Taste *ESC* abgebrochen wird.

- **Objektlage ändern**

Position und Orientierung der MBS Elemente können in der 3D-Ansicht interaktiv verändert werden. Das Verschieben bzw. Drehen des Objekts erfolgt durch bewegen der Maus bei gedrückter *linker Maustaste*. Der jeweilige Transformationsvorgang kann durch zahlreiche Funktionen vereinfacht werden, welche während des Vorganges bei gedrückter *linker Maustaste* durch folgende Funktionen aktiviert bzw. deaktiviert werden können:

| | |
|---|---|
| *linke Maustaste + D* | Umschalten zwischen diskreter/kontinuierlicher Eingabe |
| *Linke Maustaste + X,Y oder Z* | Aktivieren von Bedingungen, die die Transformation entlang definierter Achsen ermöglichen. |
| *Leertaste* | Umschalten des Referenzpunktes, dessen Koordinatenachsen für die Bedingungen verwendet werden. |
| *ESC* | Der Transformationsvorgang wird abgebrochen. |

Beide Vorgänge erfolgen mit gedrückter linker Maustaste. Die vorgenommene Transformation wird bestätigt indem die linke Maustaste losgelassen wird, es sei denn der Vorgang wurde vorher abgebrochen. Die Transformation eines Elementes wird auf alle Nachfolger übertragen, um die kinematische Kette zu erhalten.

Der Rotationsvorgang wird gestartet, indem der Rotationsmodus mit der Taste *R* aktiviert wird. Das anschließend selektierte Element wird bei gedrückter Maustaste gedreht. Ohne definierte Bedingung erfolgt die Rotation um die Sichtachse, mit den entsprechenden Tasten *x, y,* und *z* erfolgt die Rotation um die jeweilige Achse des gewählten Referenzkoordinatensystems. Als Referenz dient standardmäßig das Koordinatensystem des Körpers selber. Zusätzlich kann das Koordinatensystem des Eingabe-Konnektors, sowie, falls das Element verbunden ist, des Ausgabe-Konnektors des Vorgängers und das Koordinatensystem des Vorgängers ausgewählt werden.

Um ein Element zu verschieben kann es an seinen Konnektoren angefasst und bewegt werden. Mithilfe der zuschaltbaren Bedingungen erfolgt die Verschiebung entlang der jeweiligen Koordinatenachse des Referenzsystems. Die Bedingungen lassen sich hier auch kombinieren, um das Element entlang einer Koordinatenebene zu bewegen. Als Referenzsystem stehen hier das Körperkoordinatensystem sowie das globale Koordinatensystem zur Verfügung.

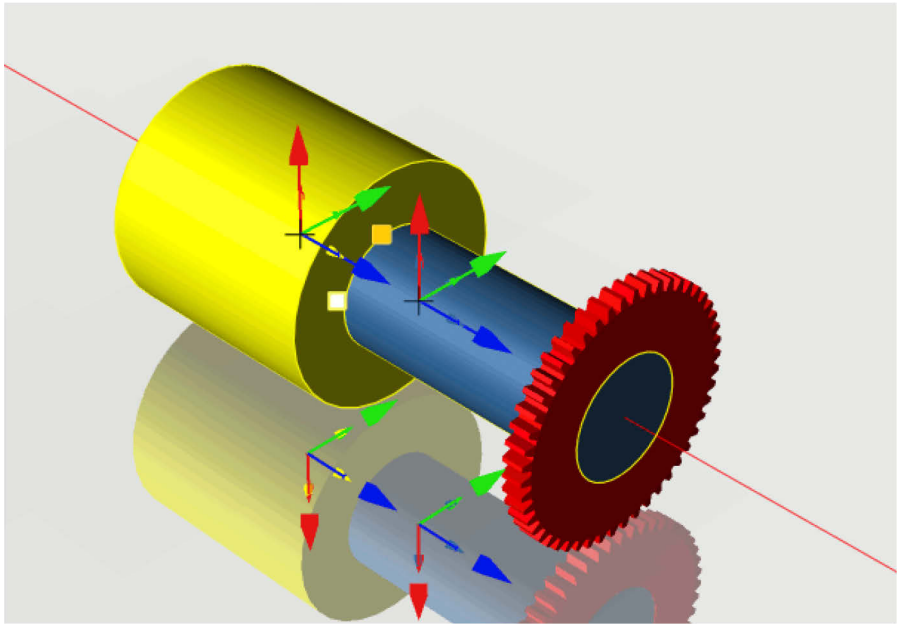


**Bild 5.36:** Element verschieben entlang einer Achse

- **Elemente verbinden**

Um Elemente innerhalb der 3D Ansicht zu verbinden werden die Konnektoren der Elemente übereinandergelegt. Der Vorgang entspricht einer Transformation eines Körpers, indem einer seiner Konnektoren auf einen passenden Konnektor eines anderen Elements gelegt wird. Bei einer Verbindung wird das vorher verschobene Element so positioniert, das sich die Koordinatensysteme der beiden Konnektoren der Verbindung überdecken.
Um den Vorgang zu erleichtern werden alle passenden Konnektoren des aktuellen Kontextes angezeigt, sobald ein Konnektor eines Elements angefasst wird.

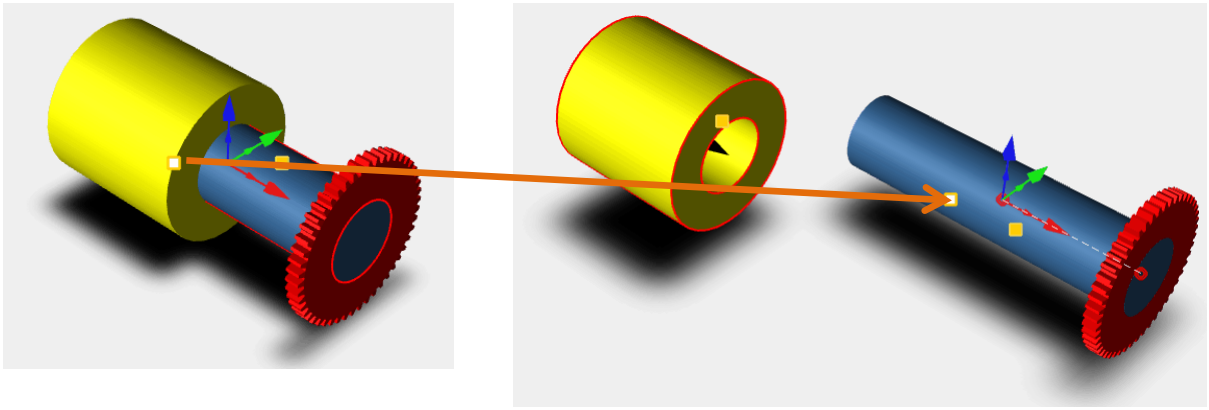
- **Elemente trennen**

Ein verbundenes Element wird an einem seiner Konnektoren angefasst, und von der Verbindung weggezogen.

Das ***Wegziehen am Eingangs-Konnektor*** trennt das Element von seinem Vorgänger, aber nicht von seinen Nachfolgern. Die Nachfolger werden mit dem Element verschoben.

Das ***Wegziehen am Ausgang-Konnektor*** trennt das Element von allen bestehenden Verbindungen.

Eine Rotation eines Elementes erhält alle bestehenden Verbindungen. Um ein Element zu verschieben ohne seine Verbindungen zu trennen, kann die *Shift*-Taste beim initiieren des Verschiebevorgangs gedrückt werden.



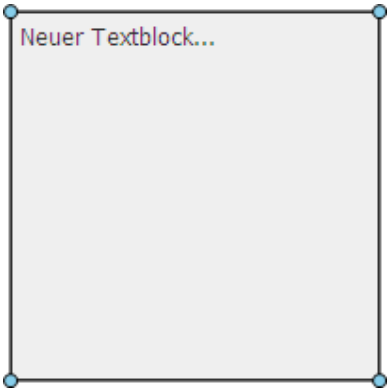
## 5.4.5 Spezielle Modellkomponenten

### 5.4.5.1 Einfügen spezieller Modellkomponenten

SimulationX stellt eine Reihe von Komponenten bereit, mit denen sich ein Simulationsmodell ergänzen lässt. Eine solche Komponente kann über das Menü ***Einfügen*** ausgewählt werden. Im Modell sind diese als rechteckige Blöcke dargestellt. Mit der Maus können sie markiert, verschoben und in der Größe verändert werden. Über ein Kontextmenü lässt sich der zugehörige **Eigenschaftsdialog** öffnen.

- **Textblöcke und Grafiken**

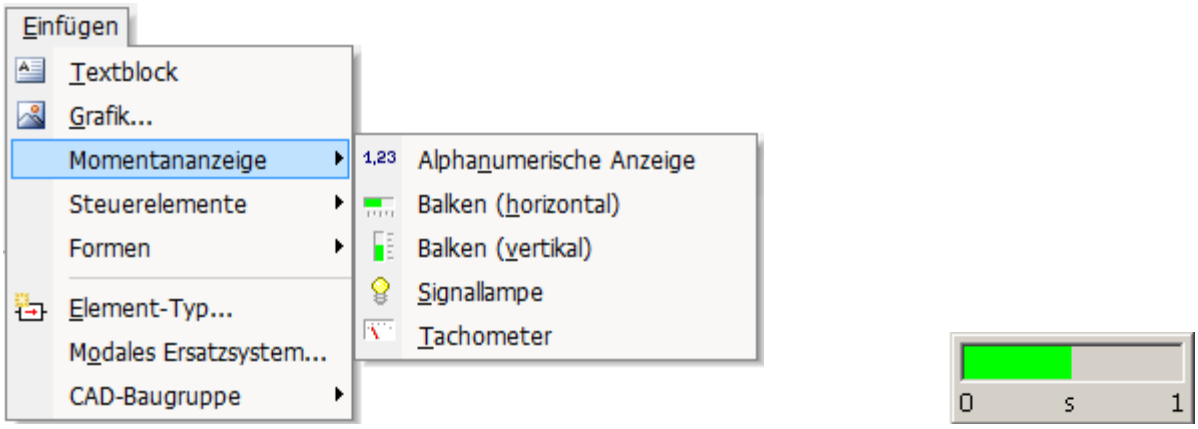
Mit Textblöcken und Grafiken kann ein Simulationsmodell illustriert werden. Durch Klick auf einen Textblock wird dessen Bearbeitungsmodus eingeschaltet. Ein Doppelklick öffnet den Eigenschaftsdialog. Innerhalb der Symbolleiste **Text** stehen umfangreiche Formatierungsmöglichkeiten zur Verfügung.
Über die Zwischenablage können Sie Texte aus gängigen Textverarbeitungsprogrammen übernehmen.



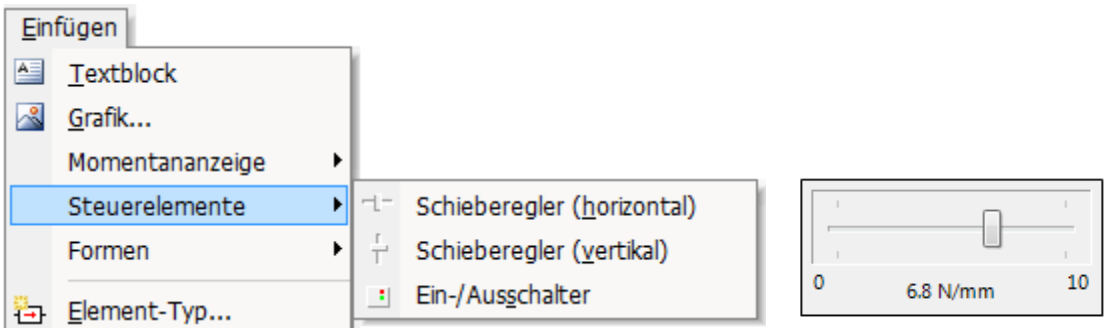
- **Momentananzeigen**

Mit einer Momentananzeige lässt sich der Zustand einer Ergebnisgröße während der Simulationsrechnung numerisch anzeigen oder grafisch visualisieren (s. auch **Kap. 7.2**)

Um einer Momentananzeige eine Ergebnisgröße zuzuweisen, öffnen Sie den Eigenschaftsdialog per Kontextmenü (rechte Maustaste). Auf der Dialogseite Momentananzeige finden Sie ein Auswahlfeld für die gewünschte Ergebnisgröße. Momentananzeigen sind auch für die Animation der Eigenschwingformen verwendbar (siehe **Kapitel 6.5.1.6).**




- **Steuerelemente**




Mit Steuerelementen lassen sich während der Simulationsrechnung Parameter verändern.
Um einem Steuerelement einen Parameter zuzuweisen, öffnen Sie den Eigenschaftsdialog. Auf der Registerseite ***Steuerelement*** finden Sie ein Auswahlfeld mit allen im Modell definierten Parametern, aus dem Sie den gewünschten Parameter auswählen können.

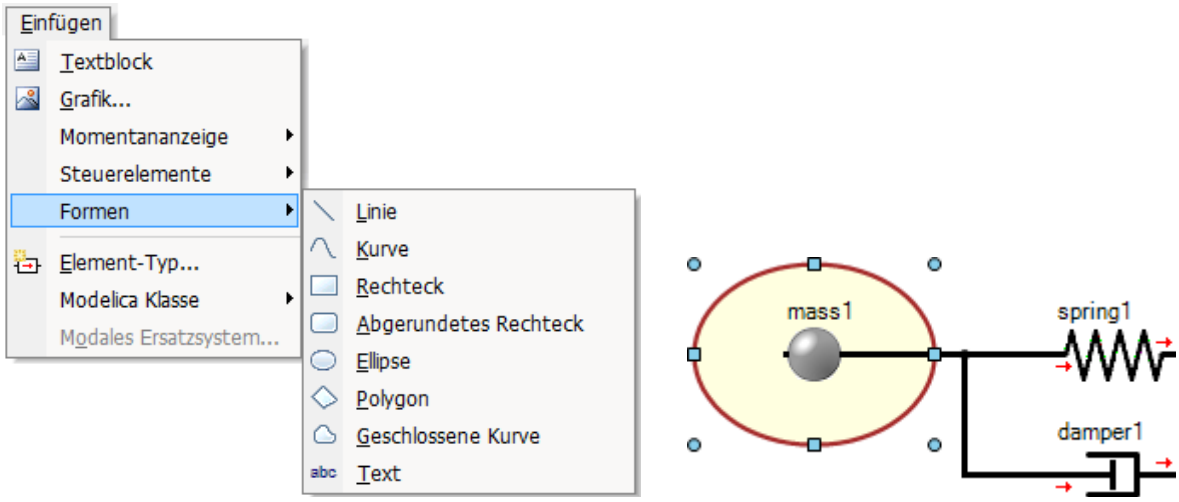
- **Formen**

Mit Formen lässt sich die Strukturansicht eines Modells noch anschaulicher gestalten. Es stehen verschiedene Grafikprimitive zur Verfügung. Die Auswahl der Grafikprimitive erfolgt über das Menü ***Einfügen*** oder die Symbolleiste ***Zeichnen***. In der Symbolleiste ***Zeichnen*** können verschiedene Anpassungen vorgenommen

werden.

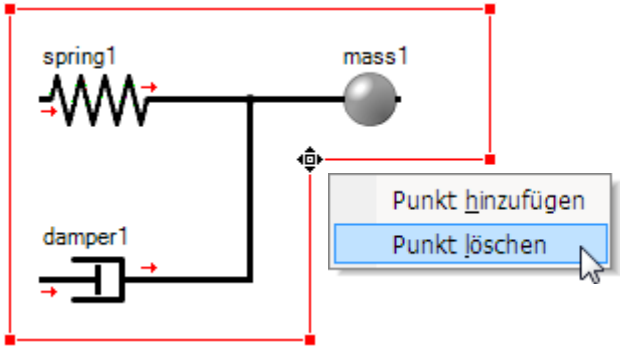
So zeichnen Sie innerhalb der Strukturansicht eine Ellipse:

1. Klicken Sie in der Symbolleiste ***Zeichnen*** auf ***Ellipse.***
2. Ziehe Sie bei gedrückter linker Maustaste in der Strukturansicht die Ellipse auf die gewünschte Größe. Die neu gezeichnete Form ist nun selektiert.
3. Passen Sie Linienstärke, -art und -farbe sowie Füllfarbe und -muster Ihren Wünschen entsprechend an.




So zeichnen Sie innerhalb der Strukturansicht ein Polygon:

1. Klicken Sie in der Symbolleiste ***Zeichnen*** auf ***Polygon.***
2. Klicken Sie in der Strukturansicht auf die Stelle an der sich der erste Punkt des Polygons befinden soll.
3. Klicken Sie nun auf die Stelle, an der sich der zweite Punkt befinden soll. Verfahren Sie so mit allen weiteren Punkten der Kurve.
4. Beenden Sie das Zeichnen mit einem Doppelklick.
5. Sie können die Punkte der Kurve im Nachhinein verschieben sowie über das Kontextmenü neue Punkte hinzufügen oder bestehende löschen.




Formen sind kompatibel zum Modelica-Sprachstandard.

SIMULATION X®

Powered by ITI

Analysis

# Analysis

# 6 Analyse

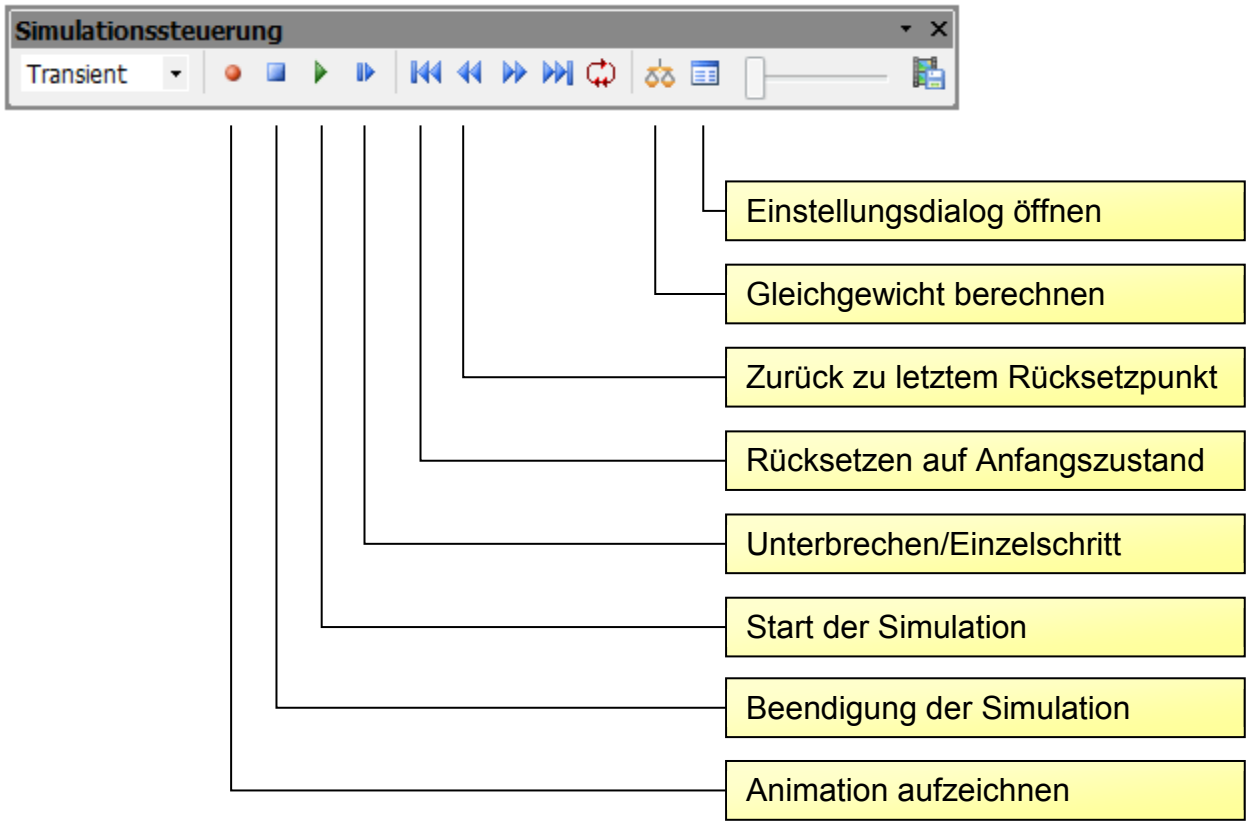
Die folgenden Abschnitte beschreiben die in SimulationX verfügbaren Berechnungs- und Analyseverfahren. Die Berechnungsaufgaben für die Modelle sind:

- **Transiente** Simulation **im Zeitbereich**
- **Animation** der Berechnungsergebnisse
- **Stationäre** Simulation **im Frequenzbereich**
- Berechnung des **Gleichgewichtszustandes**
- **Lineare Systemanalyse.**

Des Weiteren wird die **Variantenrechnung** beschrieben. Als Drittes lernen Sie mit der **Ordnungsanalyse** ein Werkzeug zur Ergebnisauswertung bei der Drehschwingungsanalyse kennen.

## 6.1 Transiente Simulation im Zeitbereich

Die Steuerung der Simulation im Zeitbereich wird über die Schaltflächen der Symbolleiste ***Simulationssteuerung*** vorgenommen. Kontrollieren Sie vor Beginn der Rechnung, ob in der Auswahlbox die Analyseart ***Transient*** eingestellt ist.




Die Berechnung erfolgt im Zeitintervall `tStart ≤ t ≤ tStop`. Für die Variablen, deren Protokollattribut gesetzt ist, werden die Berechnungsergebnisse über der Simulationszeit gespeichert. Während der Rechnung werden sichtbare

Ergebnisfenster ständig aktualisiert, so dass sich der Verlauf von Berechnungsergebnissen unmittelbar verfolgen lässt.
Bei der Ausführung im Schrittbetrieb wird jeweils ein Löserschritt berechnet. Zum Zeitpunkt `t=tStart` wird dabei kein Zeitschritt, sondern nur die Anfangswertberechnung ausgeführt. Die Ergebnisse der Anfangswertberechnung werden im Modellexplorer angezeigt. Bleibt die Schaltfläche ***Unterbrechen/Einzelschritt*** gedrückt, wird die schrittweise Berechnung bis zum Loslassen fortgesetzt.

Bei Betätigung der ***Stopp***-Taste während einer laufenden Rechnung wird bis zum Ende des aktuellen Zeitschritts weitergerechnet und damit ein gültiger Wert sichergestellt. Bei großen Modellen kann die Berechnung dieses Schritts einige Zeit dauern, so dass der Eindruck entsteht, die Simulation würde nicht angehalten. Wird die ***Stopp***-Taste dann erneut betätigt, erfolgt ein sofortiger Abbruch.

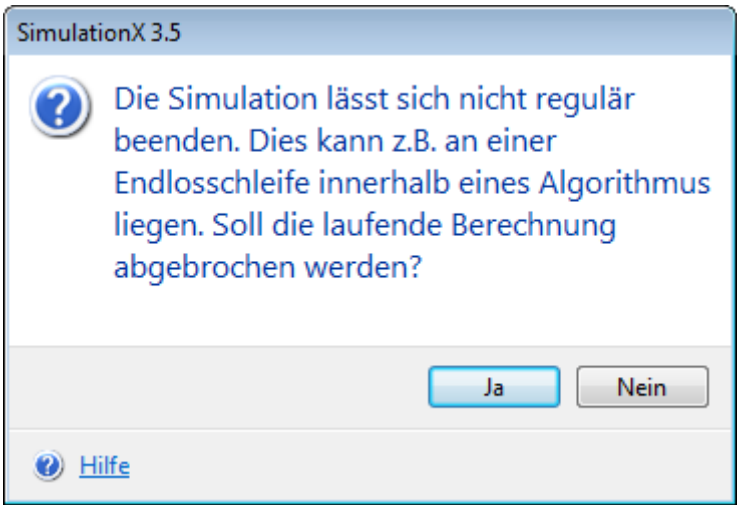



**Bild 6.1:** Abbruch der Berechnung

Die Rechnung wird in diesem Fall auf den letzten gültigen Zeitpunkt zurückgesetzt. Bei Beendigung der Rechnung sowie Abbruch der Rechnung durch Betätigung der ***Stopp***-Taste oder bei Auftreten eines Fehlers wird im Ausgabebereich neben der Rechenzeit auch die aktuelle Simulationszeit ausgegeben.

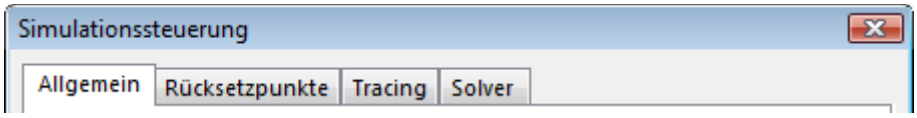
Beim ***Rücksetzen*** eines Modells werden folgende Aktionen durchgeführt:

- Die aktuelle Simulationszeit `t` wird auf die Startzeit `tStart` gesetzt.
- Alle Ergebnisprotokolle von Variablen werden gelöscht.
- Wurde das Ergebnis einer vorangegangenen Rechnung als Initialisierungszustand gespeichert (siehe **6.1.1.3**), so wird das Modell mit diesem Zustand initialisiert. Die Anfangswerte (in den Elementen und Verbindungen) des Modells sind in diesem Fall deaktiviert, d.h. sie werden beim Start einer neuen Simulation ignoriert.

Über den Menüpunkt ***Simulation/Einstellungen Transient…*** oder den Button können die Eigenschaften des Lösungsverfahrens beeinflusst werden. Dazu gehören die Einstellung von verfahrensspezifischen Parametern (Registerseiten ***Allgemein*** und ***Solver***) und das Anzeigen bestimmter Ereignisse während der Berechnung (***Tracing***).
Ihr Simulationsmodell wird in ein System differentiell-algebraischer Gleichungen überführt. Dieses wird durch einen Löser berechnet.

## 6.1.1 Simulationssteuerung

Im Folgenden sind alle Simulationseinstellungen zusammengefasst und kurz erklärt. Über das Menü ***Simulation/Einstellungen Transient…*** gelangen Sie zum Dialog für die *Simulationssteuerung*.




Hier können Sie alle Einstellungen, die die Berechnung des Modells betreffen, wie zum Beispiel Simulationszeit, Schrittweite, Genauigkeit oder Löser vornehmen.

### 6.1.1.1 Allgemeine Einstellungen

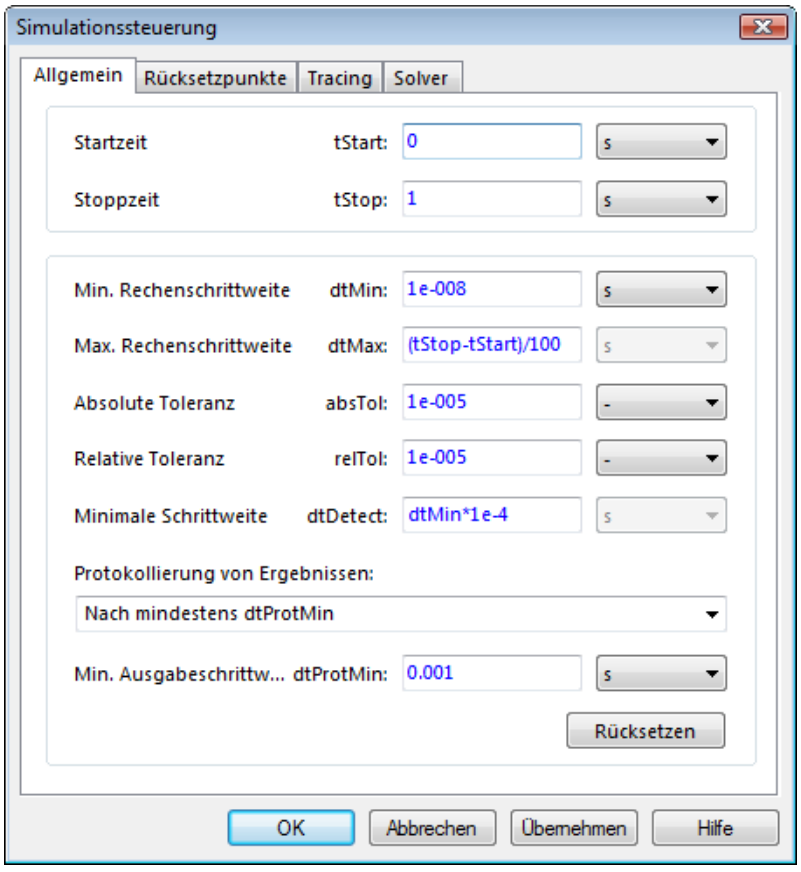



**Bild 6.2:** Allgemeine Einstellungen

Die Registerseite *Allgemein* des Dialoges *Simulationsteuerung* enthält folgende Parameter (siehe **Bild 6.2**):

- Mit der **Start-** und **Stoppzeit** wird das Simulationsintervall festgelegt. Die Stoppzeit muss immer größer als die Startzeit sein.
- Die **Minimale Rechenschrittweite** `dtMin` wird vom Löser - außer bei der Suche nach einer Unstetigkeit - nicht unterschritten.
- Die **Maximale Rechenschrittweite** `dtMax` bestimmt die obere Grenze der vom Löser benutzten Schrittweite. Als Standardwert ist `(tStop-tStart)/100` eingetragen. Dadurch werden mindestens 100 Rechenschritte für das Simulationsintervall ausgeführt. Die Begrenzung der maximalen Schrittweite ist insbesondere für Modelle ohne Zustandsgrößen und Ereignisse wichtig. Bei solchen Modellen würde die Schrittweite sonst bis zur Stoppzeit anwachsen.
- Die Genauigkeit der Rechnung kann mit den Parametern **Absolute Toleranz** `absTol` und **Relative Toleranz** `relTol` beeinflusst werden. Daraus berechnet das Programm die maximale Abweichung, die während eines Simulationsschrittes auftreten darf:
  `relTol * Betrag der Zustandgröße + absTol`
- Die **Minimale Schrittweite** `dtDetect` hat Einfluss auf die Genauigkeit, mit der Ereignisse angefahren werden. Es ist sichergestellt, dass das berechnete Ereignis höchstens um den Wert `dtDetect` nach dem tatsächlichen Ereigniszeitpunkt liegt.
- Durch die **Minimale Ausgabeschrittweite** `dtProtMin` wird angegeben, mit welchem minimalen zeitlichen Abstand Ergebniswerte aufgezeichnet werden dürfen. Das heißt, zwei aufeinanderfolgende Protokollpunkte dürfen keinen kürzeren (jedoch einen größeren) Zeitabstand haben.
- Über die **Protokollierung von Ergebnissen** wird festgelegt, zu welchen Zeitpunkten und Ereignissen Berechnungsergebnisse protokolliert werden. Folgende Einstellungen sind hierfür möglich:
  - Äquidistant mit `dtProtMin`
  - Nach mindestens `dtProtMin`
  - Nach mindestens `dtProtMin` und nach Ereignissen
  - Nach mindestens `dtProtMin` sowie vor und nach Ereignissen
  - Nach mindestens `dtProtMin` und Eventiterationsschritten
  - Letzter Wert

Die Werte für die hier beschriebenen Systemvariablen können Sie auch im Modellexplorer oder im Eigenschaftsdialog für das Modell ändern. In diesen beiden Anzeigen finden Sie zusätzlich die Variablen

- **`traceOn`** Eine Beschreibung dazu finden Sie in **Kapitel 8.2.7**.

- **protOn** erlaubt die bedingte Protokollierung von Ergebnissen (standardmäßig **true** – ein), z. B. t >= 0.6 and t <= 0.7 zeichnet nur Werte in diesem Zeitbereich auf.

### 6.1.1.2 Rücksetzpunkte

Mit Hilfe eines Rücksetzpunktes kann man am Ende einer Simulation das Modell auf einen diskreten Zeitpunkt `tRP` (mit `tStart < tRP < t`) zurücksetzen. Rücksetzpunkte können manuell oder automatisch erzeugt werden (Dialog ***Simulationssteuerung*** Registerseite ***Rücksetzpunkte***). Dies muss jedoch **vor** Beginn der Simulation definiert werden. Das Rücksetzen zu einem Rücksetzpunkt kann nur über den Dialog *Simulationssteuerung/Rücksetzpunkte* erfolgen. Hierbei werden folgende Aktionen durchgeführt:

- Die aktuelle Simulationszeit t wird auf die Zeit des Rücksetzpunktes `tRP` gesetzt.
- Aus den Ergebnisprotokollen von Variablen werden alle Einträge mit `t > tRP` gelöscht.
- Alle bereits erzeugten Rücksetzpunkte, die zeitlich **nach** dem gewählten Rücksetzpunkt liegen, werden gelöscht.
- Das Modell wird mit dem Zustand des Rücksetzpunktes neu initialisiert.

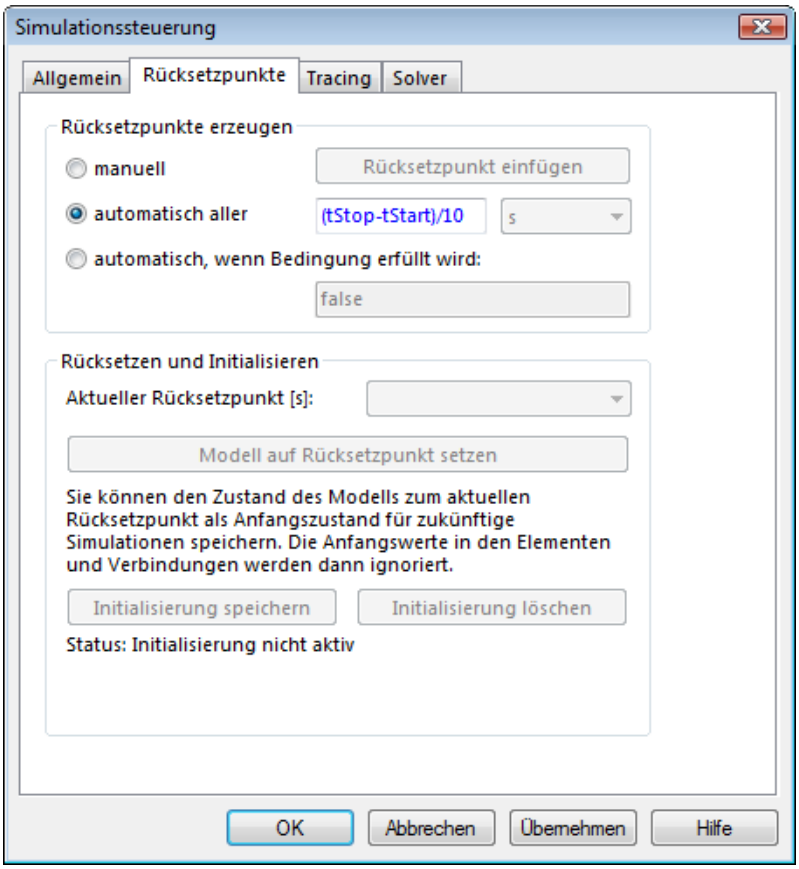
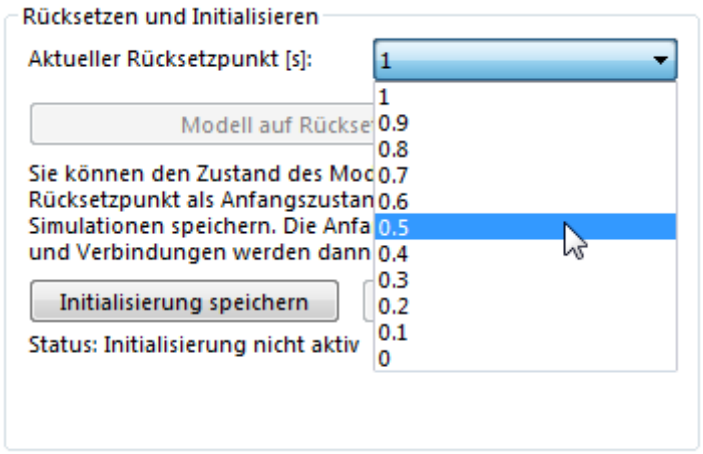



**Bild 6.3:** Dialog Rücksetzpunkte

**Bild 6.3** zeigt eine automatische Erzeugung von 10 Rücksetzpunkten über die Simulationszeit.




Nach einem Simulationslauf können Sie das Modell dann über einen erneuten Aufruf des Dialoges ***Rücksetzpunkte*** auf einen der erzeugten Punkte setzen und von diesem aus die Simulation starten.

### 6.1.1.3 Initialisierungszustand speichern

Mit Hilfe dieser Funktion ist es möglich, den aktuellen Zustand eines Modells oder den Zustand in einem Rücksetzpunkt (beziehungsweise zum aktuellen Zeitpunkt) als Anfangszustand für weitere Simulationen festzulegen. Hiermit können z.B. (transient oder statisch) berechnete Gleichgewichtszustände als Ausgangspunkt für weitere transiente Simulationen (für Variantenrechnungen oder Optimierungen) gewählt werden.
Wurde ein solcher Initialisierungszustand gespeichert (Dialog ***Simulationssteuerung/Rücksetzpunkte*** über den Schalter ***Initialisierung Speichern***), so wird

- dem Parameter „Startzeit“ `tStart` die jeweilige Zeit `t` des Initialisierungszustandes zugewiesen (anderenfalls würden bei zeitabhängigen Modellen inkonsistente Zustände entstehen),
- als Status die Meldung „Initialisierung aktiv“ angezeigt,
- nach dem Rücksetzen das Modell mit diesem Zustand automatisch initialisiert.

Die Anfangswerte in den Elementen und Verbindungen des Modells werden gesperrt, da diese bei einem erneuten Start der Simulation ignoriert werden.
Der gespeicherte Initialisierungszustand kann jedoch mit Hilfe der Schaltfläche ***Löschen Initialisierung*** (Dialog ***Simulationssteuerung/Rücksetzpunkte***) wieder gelöscht werden. Dabei wird

- dem Parameter „Startzeit“ `tStart` wieder sein alter Wert zugewiesen,
- als Status die Meldung „Initialisierung nicht aktiv“ angezeigt,
- nach dem Rücksetzen des Modells **keine** Initialisierung durchgeführt.

Bei einem Neustart der Simulation werden nun wieder die Anfangswerte in den Elementen und Verbindungen des Modells verwendet.

### 6.1.1.4 Tracing-Einstellungen

Während der Berechnungen können verschiedene Informationen angezeigt werden, um Ihnen oder dem ITI-Support-Team eine Fehlersuche zu erleichtern. Mit Hilfe der Baumansicht kann die Anzeige verschiedener Trace-Nachrichten (Menüpunkt ***Simulation/Einstellungen Transient…*** in der Registerseite ***Tracing***) im Ausgabebereich ein- bzw. ausgeschaltet werden.

Für den Löser sind das in der Hauptsache folgende Punkte:

- Auswertung (Statistik über die ausgeführten Rechenschritte, benötigte Zeit...),
- Aktuelle Aktion (protokolliert den Ablauf des Lösungsprozesses),
- Informationen zu gültigen bzw. ungültigen Schritten,
- Anzeige der berechneten Zustandsgrößen, Ableitungen, Newtonkorrekturen, Residuen und Jacobimatrizen,
- Indizes der Zustandsgrößen,
- Zustandsgrößen, die ihren zulässigen Bereich verlassen wollen
- Fehler in den Residuen durch Lösung des linearen Systems (bei nichtlösbaren linearen Gleichungssystemen oder Rundungsfehler).

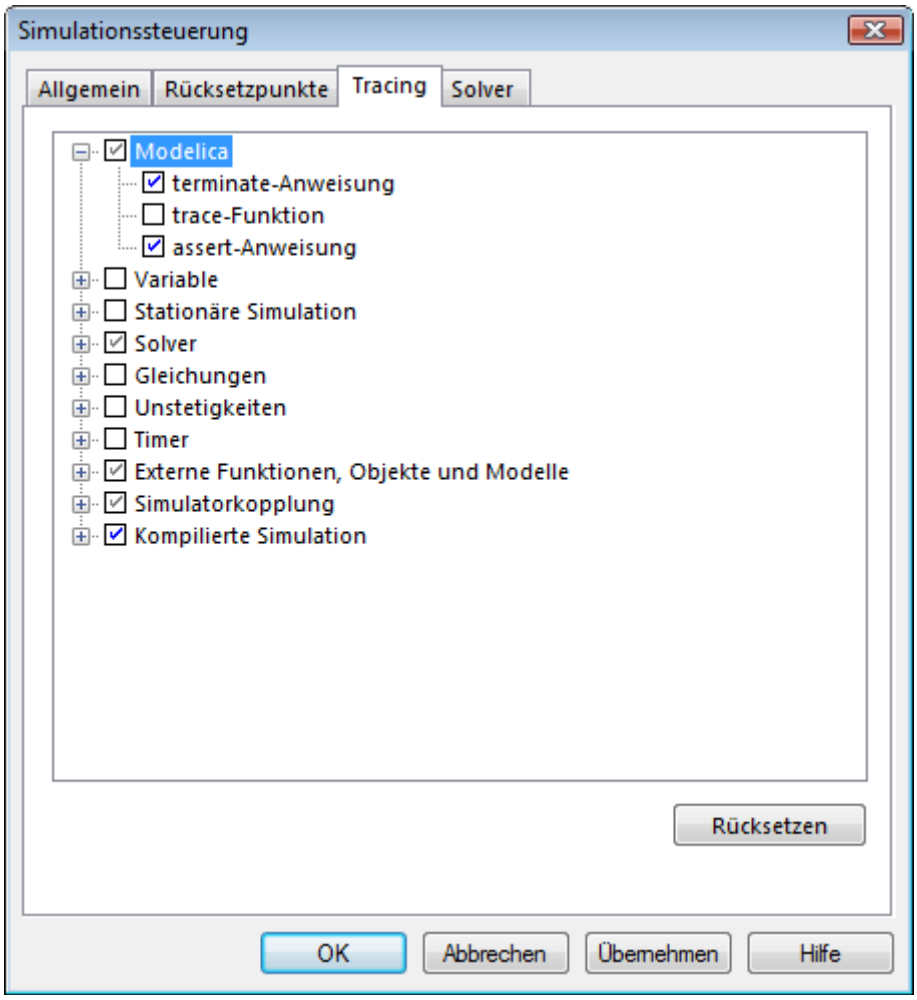



**Bild 6.4:** Dialog Tracing

Andere Tracing-Informationen betreffen:

- Modelica (Ausschriften von `trace-` und `terminate`-Anweisungen),
- Variablen (diskrete Variablen, Überschreitungen von Grenzen beim Wechsel zwischen Bereichen mit unterschiedlichem Verhalten),
- Ereignisse (Zeitpunkt und Quellen von Ereignisse),
- Timer (Quellen und Werte von zeitabhängigen Ereignissen).

Einige der Informationen sind nur für bestimmte Löser verfügbar. Die meisten Informationen können beim BDF- und MEBDF-Verfahren ausgegeben werden.

Die Trace-Informationen werden als Baum dargestellt. Es gibt verschiedene Arten:

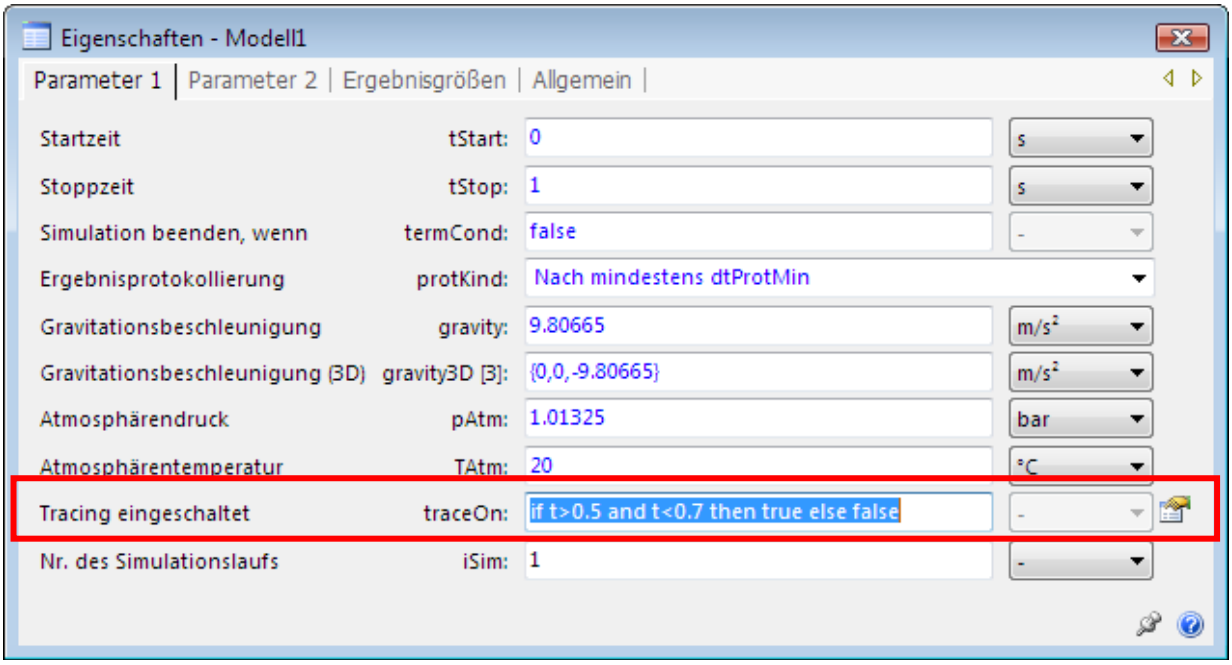
- Werte
- Informationen
- Warnungen
- Fehler

Wenn Sie die Einträge anklicken, erhalten Sie weitere Informationen. Die angezeigten Namen der Variablen sind nach folgenden Regeln zu interpretieren:

- "_der_" bedeutet Ableitung, z. B. _der_connection1.v ist die Ableitung der Geschwindigkeit connection1.v und somit ein Synonym der Beschleunigung connection1.a
- "_" im Namen einer Verbindung zeigt, dass diese Verbindung an ein Compound angeschlossen ist, zum Beispiel connection1_.p0
- compound1.connection1.p bedeutet p der Verbindung connection1 in compound1
- Für Namen von im Modell vorkommenden Veränderlichen, für die nicht die ersten beiden Fälle zutreffen, finden Sie die zugehörigen Variablen im Modell mit der Suchfunktion. Dazu ist der Name zu markieren, mit der rechten Maustaste anzuklicken und im Dialog "Suchen im Modell" zu wählen.

Werte werden nur angezeigt, wenn der Parameter `traceOn` des **Modells** auf `true` steht. Durch Doppelklick in die Strukturansicht gelangen Sie zum Eigenschaftsdialog des Modells, wo sie über das Register *Parameter 1* auf diesen Parameter zugreifen können.

Um das Tracing auf ein Zeitintervall zu beschränken, tragen Sie dort folgende Relation ein: `t>0.5 and t<0.7` oder wie im folgenden Bild:




Die Tracing-Einstellungen werden nicht mit dem Modell gespeichert. Sie werden in der Windows-Registry gespeichert und gelten somit für alle Modelle auch nach einem Neustart von SimulationX.

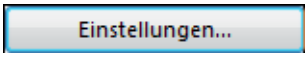
### 6.1.1.5 Lösereinstellungen

Auf der Seite "***Solver***", kann einer der folgenden Löser ausgewählt werden:

- BDF-Verfahren
- MEBDF-Verfahren
- Löser mit konstanter Schrittweite (externer Löser)
- CVODE-Verfahren (externer Löser)

Weitere Einstellungen hängen von der getroffenen Auswahl ab und werden in den entsprechenden Abschnitten beschrieben.

Über die Schaltfläche ***Weitere Parameter...*** erhalten Sie Zugang zu zusätzlichen Einstellungen des aktuell ausgewählten Lösers. **Änderungen an diesen Parametern sollten nur durch erfahrene Nutzer erfolgen!** Bitte kontaktieren Sie im Zweifelsfall den ITI-Support.

Mit der Schaltfläche 

können Einstellungen der Globalen Symbolischen Analyse verändert werden. Diese haben folgende Bedutung:

- Für ODE optimiert:
  Wenn das resultierende Gleichungssystem eines Modells ein System von gewöhnlichen Differentialgleichungen (ODE) bildet, kann diese Einstellung die Performanz der Simulation verbessern. Wenn es sich nicht um eine ODE handelt, wird eine Warnung ausgegeben. Diese Option ist nur im Zusammenhang mit dem BDF- oder MEBDF-Verfahren wirksam.

- Keine symbolische Ableitung:
  Die Globale Symbolische Analyse vermeidet es, einige Gleichungen zu differenzieren, um die Zahl der Zustandsgrößen zu reduzieren.
- Keine symbolische Integration:
  Die Globale Symbolische Analyse vermeidet es, einige Gleichungen zu integrieren, um die Zahl der Zustandsgrößen zu reduzieren.
- Keine Indexreduktion:
  Wenn das Gleichungsystem eine DAE mit einem höheren Differentiationsindex bildet, werden normalerweise einige Gleichungen symbolisch differentiert, um den Index zu reduzieren. Das kann bei Nutzung des BDF- oder MEBDF-Verfahrens ausgeschaltet werden. Ein kleineres finales Gleichungssystem und eine kürzere Simulationszeit kann die Folge sein, allerdings verbunden mit möglichen Schwierigkeiten während der Behandlung von Unstetigkeiten und einem leichten Drift der Ergebnisse.
- Numerische Berechnung der Jacobimatrix:
  Normalerweise wird die Jacobimatrix symbolisch berechnet. Wenn man diese Option ausschaltet, wird die Jacobimatrix durch SimulationX numerisch mittels Differenzenquotienten berechnet. Die Berechnungszeit, die für die Globale Symbolische Analyse **vor** der eigentlichen Simulation benötigt wird, sinkt. Allerdings muss man bedenken: die Berechnung der numerischen Jacobimatrix erfordert meistens mehr Zeit **während** der Simulation. Außerdem sind solche Matrizen oft weniger genau als die symbolischen.
- Nutze alternative numerische Ableitung von Funktionen
  Statt der Zeitableitung mit Störung der Eingangsgrößen der Funktion wird eine partielle Ableitung mit Bildung des Differenzenquotienten verwendet.
- Auflösen linearer Gleichungssysteme
  Es wird der lineare Gleichungsanteil eines Gleichungsblockes ermittelt und so weit wie möglich aufgelöst.
- Große Ausdrücke reduzieren (für C-Code Generierung)
  Bei Problemen des C-Compilers mit sehr langen Ausdrücken kann mit dieser Einstellung durch Einführung von Teilausdrücken Abhilfe geschaffen werden.
- Keine Events bei min/max Funktion
  Es werden nun keine Events für min/max Funktion mehr generiert.

### 6.1.2 Ablauf der Berechnung im Zeitbereich

Die Berechnung im Zeitbereich führt folgende Schritte aus:

Nach dem Betätigen des ***Start***-Schalters beginnt die Globale Symbolische Analyse. Dieser Schritt überführt das SimulationX-Modell, das in der Modellierungssprache Modelica repräsentiert wird, in eine berechenbare Darstellung und vereinfacht es. Das Modell befindet sich nun in einer Löser-internen Form.

Konsistente Anfangswerte werden als Nächstes bestimmt. Die Simulation über der Zeit startet danach. Nach jedem erfolgreichen Zeitschritt überprüft der Löser, ob ein Ereignis (z.B. eine Unstetigkeit) aufgetreten ist. Wenn das der Fall ist, wird der

Zeitpunkt dieses Ereignisses so präzise wie möglich bestimmt, die kontinuierliche Integration angehalten und das Modell reinitialisiert.

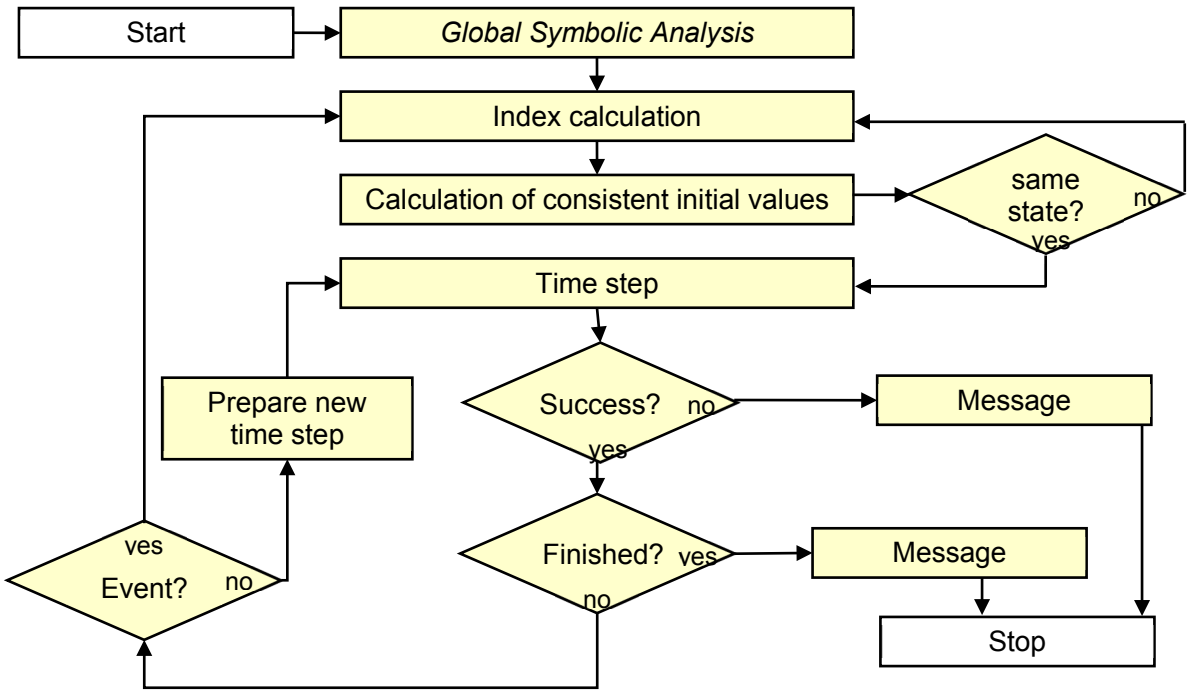



**Bild 6.5:** Ablaufschema für die Simulation im Zeitbereich

Im folgenden Abschnitt werden die Schritte im Detail beschrieben.

### 6.1.2.1 Die Globale Symbolische Analyse

Die Globale Symbolische Analyse überführt das Modell in eine berechenbare Darstellung und vereinfacht es. Die folgenden Analyseschritte werden durchgeführt:

- Umwandlung des hierarchischen Modells in ein “flaches”. Dafür werden Compounds und abgeleitete Typen aufgelöst.
- Auflösung der Referenzen
- Ersetzen einfacher Gleichungen
- Bestimmung des Indexbereichs der Schleifenvariablen
- Bestimmung der Dimension der Variablen
- Umwandlung der multidimensionalen Gleichungen in eindimensionale Gleichungen
- Auflösung von Sprachkonstrukten
- Bildung von Unstetigkeiten (Zerofunctions / root functions)
- Einführung von Variablen für spezielle Funktionen
- Zuweisung von Variablen zu den Gleichungen
- Umstellung der Gleichungen nach den zugewiesenen Variablen
- Bestimmung der Berechnungsreihenfolge der Gleichungen
- Bestimmung von Zuständen und Gleichungen

- Indexreduktion
- Symbolische Jacobimatrix

Abhängig vom gewählten Löser wird das Modell nun entweder als ein implizites System von differentiell-algebraischen Gleichungen (DAE) repräsentiert:

$$0 = f(x, \dot{x}, p, t), \tag{6.1}$$

oder als ein System gewöhnlicher Differentialgleichungen (ODE):

$$\dot{x} = f(x, p, t), \tag{6.2}$$

mit:

x ..... Zustandsgröße
$\dot{x}$ .... Zeitableitung der Zustandsgröße
p ..... Parameter
t ..... Zeit

Im zuletzt genannten Fall werden die impliziten Teile des Modells durch kleine lineare oder nichtlineare Gleichungssysteme während der Berechnung der rechten Seite (RHS) der Differentialgleichung gelöst.

#### 6.1.2.2 Berechnung konsistenter Anfangswerte

Die Anfangswerte charakterisieren den Zustand des Modells zu Beginn der Rechnung. Sie können von Ihnen an den Verbindungen (z. B. Mechanik) und/oder an bestimmten Elementen (elektrischer Kondensator) vorgegeben werden. In SimulationX sind die Anfangswerte spezielle Parameter des Modells. Sie haben folgende Eigenschaften:

- Sie müssen konstant sein. Die Werte werden nur zum Startzeitpunkt ausgewertet. Sie können auch Ausdrücke und Funktionen eingeben. Es erscheint eine Fehlermeldung, falls diese nicht konstant sind.
- Anfangswerte können als ***fixiert*** oder ***frei*** vorgegeben werden. Klicken Sie dazu auf das Symbol auf der rechten Seite des Eingabefeldes.

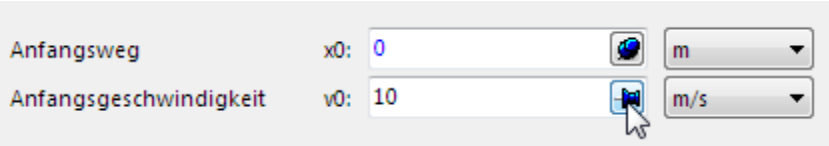



**Bild 6.6:** Ein fixierter und ein freier Anfangswert

Per Default sind alle Anfangswerte mit dem Wert Null fixiert. Bei den meisten Modellen entspricht das dem niedrigsten Energiegehalt im System.

Im Allgemeinen können Sie die Anfangswerte einfach eingeben und fixieren. Bei bestimmten Modellklassen ist das etwas komplizierter.

Im Folgenden wird das näher erläutert.

Anfangswerte sind dann konsistent, wenn die Werte der Zustandsgrößen und ihrer Ableitungen das System differentiell-algebraischer Gleichungen zum Anfangszeitpunkt erfüllen.

Die Anfangswerte gewöhnlicher Differenzialgleichungen (ODE: **O**rdinary **D**ifferential **E**quation) sind immer konsistent, d.h. für jeden Satz von Anfangswerten. Der Berechnungsalgorithmus muss hier lediglich die höchsten Ableitungen der Zustandsgrößen berechnen.
Das soll am Beispiel eines mechanischen Einmassenschwingers gezeigt werden:

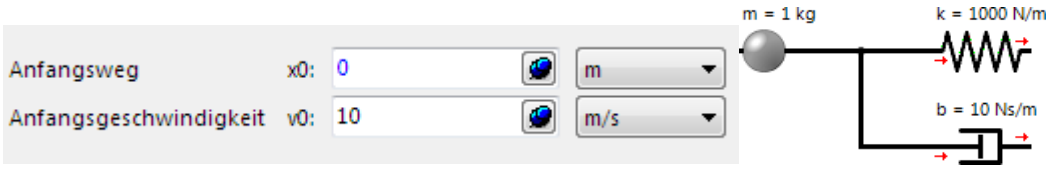



**Bild 6.7:** Modell eines Ein-Massen-Schwingers mit Anfangswerten

Als Anfangswerte sind die Geschwindigkeit und die Auslenkung der Masse vorgegeben. Als Anfangswert für die Beschleunigung wird

$$a(t=0) = -1100\frac{m}{s^2} \qquad \textbf{(6.3)}$$

berechnet.

Wenn in der Gleichung auch algebraische Anteile auftreten, besitzt das Modell nicht für jeden Anfangswert eine Lösung.
Beispiele dafür sind:

- Modelle aus der Mechanik mit Zwangsbedingungen (Gelenke, Übersetzungen),
- Elektrische Schaltungen mit parallelgeschaltenen Kondensatoren (C-Maschen) oder Induktivitäten in Reihe (L-Knoten).

Man spricht dann von differentiell-algebraischen Gleichungen (DAE: **D**ifferential **A**lgebraic **E**quation). Hier sind die Anfangswerte nicht mehr automatisch konsistent, wie folgendes Beispiel zeigt:

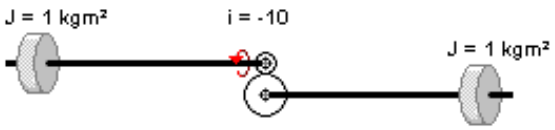



**Bild 6.8:** Mechanik-Modell mit einer Zwangsbedingung (Übersetzung)

Die Übersetzung entsteht hier durch die ideale (starre) Modellierung eines Getriebes.

Sie könnten jetzt an beiden Trägheiten Anfangswerte für Winkel und Drehzahl vorgeben. Jedoch nicht alle Kombinationen erlauben dem Berechnungsverfahren das Herstellen eines konsistenten Zustandes!

Die folgende Tabelle zeigt einige Kombinationen:

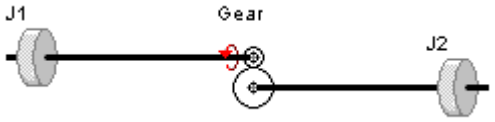
| | |  | |
|---|---|---|---|
| 1 | Inkonsistent und fixiert | phi0: 0<br>om0: 10 | phi0: 0<br>om0: 0 |
| 2 | Inkonsistent und frei | phi0: 0<br>om0: 10 | phi0: 0<br>om0: 0 |
| 3 | Konsistent und fixiert | phi0: 0<br>om0: 10 | phi0: 0<br>om0: -100 |

**Tabelle 6.1:** Verschiedene Kombinationen von Anfangswertvorgaben

In der ersten Variante sind auf beiden Seiten der Zwangsbedingung Anfangswerte vorgegeben und fixiert. Diese sind aber nicht konsistent. Der Algorithmus kann keine Konsistenz herstellen, da ihm dazu keine freien Größen zur Verfügung stehen. Diese fehlerhafte Parametrierung entsteht zumeist, wenn nur auf einer Seite J1 der Zwangsbedingung Anfangswerte vorgegeben werden. Die Anfangswerte von J2 entsprechen noch den Default-Werten und müssen von Ihnen verändert werden.

Bei den anderen beiden Möglichkeiten können konsistente Anfangswerte hergestellt werden: In Variante 2 kann `J2.om0` vom Algorithmus so bestimmt werden, dass die Zwangsbedingung nicht verletzt wird. In der dritten Parametrierung sind die Anfangswerte schon konsistent.

**Hinweis für ITI-SIM Benutzer:** Dieses Beispielmodell konnten Sie mit ITI-SIM nicht aufbauen! Zwei Trägheiten (rotatorische Knoten) konnten dort nicht starr miteinander verbunden werden. Zwischen der Übersetzung und einer der Trägheiten musste in ITI-SIM ein Element (z. B. eine Feder) geschaltet werden. Dadurch entspricht das Modell einer gewöhnlichen Differenzialgleichung und beliebige Kombinationen von Anfangswerten waren möglich, weil die Feder als elastisches Element die Verdrehung aufnehmen kann. Inkonsistente Anfangswerte konnten in ITI-SIM also nicht auftreten.

Folgende Hinweise sollen Ihnen den Umgang mit Anfangswerten erleichtern:

- Geben Sie so viele Anfangswerte vor wie möglich und fixieren Sie diese auch. So können Sie sicherstellen, dass sich der von Ihnen gewünschte Anfangszustand auch einstellt.
- Überlegen Sie bei der Verwendung von Zwangsbedingungen genau, wo Sie Anfangswerte vorgeben möchten. Beachten Sie, dass Sie bei Zwangsbedingungen einige Anfangswerte nicht beliebig wählen können, sondern dass diese den Zwangsbedingungen genügen müssen! Setzen Sie solche Anfangswerte auf *frei* oder fixieren Sie sie bei einem gültigen Wert.
- Vermeiden Sie freie Anfangswerte bei Modellen, die gewöhnlichen Differenzialgleichungen entsprechen. Zwar wird versucht, mit diesen einen konsistenten Anfangszustand zu finden, wenn dies aber scheitert, werden sie als veränderlich angesehen. Der Algorithmus wählt dann den nächstgelegenen konsistenten Anfangszustand aus. Das muss nicht unbedingt der von Ihnen gewünschte sein.

Die Meldung „Es konnten keine konsistenten Anfangswerte berechnen werden!“ kann auch auf einen Fehler im Modell hinweisen. Siehe **Kapitel 11 FAQ's.**

Bevor diese Meldung erscheint, wird wie bei der Berechnung des Gleichgewichtszustandes versucht, konsistente Anfangswerte mit einer Einbettung zu berechnen, falls dies definiert ist (siehe **6.1.2.3)**. Dies gilt jedoch nur für die Anfangswertberechnung zum Startzeitpunkt.

Ein konsistenter Zustand muss auch nach Ereignissen hergestellt werden (Reinitialisierung des Modells).
Dazu wird der gleiche Algorithmus verwendet. Die Informationen darüber, welche Zustandsgrößen fixiert oder frei sind, werden dann allerdings aus der DAE gewonnen.

Als frei gelten dabei im Wesentlichen:

- rein algebraische Zustandsgrößen,
- Zustandsgrößen, die eine Ableitung besitzen, die allerdings an keiner Stelle im Modell verwendet wird,
- die höchsten Ableitungen der Zustandsgrößen,
- die 1. Ableitung, wenn die 2. Ableitung zwar existiert, aber nirgends im Modell verwendet wird.

Alle anderen Zustände sind fixiert und verbleiben auf ihren Werten vor dem Ereignis. Sie können also keinen Sprung aufweisen.
Ein häufiger Modellierungsfehler ist, dass sich bei Ereignissen Zustandsgrößen sprunghaft ändern sollen, die nicht sprungfähig sind, d. h. deren Werte bei der Reinitialisierung vom Algorithmus als fixiert erkannt werden. Beispiele dafür sind sprunghafte Änderungen der Spannung über Kondensatoren, die nur mit un-

endlichen Strömen realisiert werden können. Oder sprunghafte Änderungen des Übersetzungsverhältnisses starrer Übersetzungen, die zu Sprüngen im Winkel oder der Winkelgeschwindigkeit führen, die physikalisch unmöglich sind!

**Hinweis:** Eine Ausnahme bildet das Element Anschlag (Rotatorische bzw. Translatorische Mechanik).

#### 6.1.2.3 Einbettung (Homotopie)

Bei der Anfangswert- oder der Gleichgewichtsberechnung eines Modells kann es vorkommen, dass keine Lösung gefunden wird, obwohl eine existiert. In diesem Fall kann die Einbettung helfen. Eine weitere Anwendung der Einbettung besteht darin, bei der Existenz mehrerer Lösungen die gewünschte zu bestimmen.
Dies kann insbesondere bei stark nichtlinearen Modellen aus der Thermofluidik, MBS-Mechanik oder Elektronik auftreten.

- **Einbettungsvarianten**

SimulationX unterstützt mehrere Varianten der Einbettung:

1. Natürliche Einbettung mit dem Modelica-homotopy-Operator
2. Automatische Einbettung (Fixpunkt-Homotopie, Newton-Homotopie)

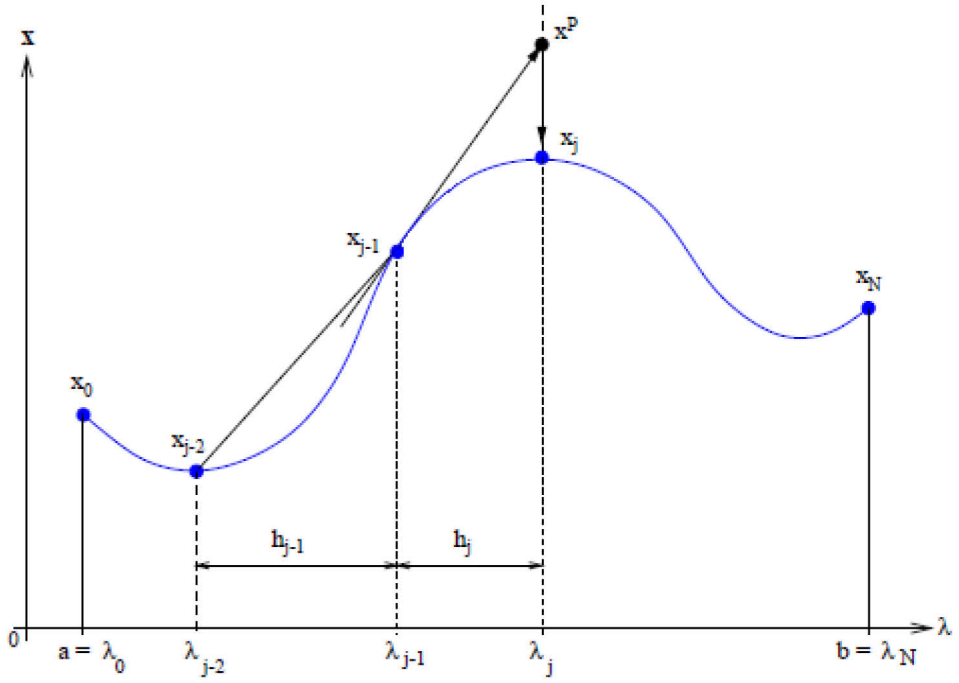
Diese laufen nach dem folgenden Muster ab:

1. Zum Originalgleichungssystem $f(x)=0$ (x bezeichnet die Veränderlichen, d.h. Zustandsgrößen und ggf. deren Ableitungen, während der Anfangswert- oder Gleichgewichtsberechnung) wird der Homotopieparameter $\lambda$ (lambdaHomotopy) derart hinzugefügt, dass das veränderte Gleichungssystem $h(x,\lambda)=0$ eine einfache Lösung für $\lambda=0$ besitzt. Wie dies geschieht, wird weiter unten beschrieben.
2. Mittels eines Kurvenverfolgungsverfahren wird danach durch Vergrößerung von $\lambda$ schrittweise zum Originalproblem übergegangen, wobei $h(x,1)=f(x)$ ist.

- **Funktionsweise des Kurvenverfolgungsverfahrens**

Die Funktionsweise des Kurvenverfolgungsverfahrens kann grob wie folgt beschrieben werden:
Auf Grundlage einer Unterteilung des Parameterintervalls $\lambda \in \Lambda = [0;1]$ in N Teilintervalle werden sukzessive die Punkte der Lösungskurve $\{(x,\lambda):h(x,\lambda)=0\}$ beginnend mit $(x_0,\lambda_0=0)$ bestimmt. Ausgehend von einem bisher berechneten Punkt $(x(\lambda_{j-1}),\lambda_{j-1})$ auf der Lösungskurve erfolgt die Berechnung des nächsten Punktes folgendermaßen:

1. Mit einer Schrittweite $h_j > 0$ wird ein neuer Parameterwert bestimmt: $\lambda_j = \lambda_{j-1} + h_j$.
2. Für den neuen Lösungsvektor $x_j = x(\lambda_j)$ wird ein Prädiktorwert $x^P$ bestimmt. (Prädiktorschritt).
3. Der Prädiktorwert wird mittels eines Newton-ähnlichen Verfahrens iterativ verbessert (Korrektorschritt).




- **Natürliche Einbettung**

Bei einem Modell, für das SimulationX nicht die gewünschte Lösung berechnet, kann der Anwender oftmals seine Systemkenntnis einsetzen.

Kennt der Anwender für modifizierte Parameterwerte eine gute Näherungslösung des Systems, so kann er diese angeben. Von der Näherungslösung aus kann SimulationX eine numerisch exakte Lösung für das modifizierte System finden. Die Vorgabe der bekannten Näherungslösung erfolgt über Startwerte. Die gewünschte Änderung der Parameterwerte der bekannten Lösung zu den gewünschten Werten wird mit Hilfe des Homotopieoperators beschrieben.

In speziellen Anwendungsfällen sind verschiedene Zustände einer Anfangslage für einen geänderten Wert eines Modellparameters bekannt. Mittels des Modelica-homotopy-Operators und des Kurvenverfolgungsverfahren ist es dann möglich, einen Weg des Systems im Zustandsraum zu beschreiben, der diese Anfangslage mit der gewünschten Endlage verbindet.
Dazu werden während der Modellierung Anfangs- oder Parameterwerte durch einen Ausdruck der Form

```
homotopy(actual, simplified)
```

ersetzt. An der Stelle von `actual` steht dann die Berechnungsvorschrift bzw. der Wert, für den das System berechnet werden soll. In `simplified` steht dagegen ein Ausdruck, für den, wie oben beschrieben, eine Anfangs –oder Gleichgewichtslage bekannt ist bzw. berechnet werden kann.

Im Gleichungssystem wird `homotopy(actual, simplified)` durch

$$\lambda \cdot \textbf{actual} + (1-\lambda) \cdot \textbf{simplified} \qquad \textbf{(6.4)}$$

ersetzt und das Kurvenverfolgungsverfahren startet mit $\lambda = 0$.

- **Beispiel**

Beim abgebildeten MKS-Modell sollen verschiedene Gleichgewichtslagen für `springDamper3D1.k=10` bestimmt werden. Für den Wert `springDamper3D1.k=0` sind die stabile und die instabile Gleichgewichtslage bekannt (`revoluteJoint1.phiRel=pi/2` und `phiRel=-pi/2`). Diese Gleichgewichtslagen können als Anfangswerte bei `springDamper3D1.k=0` vorgegeben werden.

**Bild 6.9:** Strukturansicht

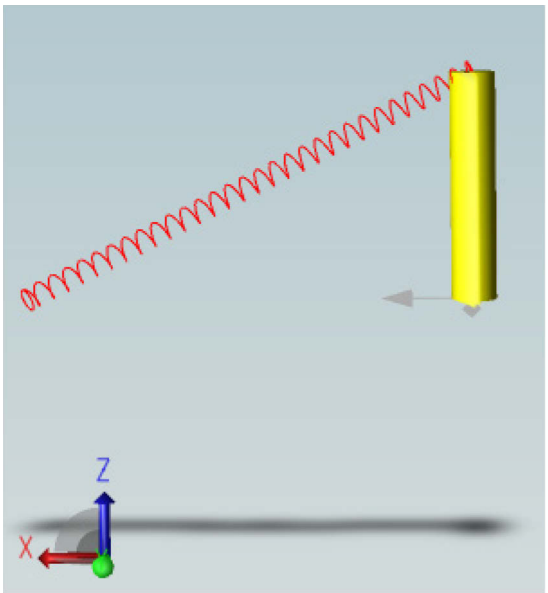



**Bild 6.10:** Instabile Gleichgewichtslage bei k=0, phiRel0=-pi/2

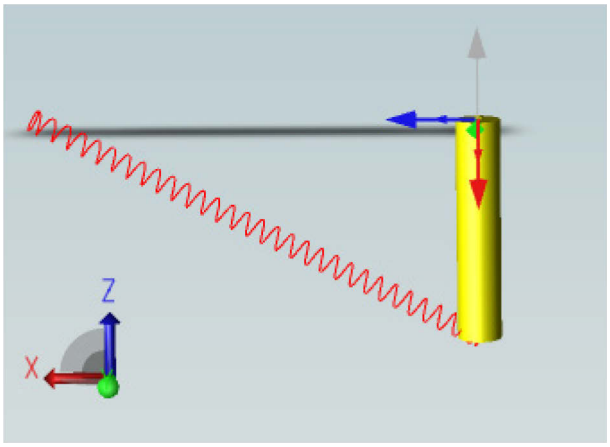



**Bild 6.11:** Stabile Gleichgewichtslage bei k=0, phiRel0=pi/2

Als Wert von springDamper3D1.k tragen wir dazu den Ausdruck homotopy(10,0) ein. Dann wird in Abhängigkeit von revoluteJoint1.phiRel0 die stabile oder instabile Gleichgewichtslage bei springDamper3D1.k=10 berechnet.

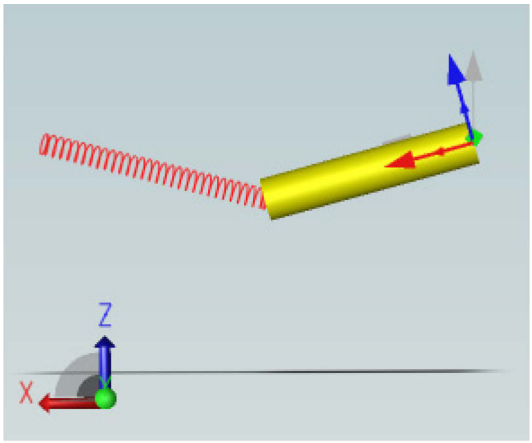



**Bild 6.12:** Stabile Gleichgewichtslage bei k=10, phiRel0=pi/2

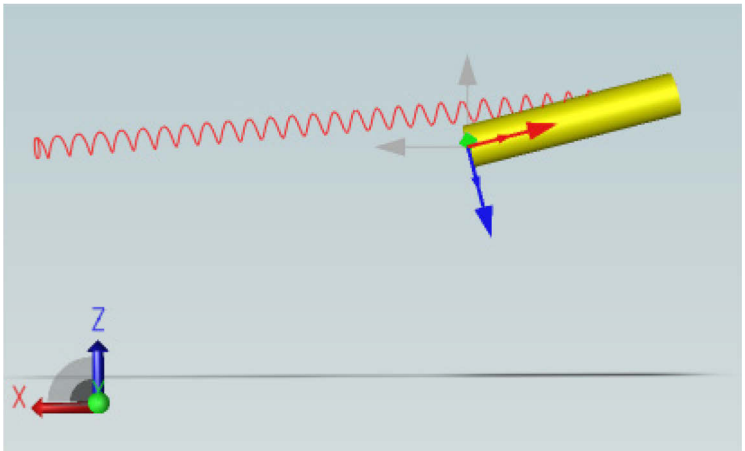



**Bild 6.13:** instabile Gleichgewichtslage bei k=10, phiRel0=-pi/2

Mit Hilfe der Einbettung gelingt somit eine  Festlegung der berechneten Gleichgewichtslage.

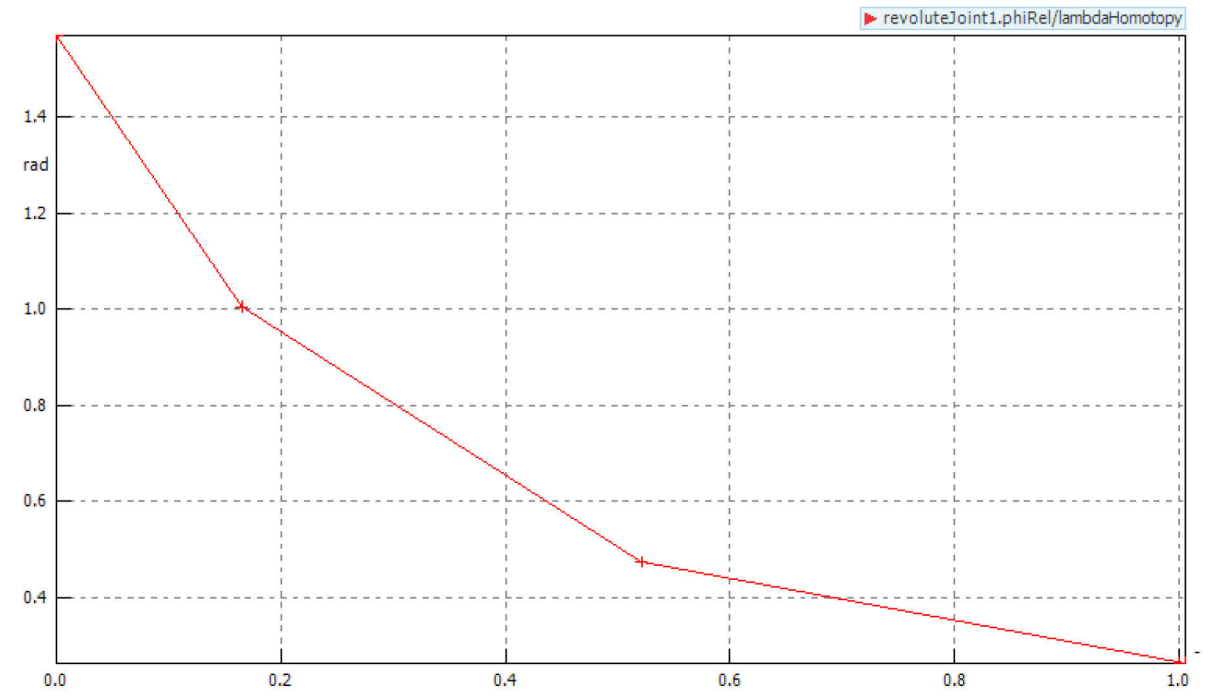



**Bild 6.14:** Einbettungspfad bei der stabilen Gleichgewichtslage: revoluteJoint1.phiRel in Abhängigkeit vom Homotopieparameter lambdaHomotopy

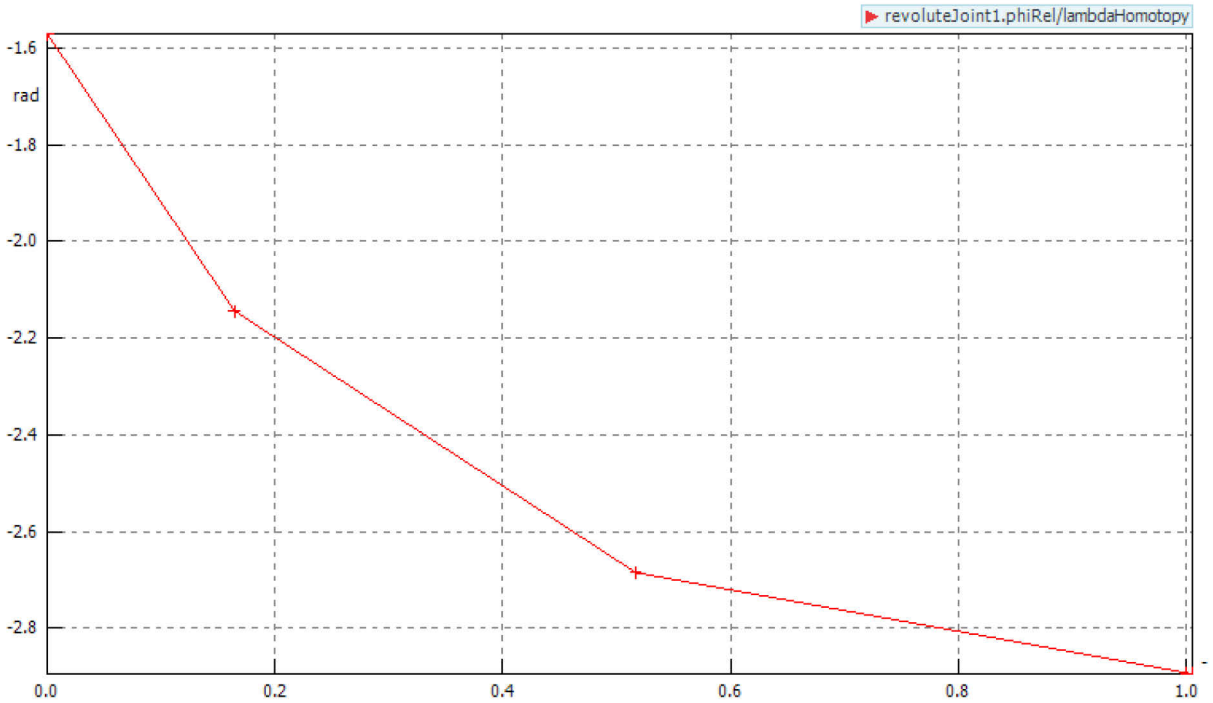



**Bild 6.15:** Einbettungspfad bei der instabilen Gleichgewichtslage: revoluteJoint1.phiRel in Abhängigkeit vom Homotopieparameter lambdaHomotopy

- **Automatische Einbettung**

Ist die Anwendung des Modelica-homotopy-Operators nicht ersichtlich oder kann keine Lösung durch natürliche Einbettung gefunden werden, dann ist die automatische Einbettung eine Alternative. Sie kann mittels des Parameters

**`iHomotopyType`** im Dialog ***Simulationssteuerung/Solver/Weitere Parameter…*** eingeschaltet werden. Für die Werte 2 oder 3 wird entweder eine

a. Fixpunkt-Homotopie

$$g(x,\lambda) = \lambda \cdot h(x,\lambda) + (1-\lambda)\cdot(x - x_0) \qquad \textbf{(6.5)}$$

b. Newton-Homotopie

$$g(x,\lambda) = \lambda \cdot h(x,\lambda) + (1-\lambda)\cdot(h(x,\lambda) - h(x_0,0)) \qquad \textbf{(6.6)}$$

durchgeführt. (Wenn der Modelica-homotopy-Operator nicht im Modell verwendet wird, steht in **(6.5)** und **(6.6)** nur $f(x)$ statt $h(x,\lambda)$.

- **Tracing / Ergebnisfenster**

Zur Nachverfolgung der Berechnungsergebnisse können Ausgabe (siehe Abschnitt **6.1.1.4**) und Ergebnisfenster (siehe Kapitel **7.3**) genutzt werden. So können z.B. Ergebnisgrößen in Abhängigkeit vom Homotopie-Parameter `lambdaHomotopy` angezeigt werden. Mit der Ergebnisfenster-Transformation „Index“ (Ergebnisfenster-Menü: ***Hinzufügen/Transformation/ Index)*** ist es möglich, die protokollierten Werte von nur einer Ergebnisgröße zu betrachten.
Damit während der Einbettung eine Protokollierung erfolgt, ist der Wert des Parameters **`bProtocolValuesForHomotopy`** (Dialog ***Simulationssteuerung/Solver/Weitere Parameter…***) auf true zu setzen.

Im Ausgabebereich kann ein kompletter Überblick über die im Löser verwendeten Variablen gewonnen sowie die Ursache für fehlgeschlagene Kurvenverfolgung ermittelt werden. Es können die Werte der bei der Einbettung berechneten Zustandsgrößen, Newtonkorrekturen, Residuen, Fehler in den Residuen durch Lösung des linearen Systems, Jacobimatrizen und die Nummer des aktuellen Iterationsschrittes (Aktuelle Aktion) sowie die Schrittweite der Einbettung (entspricht der Ausgabe gültiger bzw. ungültiger Schritte bei der transienten Simulation) und die berechneten Tangentialräume ausgegeben werden.

Der Tangentialraum besteht aus allen Vektoren, die bei Multiplikation mit der Jacobi-Matrix den Nullvektor ergeben. Bei einer eindimensionalen Lösungskurve entspricht der Tangentialraum der Tangente der Lösungskurve im aktuellen Berechnungspunkt. Im Schnittpunkt sich schneidender Kurven besteht der Tangentialraum aus den Linearkombinationen der Tangenten beider Kurven im Schnittpunkt.

Die Veränderlichen können sehr unterschiedliche Größenordnungen haben und sich während der Rechnung außerdem stark ändern. Damit sehr große Veränderliche nicht zu sehr kleinen Schritten führen, werden die Veränderlichen

skaliert. Die Skalierungsfaktoren werden ebenfalls bei der Schrittweite der Einbettung ausgegeben.

- **Verfahrensparameter**

Bei fehlgeschlagener Einbettung kann die Änderung von Löserparametern helfen.

Die zur Einbettung gehörenden Parameter (Dialog ***Simulationssteuerung***/***Solver***/***Weitere Parameter…***) haben folgende Bedeutung:

**`iHomotopyType`**

Verwendete Homotopie (Einbettung):

- nur Modelica-Homotopie-Operator (siehe Gleichungssystem **(6.4)** bzw. **(6.5)**)
- Kombination von Modelica-Homotopie-Operator und Fixpunkt-Homotopie (siehe Gleichungssystem **(6.5)**)
- Kombination von Modelica-Homotopie-Operator und Newton-Homotopie (siehe Gleichungssystem **(6.6)**)

**`bUseBorderForHomotopy`**

Möglicherweise treten überbestimmte Systeme auf. D. h., es gibt mehr Gleichungen als Veränderliche. Diese sind häufig nicht lösbar. Das gilt selbst dann, wenn das Originalsystem lösbar ist. In diesem Fall ist es möglich, zusätzliche Variablen einzuführen. Dies entspricht der Einführung zusätzlicher Spalten in der Jacobi-Matrix (und der Korrektur der Ableitung nach $\lambda$). Mit diesem Parameter steuern Sie diese Ränderung der Jacobi-Matrix für überbestimmte Systeme bei der Kurvenverfolgung:

- Wenn es höchstens so viele Gleichungen wie Variablen gibt, ist dieser Parameter bedeutungslos.
- Andernfalls werden weitere zusätzliche Variablen eingeführt. Bei false sind diese immer 0. Das Gleichungssystem ist dasselbe wie ohne diese Variablen. Sollte dieses System aber zu nichtlösbaren linearisierten Systemen führen, kann **`bUseBorderForHomotopy`** auf true gesetzt werden. Dann werden diese zusätzliche Spalten zur Regularisierung der linearisierten Systeme genutzt. Eventuell kann jetzt doch eine Lösung gefunden werden. Dies ist jedoch nicht sicher. Das gilt insbesondere, wenn dann bei der Kurvenverfolgung sich hauptsächlich diese zusätzlichen Variablen ändern.

**dblMaxPredictorDistanceForHomotopy**

Maximale Länge der ersten Newton-Korrektur in einem Homotopieschritt:

- sollte verkleinert werden, wenn mehr Schritte bei der Kurvenverfolgung durchgeführt werden sollen, um die Lösungskurve genauer darzustellen

**dblMaxConvergenceRateForHomotopy**

Größte akzeptierte Konvergenzrate in einem Homotopieschritt; wenn geschätzte Konvergenzrate größer als diese Zahl ist, wird Iteration als nicht konvergent betrachtet:

- bei vielen ungültigen Schritten wegen mangelnder Konvergenz kann diese vergrößert werden (aber kleiner als 1);
- Vergrößerung erlaubt aber langsamere Konvergenz am Anfang eines Homotopieschritts – mehr benötigte Iterationsschritte je Homotopieschritt

**iMaxIterForHomotopy**

Maximale Iterationszahl im Newton-Algorithmus in einem Homotopieschritt:

- bei vielen ungültigen Schritten oder Abbruch wegen zu großer Iterationszahl kann dies vergrößert werden;
- wenn der Homotopieschritt dann trotzdem wegen zu großer Iterationszahl verworfen wird (mangelnde Konvergenz), dauert die Rechnung länger

**dblStepMinForHomotopy**

Minimale Schrittweite bei der Kurvenverfolgung:

- bei kleinerer Schrittweite bricht die Rechnung ab;
- wenn eine kleinere Schrittweite notwendig ist (z.B. starke Nichtlinearitäten), kann sie verkleinert werden

**dblStepMaxForHomotopy**

Maximale Schrittweite bei der Kurvenverfolgung: Schrittweite wird nach oben begrenzt

**dblStepSizeAtBeginOfHomotopy**

(Skalierte) Schrittweite am Anfang der Kurvenverfolgung: mit dieser Schrittweite wird angefangen; sollte zwischen minimaler und maximaler Schrittweite liegen;

- wenn der Einbettungspfad / die Lösungskurve fast linear ist, kann dieser Wert vergrößert werden;

- wenn er zu groß gewählt wurde, gibt es am Anfang der Kurvenverfolgung viele ungültige Schritte

**dblStepSizeReductionForHomotopy**

Verkleinerungsfaktor für die Schrittweite bei der Kurvenverfolgung;

- bei ungültigen Schritten wird die Schrittweite mit diesem Faktor multipliziert, bei erfolgreichen Schritten höchstens durch diesen Faktor geteilt; sollte nicht zu groß sein

**dblAngleMaxForHomotopy**

Maximaler Winkel zwischen den Tangentialräumen aufeinanderfolgender Kurvenverfolgungsschritte in Radiant; kleiner als $\pi/2$ (= 1.57); wenn Winkel zu groß ist, wird Schritt verworfen:

- bei vielen ungültigen Schritten wegen zu großem Winkel kann dieser Wert vergrößert werden;
- bei zu großem Wert ist es möglich, dass wieder zurück gegangen wird; die Hälfte des Wertes ist der angestrebte Wert für den Winkel zwischen den Tangentialräumen – wird bei der Schrittweitensteuerung verwendet

**iMaxPointsForHomotopy**

Maximale Anzahl von Stützpunkten bei der Kurvenverfolgung; bei Überschreitung wird die Kurvenverfolgung abgebrochen;

- wenn mehr Schritte benötigt werden, kann der Wert vergrößert werden;
- wenn durch die Einbettung keine Lösung gefunden werden kann, kann der Wert auch verkleinert werden, um unnötige Rechnungen zu vermeiden

**dblBoundWrongDirForHomotopy**

Untere Schranke für den Einbettungsparameter lambdaHomotopy, damit der Einbettungspfad nicht in die falsche Richtung verfolgt wird (muss negativ sein);

- der Einbettungspfad kann zeitweise auch in Richtung kleiner werdender Werte von lambdaHomotopy gehen;
- der Einbettungspfad kann aber auch von lambdaHomotopy = 0 in Richtung $-\infty$ gehen; in diesem Fall wird bei Unterschreitung des Wertes wieder bei lambdaHomotopy = 0 angefangen und in die entgegengesetzte Richtung gegangen;
- wenn das auch nicht zu einer Lösung führt, wird abgebrochen

**bAlwaysGlobalNewtonForHomotopy**

Nutzt ein globalisiertes Newtonverfahren für alle Punkte bei der Kurvenverfolgung (nicht nur für den Anfangs- und Endpunkt);

- führt zu mehr Jacobi-Matrix-Berechnungen – kann aber das Konvergenzverhalten verbessern

**bUseHomotopyFirst**

Bei true wird zuerst die Einbettung für die Gleichgewichts- oder Anfangswertberechnung genutzt (sonst wird die Einbettung erst nach erfolgloser Berechnung genutzt)?

- Wenn nur mit der Einbettung eine oder die gewünschte Lösung gefunden werden kann, kann dies verwendet werden.

Bei der Anfangswertberechnung zum Startzeitpunkt:

- Lassen Sie false stehen, wenn mit der zusätzlichen Fixierung von ODE-Zustandsgrößen bei der ersten Anfangswertberechnung eines Eventschrittes auch mit Einbettung keine konsistenten Anfangswerte gefunden werden können (überflüssige Einbettungsschritte) oder wenn in späteren Eventschritten zum Startzeitpunkt die Einbettung vermieden werden soll, z.B. um nicht wieder mit lambdaHomotopy = 0 anzufangen.

**bProtocolValuesForHomotopy**

Die Werte auf der Kurve bei der Einbettung protokollieren? Nur dann können sie in Ergebnisfenstern betrachtet werden.

### 6.1.2.4 Zeitschritte

Zur Berechnung über der Zeit stehen mehrere Lösungsverfahren zur Verfügung. Die Auswahl erfolgt im Dialog ***Simulationseinstellungen***, Registerseite ***Solver***. Mehr Informationen zu den Lösern finden Sie im **Kapitel 6.1.3.**

### 6.1.2.5 Ereignisse

Ein Ereignis ist ein Zeitpunkt während der Simulation, an dem sich:

- Zustandsgrößen oder deren Ableitungen sprunghaft ändern (Unstetigkeiten) und/oder
- Änderungen in der Struktur der Gleichungen des Modells ergeben.

Unstetigkeiten können von der Zeit abhängig sein:

- zeitdiskrete Signalglieder,
- Ausdrücke `sample()` oder `delay()`.

Sie können aber auch vom Zustand des Modells beeinflusst werden:
- an den Stützstellen von Kennlinien bei linearer oder Treppen-Interpolation,
- Wechsel zwischen Anlage und Spiel bei Feder-Dämpfer-Spiel-Elementen,
- bedingte Anweisungen wie `if x>10 then 0 else 5`,

Bei Strukturumschaltungen gelten plötzlich andere Gleichungen oder es ändert sich deren Anzahl. Beispiele für Strukturumschaltungen sind:
- Wechsel zwischen Haft- und Gleitreibung in der Mechanik,
- ideale mechanische Anschläge,
- Öffnen und Schließen von idealen elektrischen Schaltern.

Zeitabhängige Ereignisse tragen sich in die Ereigniswarteschlange von SimulationX ein. Die Schrittweitensteuerung des Lösers stellt sicher, dass bis zu diesen Zeitpunkten gerechnet und dann die Ereignisbehandlung gestartet wird.

Zustandsabhängige Unstetigkeiten werden mit Hilfe von sogenannten Zero-functions erkannt. Zum Ereigniszeitpunkt ändern diese ihr Vorzeichen. Kündigt sich ein Vorzeichenwechsel an, wird die Schrittweite so gewählt, dass das Ereignis möglichst genau getroffen wird. Ist innerhalb eines Zeitschrittes ein solcher Vorzeichenwechsel eingetreten, wird versucht, das Ereignis genau anzufahren. Befinden wir uns höchstens mit der minimalen Schrittweite `dtDetect` hinter dem Ereignis, wird die kontinuierliche Integration gestoppt und die Ereignisiteration gestartet. Mit `dtDetect` können Sie also die Genauigkeit einstellen, mit der Ereignisse getroffen werden. Die minimale Rechenschrittweite sollte ein ganzzahliges Vielfaches von `dtDetect` sein.
Die Werte der elementaren Relationen bleiben während der kontinuierlichen Integration konstant, d.h. kurz vor einem Ereignis wird noch mit den alten Werten gerechnet, obwohl sich die Relation schon geändert haben kann. Zum Ereigniszeitpunkt wird die kontinuierliche Integration angehalten und es wird eine Ereignisiteration durchgeführt. Während der Iteration bleibt die Modellzeit konstant. Während der Iteration ändern die Relationen ihre Werte und danach werden für den gleichen Zeitpunkt neue konsistente Anfangswerte bestimmt. So erhält man zwei Werte eines Werteverlaufs an dem Zeitpunkt der Unstetigkeit: einen vor dem Ereignis und einen danach.

Bei der Ereignisiteration spielen die diskreten Variablen eine wesentliche Rolle. Sie können ihre Werte ausschließlich zu Ereigniszeitpunkten ändern. Wenn eine diskrete Variable in Folge des neuen konsistenten Zustandes ihren Wert ändert, wird die Iteration wiederholt. Erst wenn sich alle diskreten Variablen nicht mehr ändern, ist die Ereignisiteration beendet und es wird mit der kontinuierlichen Integration fortgefahren. Es kann Modelle geben, deren diskrete Variablen sich während der Event-Iteration immer wieder aufs Neue ändern. Dieses Verhalten deutet meist auf ein Modellierungsproblem hin. Die Berechnung wird dann nicht

fortgesetzt und die Fehlermeldung „Zyklus in der Event-Iteration“ wird ausgegeben.

Mit dem **`noEvent`-Operator** kann man verhindern, dass bestimmte Relationen ungewollt Ereignisse auslösen.

Beispiel:

```
y:=noEvent(if u>uMax then uMax else if u<uMin then uMin else u);
```

Die Ausdrücke in den `if`-Anweisungen werden direkt ausgewertet, ohne dass die Werte der Relationen konstant gehalten werden und ohne Ereignisse zu erzeugen.

Das ist auch dazu nutzbar, um den Definitionsbereich einer Funktion abzusichern:

```
y:= if noEvent(x>=0) then sqrt(x) else 0;
```

Ohne `noEvent` würde ein Fehler auftreten, weil kurz vor der Iteration x schon negativ ist obwohl x>=0 noch wahr ist.

Der `noEvent` Operator sollte nur nach gründlicher Überlegung verwendet werden. Denn er kann Ereignisse ausblenden, die für eine exakte Berechnung notwendig sind.
Vollkommen legitim und sogar zu empfehlen ist `noEvent` beispielsweise in folgenden Fällen:

- Das Ereignis hat keinen Einfluss auf eine Zustandsgröße. Das ist oft der Fall, wenn Modellteile nur für die Auswertung bestimmt sind. Die kontinuierliche Integration kann dann ohne Probleme über diesen Zeitpunkt hinwegrechnen. Es muss allerdings mit einem gewissen Fehler in der Relation gerechnet werden, da nicht sichergestellt ist, dass der Umschaltzeitpunkt exakt getroffen wird.
- Zum Abfangen von Werten außerhalb des Definitionsbereiches von Operationen und Funktionen (z. B. Division durch Null, `tan`, `arcsin`).
- Das Ereignis ändert den Anstieg eines Verlaufes nur wenig (z. B. lineare Fortsetzung der e-Funktion).
- Sie verwenden eine Relation, um einen stetigen Verlauf der höchsten relevanten Ableitung einer Zustandsgröße zu erzeugen.

Den letzten Punkt möchten wir an einem Beispiel näher erläutern:

Als Sollwertverläufe für Positioniervorgänge werden gern Abschnitte von sin² Funktionen verwendet.

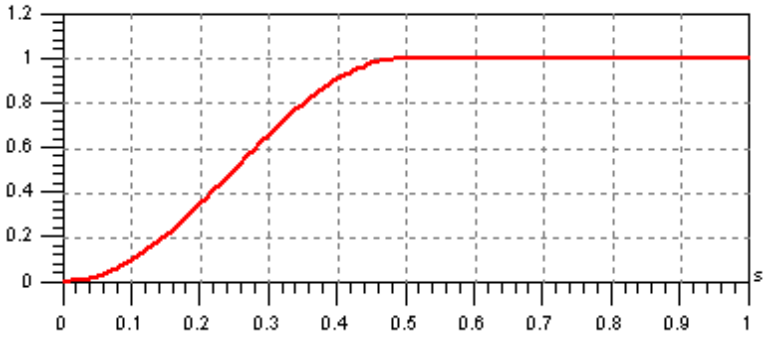



**Bild 6.16:** sin² Abschnitt als Sollwert für einen Positioniervorgang

Ein solches Signal können Sie mit dem Ausdruck

```
if t<=0.5 then sin(pi*t)^2 else 1
```

erzeugen. Für die Relation wird automatisch ein Ereignis bei 0.5 s erzeugt, obwohl dort gar keine Unstetigkeit auftritt. Der `noEvent`-Operator:

```
if noEvent(t<=0.5) then sin(pi*t)^2 else 1
```

führt hier zum Ausbleiben des Ereignisses und damit zu einer Beschleunigung der Rechnung, ohne dass damit Genauigkeitseinbußen verbunden sind.

Die noEvent-Eigenschaft betrifft alle Relationen innerhalb der noEvent-Funktion. Alle Anweisungen innerhalb einer when-Anweisung werden implizit so behandelt, als würden sie in einer noEvent-Funktion stehen. Das ist keine Einschränkung, da der Körper einer when-Anweisung während der normalen Integration nicht ausgeführt wird.

### 6.1.3 Verfügbare Löser

Zur Berechnung der transienten Simulation stehen folgende Löser zur Verfügung:

- BDF-Verfahren (**B**ackward **D**ifferential **F**ormulas – rückwärtige Differenzenquotienten). Dieser Löser entspricht einer Erweiterung von DASSL **[1]**.
- MEBDF-Verfahren (**M**odified **E**nhanced BDF). Das ist ein Löser, der auf Jeff Cash **[2]** zurückgeht.
- Löser mit konstanter Schrittweite (externer Löser).
- **CVODE** - Verfahren (externer Löser). Löser von der SUNDIALS Suite **[5]** (siehe: www.llnl.gov/casc/sundials/)

Die Auswahl des Verfahrens erfolgt im Dialog ***Simulationseinstellungen***, Registerseite ***Solver***.

Externe Löser nutzen eine kompilierte Form des Modellalgorithmus für eine Erhöhung der Rechengeschwindigkeit. SimulationX unterstützt die C-Compiler des Visual Studios (Version 6 oder höher; wie auch die freie Express Edition).

Einer dieser Compiler muss auf dem Computer installiert sein und kann über die ***Simulationseinstellungen***, Register ***Solver*** Gruppe ***Parameter*** ausgewählt werden.

### 6.1.3.1 BDF- und MEBDF-Verfahren

Beide Verfahren sind gut für folgende Modelle geeignet:

- Steife und nicht-steife Modelle (das System hat Eigenfrequenzen und/oder Zeitkonstanten in einem weiten Bereich).
- Modelle mit Unstetigkeiten. Die Ereignisbehandlung ist in **Kapitel 6.1.2.5** dargestellt.
- Das MEBDF-Verfahren ist speziell für Modelle mit höherem Index gestaltet (z.B. MKS-Modelle mit geschlossenen Ketten). Die Ergebnisse für solche Modelle sind exakter und die Simulationsgeschwindigkeit oft schneller.

Diese Verfahren nutzen die SimulationX-interne Darstellung des Modells für die Residuenberechnung entsprechend **(6.1)**. Es wird keine Kompilierung von Modellalgorithmen durchgeführt. Daher ist die Vorbereitung der Berechnung schneller, aber die RHS-Berechnung etwas langsamer als bei der Nutzung eines externen Lösers.

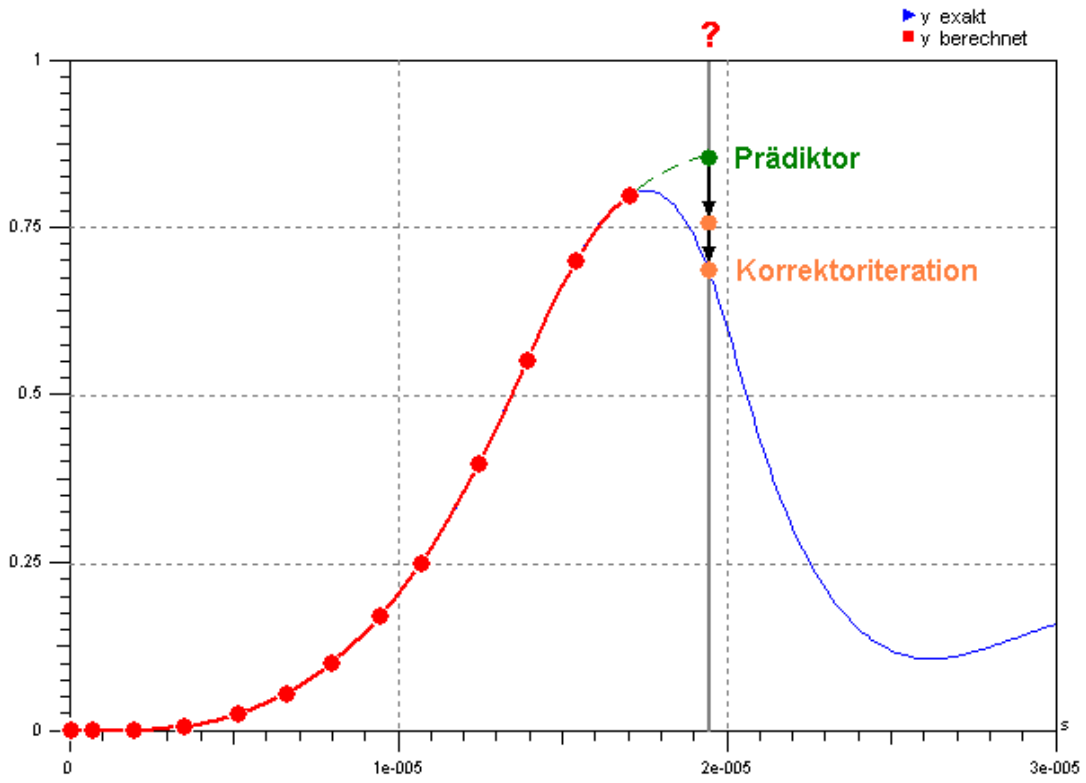



**Bild 6.17:** Prädiktor-Korrektor-Verfahren

Beide Algorithmen sind Prädiktor-Korrektor-Verfahren mit automatischer Steuerung von Schrittweite und Ordnung k. Bei dieser Art der impliziten Mehrschrittverfahren wird für jede Zustandsgröße aus k+1 bereits berechneten Werten der aktuelle Wert extrapoliert (Prädiktor). Dieser wird iterativ so lange verbessert,

bis die geforderte Genauigkeit erreicht ist und bestimmte Konvergenzkriterien erfüllt sind (Korrektor).

Wird der Korrektorwert akzeptiert, kann zum nächsten Zeitschritt übergegangen werden. Dazu wird aus dem Iterationsverlauf eine neue Schrittweite und/oder Ordnung berechnet.

Stellt sich nach einer bestimmten Anzahl von Iterationen kein Korrektorwert ein, der die Genauigkeitsforderungen erfüllt, werden Schrittweite und/oder Ordnung verkleinert und der komplette Zeitschritt wiederholt.

Die **Maximale Verfahrensordnung** kann in der *Simulationssteuerung/Solver* geändert werden. Sie gibt an, wie viele zurückliegende Zeitschritte in die Berechnung eingehen. Während einer Simulationsrechnung erfolgt eine automatische Steuerung der Verfahrensordnung. Wird 1 als maximale Verfahrensordnung gewählt, so entspricht das Verfahren dem impliziten Euler-Verfahren.

---

Ist man bei der minimalen Rechenschrittweite angelangt und ist immer noch keine Konvergenz hergestellt, erhält man einen entsprechenden Hinweis. Sie sollten dann immer zuerst die minimale Rechenschrittweite verkleinern. Erst wenn Sie damit keinen Erfolg haben, können die Fehlerschranken aufgeweicht werden. Eine Verringerung der maximalen Ordnung ist nur bei stark nichtlinearen Modellen, bei denen die Verfahren mit den Standardeinstellungen langsam rechnen oder gar keine Lösung finden, erforderlich. Die automatische Steuerung sorgt normalerweise dafür, dass immer mit der optimalen Ordnung gerechnet wird.

---

Bei jeder Korrektoriteration wird ein lineares Gleichungssystem gelöst:

$$res = J \cdot (x_{i-1} - x_i) \qquad \textbf{(6.7)}$$

mit:

- $res$ … Residuen
- $J$ … Jacobimatrix
- $x$ … Zustandsgrößen
- $i$ … Iterationszahl

Dazu stehen folgende Verfahren bereit:

- Solver für schwach besetzte Matrizen,
- Gauß-Verfahren,
- Skaliertes Gauß-Verfahren.

Im Dialog ***Simulation/Einstellungen*** Registerseite ***Solver*** können Sie eines der Verfahren auswählen.

Standardmäßig ist der **Solver für schwach besetzte Matrizen** eingestellt. Dabei wird ausgenutzt, dass beim zu lösenden Gleichungssystem **zumeist nur lokale Abhängigkeiten** bestehen. Zum Beispiel hängen bei einem Antriebsstrang die Zustandsgrößen am Motor nur indirekt über weitere Zustandsgrößen von denen am Rad ab. Dadurch entstehen sehr viele Nulleinträge in der Jacobimatrix, die bei der Lösung nicht berücksichtigt werden müssen. Das führt zu einer wesentlichen Beschleunigung der Rechnung im Vergleich zum Gauß-Verfahren.

Das **Gauß-Verfahren** ist für **extrem stark vernetzte Modelle** zu empfehlen, bei denen keine dünn besetzten Jacobimatrizen auftreten. Das ist jedoch sehr selten der Fall.

Bei Modellen mit stark differierenden Zeitkonstanten bzw. Größenordnungen der Änderungen der Zustandsgrößen kann es zu schlecht konditionierten Jacobimatrizen kommen. Der Solver für schwach besetzte Matrizen und das Gauß-Verfahren sind dann nicht mehr in der Lage, eine genaue Lösung zu berechnen. Das kann zu schlechter Konvergenz in den Korrektorschritten (viele Korrektoriterationen, kleine Schrittweiten) oder gar zum Abbruch der Simulationsrechnung führen. Das **skalierte Gauß-Verfahren** versucht, die Konditionszahl zu verbessern und damit eine exaktere Lösung zu erreichen.

Für jedes Modell gibt es die Solver-Debug-Info **`solverInfo`**. Beim BDF- und MEBDF-Verfahren hat diese Information neun Komponenten, die im Einzelnen folgende Bedeutung haben:

1. Verfahrensordnung
2. Anzahl der Prädiktor-Korrektor-Iterationen.
3. Anzahl der verworfenen Schritte (ungültige Schritte oder im Zusammenhang mit einer Unstetigkeit verworfene) zwischen vorhergehendem und aktuellem gültigen Schritt
4. Anzahl der ungültigen Schritte mit zu großem Fehlerschätzer zwischen vorhergehendem und aktuellem gültigen Schritt
5. Anzahl der (sonstigen) ungültigen Schritte wegen mangelnder Konvergenz zwischen vorhergehendem und aktuellem gültigen Schritt
6. (Vorgeschlagene) Änderung der Zeitschrittweite (im Zusammenhang mit Unstetigkeiten und verworfenen Schritten interessant)
7. Anzahl der Residuenberechnungen, aber nicht nur in gültigen, sondern auch in ungültigen Schritten, jedoch nicht in Event-Iterationen
8. Markierung für Unstetigkeiten (wird am Ende der Event-Iteration und im folgenden akzeptierten Schritt angezeigt)
9. Anzahl der Jacobimatrixberechnungen

### 6.1.3.2 Löser mit konstanter Schrittweite (externer Löser)

Dieser Löser wurde in erster Linie in SimulationX integriert, um Modelle, die zum Export auf Echtzeitplattformen vorgesehen sind, auf **Echtzeitfähigkeit** zu testen. Er beinhaltet verschiedene Verfahren zur Lösung expliziter Differentialgleichungen (ODE) entsprechend **(6.2)**, welche auch wieder beim Codeexport-

Projekttyp *Ausführbares Modell* und den unterstützten Echtzeitsimulations-umgebungen ProSys-RT, Scale-RT, dSPACE oder NI VeriStand zur Verfügung stehen.

In SimulationX kann somit vorab getestet werden, mit welcher konstanten Rechenschrittweite die Modelle stabil berechnet werden können.

Die einzige Größe im Dialog ***Simulationssteuerung***, Register ***Allgemein,*** die in die bei Berechnung der Zustandsgrößen eine Rolle spielt, ist die ***Minimale Rechenschrittweite*** `dtMin`. Wie der Name des Lösers verrät, bleibt diese Größe konstant während der gesamten Simulation.

Aufgrund der konstanten Rechenschrittweite können Ereignisse nicht direkt angefahren werden, wodurch bei großen Schrittweiten fehlerhafte Ergebnisse entstehen können. Dabei wird einmal pro Zeitschritt geprüft, ob sich das Vorzeichen einer Zerofunction geändert hat. Ist dies der Fall, findet im Anschluss eine Ereignisiteration statt.

Im Dialog ***Simulationssteuerung***, Register ***Solver*** kann man ein „***Spezielles Integrationsverfahren***“ auswählen. Zur Verfügung stehen fünf explizite Verfahren verschiedener Ordnung. Sie benötigen unterschiedlich viele RHS-Berechnungen (RHS) pro Zeitschritt, die wir mit FE bezeichnen:

0 : Euler-Vorwärts-Verfahren (1 FE, Ordnung 1),

1 : ITI-Standard (1 FE, Ordnung 2),

2 : Verfahren von Heun (2 FE, Ordnung 2),

3 : RKF23 (3 FE, Ordnung 3),

4 : DOPRI5 (6 FE, Ordnung 5).

Mit steigender Anzahl an RHS-Berechnungen pro Zeitschritt steigt auch die Rechenzeit (bei konstanter Schrittweite). Dagegen wird das Stabilitätsgebiet (siehe **Bild 6.18**) größer und die Ergebnisse sollten genauer werden. Da sich dies bisher nicht immer bewahrheitet hat, sollte man die verschiedenen Verfahren mit unterschiedlichen Schrittweiten `dtMin` im Bereich `1.E-003` bis `1.E-006` ausprobieren. Wir empfehlen, mit den Verfahren *Euler -Vorwärts* oder *ITI Standard* (Standardeinstellung) und `dtMin=1.E-005` zu beginnen.

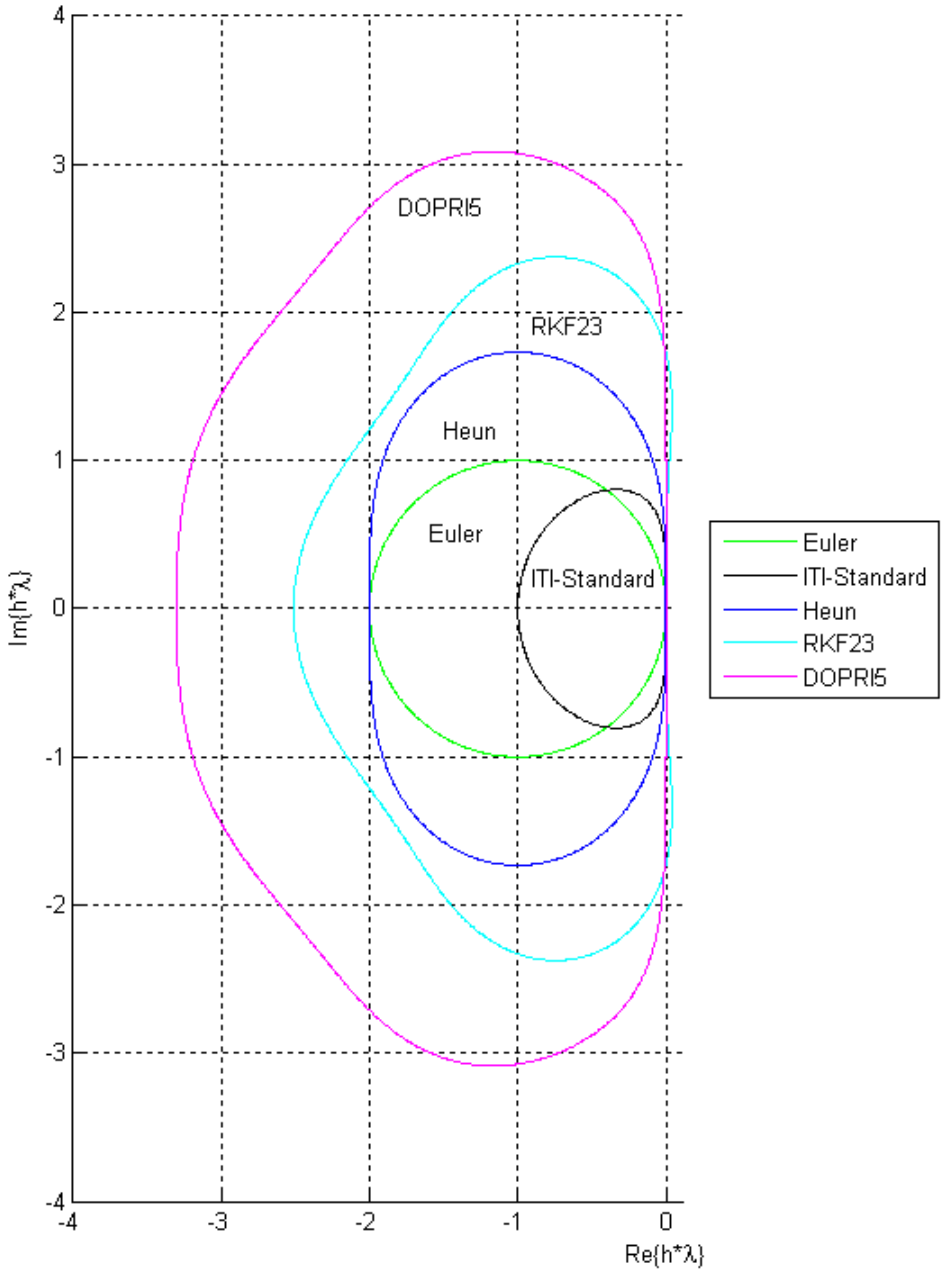



**Bild 6.18:** Stabilitätsgebiete

Unter der Schaltfläche ***Weitere Parameter...*** können weiterhin die folgenden Lösereinstellungen geändert werden:

- Weisen Fehlermeldungen bei einem Abbruch der Simulation auf implizite Blöcke hin, dann kann man versuchen, die Rechnung mit geänderten Einstellungen der Blockgleichungslöser-Parameter `blockTol` (Toleranz) oder `blockLinSolv` (Linearer Gleichungslöser) zu wiederholen. (Mehr Hinweise dazu finden Sie in Abschnitt **11.3.2**)
- Unterschreitet der Wert von Zustandsgrößen ein vorgegebenes Minimum, wird die Simulation im Allgemeinen abgebrochen und muss mit kleinerer Rechenschrittweite wiederholt werden. Setzt man `iStateMinMax` auf den

Wert 1, werden bei den Verfahren Euler-Vorwärts, ITI Standard oder Heun diese Bereichsverletzungen abgefangen und somit eine Verbesserung der Echtzeitfähigkeit erreicht. (Dabei müssen die Minima konstant sein.)

- Durch Setzen der Compiler-Optimierung `bOptimization` auf `true` kann eine Erhöhung der Rechengeschwindigkeit erreicht werden (vgl. CVODE).
- Bei der Kompilierung kann eine Objektdatei bis zu 65.536 ($2^{16}$) adressierbare Abschnitte enthalten. Setzt man `bBigObj` auf `true` wird diese Adressenkapazität auf 4.294.967.296 ($2^{32}$) erhöht. Hinweis: Eine mit `/bigobj` erstellte OBJ-Datei kann nur von einem Linker verwendet werden, der in Visual C++ 2005 (oder späteren Versionen) enthalten ist.

#### 6.1.3.3 CVODE (externer Löser)

Dieser Löser kann für folgende Modelle angewendet werden:

- Steife oder nichtsteife Modelle (Systeme mit Eigenfrequenzen und/oder Zeitkonstanten in einem großen Bereich).
- Modelle mit relativ wenigen Unstetigkeiten.
- Große Modelle werden meistens schneller berechnet als mit dem (ME)BDF-Verfahren. Das ist auf die Kompilierung der Modellalgorithmen zurückzuführen.

Auf Grund des Kompilierungsprozesses dauert die Vorbereitung der Berechnung etwas länger als die bei Nutzung des (ME)BDF-Verfahrens. Deshalb wird empfohlen, diesen externen Löser für Modelle mit fixierten Strukturen und variablen Parametern (z.B. in Variantenrechnungen) zu nutzen.

Für die externen Löser müssen die gleichen Einschränkungen wie für den C-Code - Export berücksichtigt werden. Weitere Details Ihrer SimulationX-Version finden Sie in den aktuellen Release Notes.
Der Löser ist Teil der SUNDIALS Suite (siehe: www.llnl.gov/casc/sundials/).
Bitte beachten Sie die Lizenzbedingungen des ursprünglichen Codes:

```
Copyright (c) 2002, The Regents of the University of California.
Produced at the Lawrence Livermore National Laboratory.

Written by:
S.D. Cohen, A.C. Hindmarsh, R. Serban, D. Shumaker, and
A.G. Taylor.

UCRL-CODE-155951    (CVODE)
UCRL-CODE-155950    (CVODES)
UCRL-CODE-155952    (IDA)
UCRL-CODE-155953    (KINSOL)
All rights reserved.
THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND
CONTRIBUTORS"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
```

INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE REGENTS OF THE UNIVERSITY OF
CALIFORNIA, THE U.S. DEPARTMENT OF ENERGY OR CONTRIBUTORS BE
LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY,
OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA,
OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR
TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT
OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY
OF SUCH DAMAGE.

Der Löser wurde von ITI leicht geändert in SimulationX integriert.

CVODE ist ein Verfahren zur Lösung expliziter Differentialgleichungen (ODE) entsprechend **(6.2)**, welches entweder ein BDF- oder ein Adams-Moulton-Verfahren für die Integration anwendet (Parameter `bStiff`). Unter der Schaltfläche ***Weitere Parameter...*** im Dialog ***Simulationssteuerung***, Register ***Solver*** kann eines der beiden Verfahren ausgewählt werden, wobei die BDF-Variante Standardeinstellung ist. Das Adams-Verfahren kann für nichtsteife Modelle effizienter sein.

Wie beim (ME)BDF-Löser kann auch hier die maximale Verfahrensordnung der BDF-Integrationsvariante verkleinert werden. Der Standardwert beträgt 5. Eine Verkleinerung führt vor allem dann zu Rechenzeitgewinn, wenn der Löser über die gesamte Simulationsdauer vorrangig mit einer kleinen Verfahrensordnung arbeitet. Letzteres kann man leicht mit der *Solver Debug Info* `solverInfo[1]` herausfinden (Erklärung siehe unten).

Die maximale Ordnung für die Adamsvariante beträgt 12. Wenn `bStiff`=false gesetzt wurde, wird dieser Wert automatisch vom Löser gesetzt und kann nicht verändert werden.
Durch eine effizientere Jacobimatrixberechnung gelingt es vor allem bei Modellen mit dünn besetzten Jacobimatrizen, viele RHS-Berechnungen einzusparen. Dies führt zu einer zum Teil deutlichen Verringerung der Rechenzeit.
Die ***Effiziente Jacobimatrixberechnung*** kann im Dialog ***Simulationssteuerung***, Register ***Solver*** unter *Lineares Gleichungssystem* in Verbindung mit einem von (ME)BDF her bekannten Gauß-Verfahren oder Sparse Matrix Löser (Standard-einstellung) zur Lösung des linearen Gleichungssystems eingestellt oder unter gewissen Umständen deaktiviert (*Ohne effiziente Jacobimatrixberechnung*) werden.

Unter der Schaltfläche ***Weitere Parameter...*** können weiterhin die folgenden Lösereinstellungen geändert werden:

- Weisen Fehlermeldungen bei einem Abbruch der Simulation auf implizite Blöcke hin, dann kann man versuchen, die Rechnung mit geänderten

Einstellungen der Blockgleichungslöser-Parameter `blockTol` (Toleranz) oder `blockLinSolv` (linearer Gleichungslöser) zu wiederholen. (Mehr Hinweise dazu finden Sie in Abschnitt **11.3.2**)

- Ereignisse werden wie in Kapitel **6.1.2.5** beschrieben behandelt. Die exakte Lokalisierung der Nullstellen von Zerofunctions kann über den Parameter `bZeros` ausgeschaltet werden. Das führt manchmal zu einer höheren Simulationsgeschwindigkeit, allerdings verbunden mit einem Verlust an Genauigkeit. Zustandsereignisse werden in diesem Fall später erkannt.
- Die Algorithmen zur Lokalisierung von Ereignissen können eine Hysterese berücksichtigen. Diese Feature wird durch den Parameter `bHysterese` eingeschaltet. In diesem Fall werden Ereignisse erst dann detektiert, wenn die zugehörige Zero Function um einen bestimmten Betrag (Parameter `Epsilon`) von null verschieden ist. Ereignisse werden dadurch etwas später detektiert. Für bestimmte Modelle lassen sich damit Zyklen in der Event Iteration bzw. das wiederholte Auffinden ein und desselben Ereignisses (discontinuity sticking) vermeiden. `Epsilon` sollte nicht größer als $10^{-8}$ gewählt werden.
- Die minimale Rechenschrittweite dtMin wird bei CVODE nicht nach unten begrenzt. Über den Parameter `bLimitdtMin` kann dtMin als untere Schranke gesetzt werden.
- Durch Setzen des Parameters `bOptimization` auf true kann die Rechengeschwindigkeit erhöht werden. Es wird dann die Optimierungs-einstellung /O2 des Compilers eingeschaltet, die allerdings zu signifikant längeren Übersetzungszeiten führen kann.
- Bei der Kompilierung kann eine Objektdatei bis zu 65.536 (216) adressierbare Abschnitte enthalten. Setzt man `bBigObj` auf true wird diese Adressenkapazität auf 4.294.967.296 (232) erhöht. Hinweis: Eine mit /bigobj erstellte OBJ-Datei kann nur von einem Linker verwendet werden, der in Visual C++ 2005 (oder späteren Versionen) enthalten ist.

Mehr Details über die verwendeten Algorithmen in CVODE finden Sie in **[5]** und den angegebenen Verweisen.

Wie beim BDF- und MEBDF-Verfahren gibt es die Solver-Debug-Info **`solverInfo`**. Für CVODE beinhaltet diese neun Komponenten, die im Einzelnen die folgende Bedeutung haben:

1. Verfahrensordnung (zur Zeit `t<=tStop`)
2. Änderungsfaktor der Rechenschrittweite
3. Anzahl an RHS-Berechnungen (bis zur Zeit `t`)
4. Anzahl an Iterationsschritten während einer Ereignisiteration (vorausgesetzt Protokollierung von Ereignissen aktiviert)
5. Anzahl an Jacobimatrixberechnungen (bis zur Zeit `t`)

6. Anzahl an Codeerzeugungen (bis zur Zeit `t`) (nur bei `bSparseMat`=true, sonst 0)
7. Anzahl an LU-Zerlegungen (bis zur Zeit `t`)
8. Anzahl an Lösungen des linearen Gleichungssystems (bis zur Zeit `t`)
9. Anzahl an verworfenen Schritte (zur Zeit `t`)

Die Gesamtzahl an Codeerzeugungen, LU-Zerlegungen und Lösungen des linearen Gleichungssystems findet man auch noch einmal nach Simulationsende in der Ausgabe.

## 6.1.4 Performanceanalyse der Simulation

Mittels der Performanceanalyse kann man beobachten, welche Zustandsgrößen eine Vergrößerung der Rechenschrittweite (in einem Zeitintervall oder über die gesamte Simulationszeit betrachtet) am stärksten verhindern.

### 6.1.4.1 Ausgabe

Dabei schätzt der Löser in jedem Zeitschritt den Rechenfehler für jede Zustandsgröße ab. Diese Werte werden danach gewichtet und ihre Summe über alle Zeitschritte in der Spalte *Einfluss* angezeigt.
Beim Löser mit konstanter Schrittweite entspricht diese Fehlerabschätzung der Differenz von zwei numerischen Approximationen unterschiedlicher Ordnung, die gleichzeitig in jedem Zeitschritt berechnet werden. Bei CVODE wird der Fehler während der Berechnung des nichtlinearen Gleichungssystems abgeschätzt.

### 6.1.4.2 Interpretation der Ergebnisse

Somit kann die Performanceanalyse verwendet werden, um kritische, d. h. steife und besonders nichtlineare Modellteile zu identifizieren. Ist ein Modell zum Beispiel noch nicht echtzeitfähig, kann es auf dieser Grundlage reduziert werden.

### 6.1.4.3 Bedienung

Die Performanceanalyse wird über das Menü ***Simulation/Performanceanalyse*** aufgerufen und muss vor der Benutzung (einmalig) aktiviert werden (Checkbox „Performanceanalyse aktivieren“). Die Aktivierung oder Deaktivierung kann auch während der Simulation erfolgen. Somit ist eine Analyse der Ergebnisse auch in Zeitintervallen möglich.

Ist die Checkbox „***Werte über der Simulationszeit aufzeichnen***“ aktiviert, werden alle berechneten Werte einschließlich der dazugehörigen Simulationszeit aufgezeichnet. Mittels des Schiebereglers kann man sich dann innerhalb des Aufzeichnungszeitraumes bewegen und die Änderungen genau studieren. Sonst werden nur die Werte zur aktuellen Simulationszeit (Simulation angehalten / beendet) angezeigt. Dazu muss die Schaltfläche „***Aktualisieren***“ betätigt oder die Checkbox „***Liste automatisch aktualisieren***“ aktiviert werden.

Durch Anklicken der Spaltenköpfe ist es möglich, die Liste nach der Nummer oder dem Namen der Zustandsgröße oder den berechneten Zahlenwerten auf- oder absteigend zu ordnen. Weiterhin wird durch Anklicken einer Zustandsgröße die zugehörige Komponente in Strukturansicht und Modellexplorer markiert.

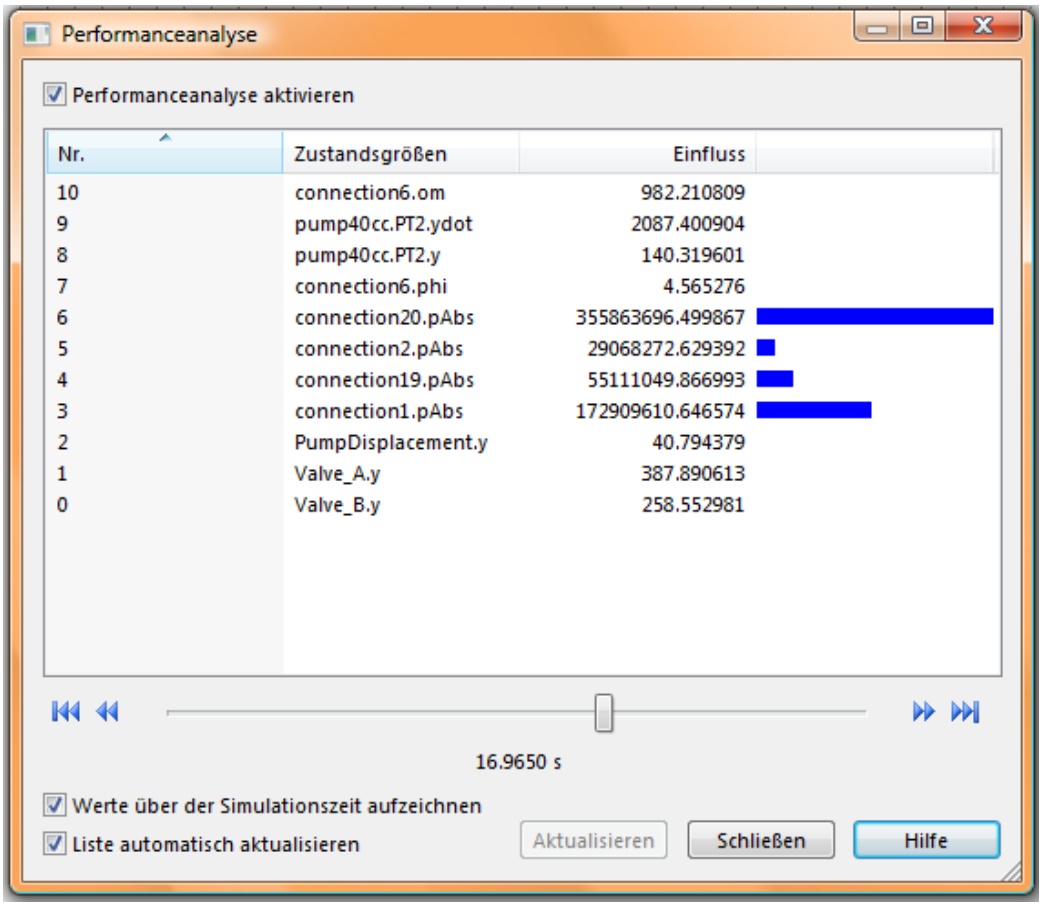



**Bild 6.19:** Diagrammansicht der Performanceanalyse

## 6.2 Animation der Berechnungsergebnisse

Die Berechnungsergebnisse einer transienten Simulation können aufgezeichnet werden und in Echtzeit als Animation wiedergegeben werden. Während einer laufenden Animation lassen sich innerhalb der 3D-Ansicht Bewegungsabläufe überprüfen.

In Ergebnisfenstern (siehe **Bild 6.20**) mit Y(t) bzw. Y(X)-Darstellung wird ein Fadenkreuz eingeblendet, das dem Verlauf der dargestellten Ergebniskurve folgt.

Im Ergebnisfenster mit Balkendarstellung (siehe **Bild 6.21**) wird die Balkengröße entsprechend dem Wert der zugehörigen Ergebnisgröße dynamisch verändert.

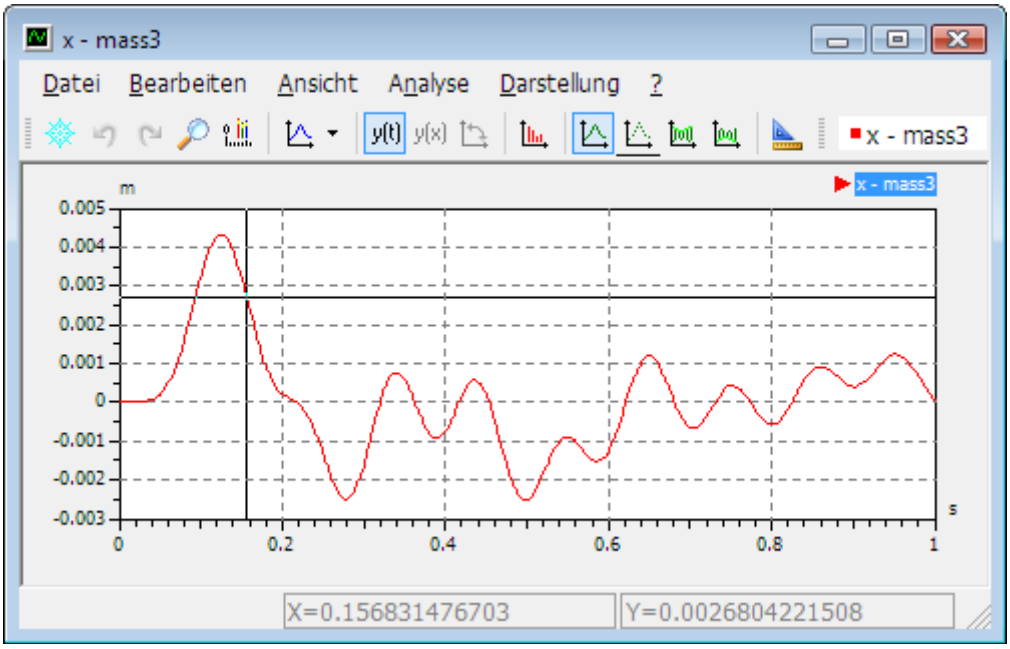



**Bild 6.20:** Animiertes Fadenkreuz in einem Ergebnisfenster mit Y(t)-Verlauf

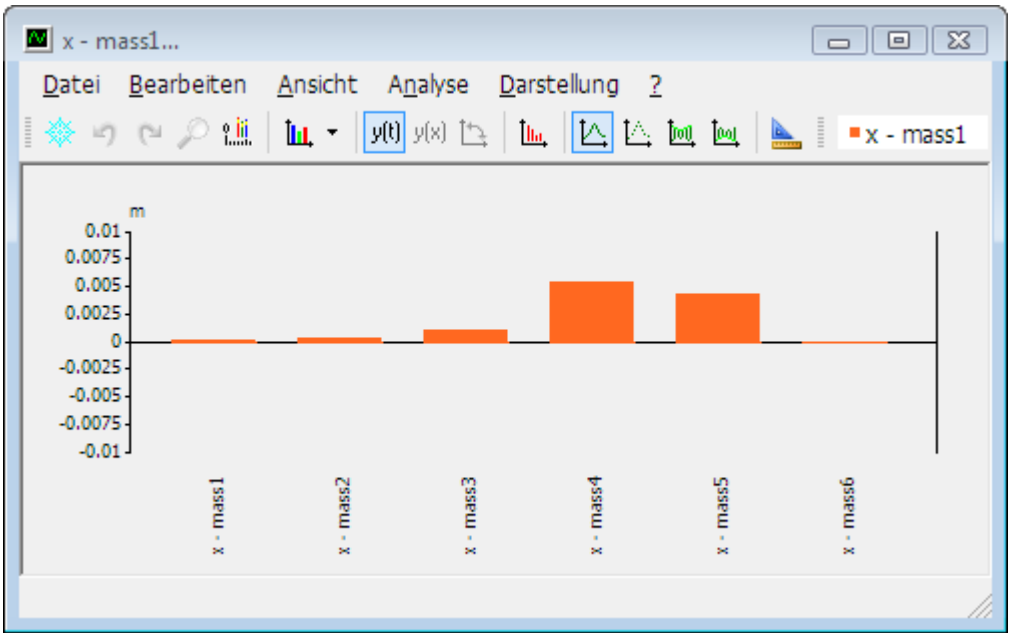



**Bild 6.21:** Animiertes Balkendiagramm in einem Ergebnisfenster

Innerhalb der Strukturansicht gestatten Momentananzeigen sowie Elementsymbole und Formen mit dynamischen Darstellungseigenschaften eine zusätzliche Visualisierung der Berechnungsergebnisse.

**Bild 6.22:** Animierte Modelica Grafik

Mittels der Animation können Zustände von Verbindungen entsprechend vorgegebener Werte eingefärbt werden. Somit ist eine Beobachtung der Ergebnisvariablen von Verbindungen (z.B. Temperatur) ohne Ergebnisfenster während der Simulation möglich.

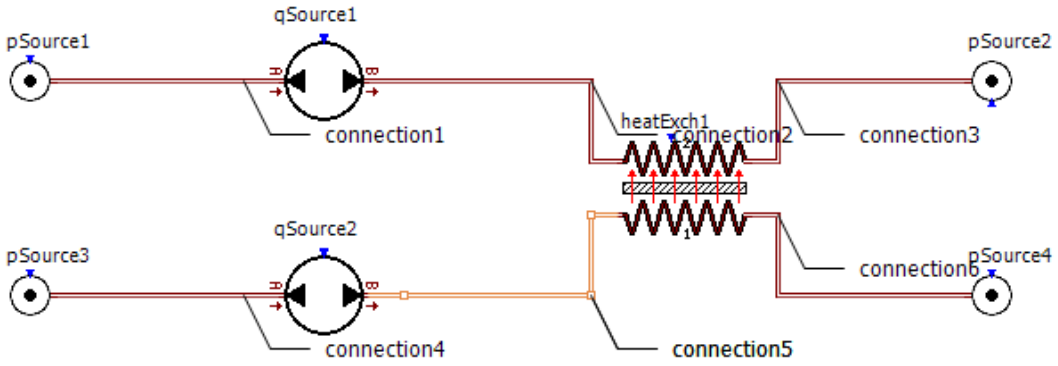



**Bild 6.23:** Animierte Verbindung (connection 5)

Im Eigenschaftsdialog der Verbindungen kann unter ***Visualisierung*** eine Ergebnisvariable ausgewählt werden, die entsprechend der Grenzwerte farblich unterschiedlich dargestellt wird.

Für das Einfärben stehen folgende Modi zur Verfügung:

- Farbverlauf zwischen Minimum und Maximum
- Diskrete Farben für zulässige und unzulässige Bereiche

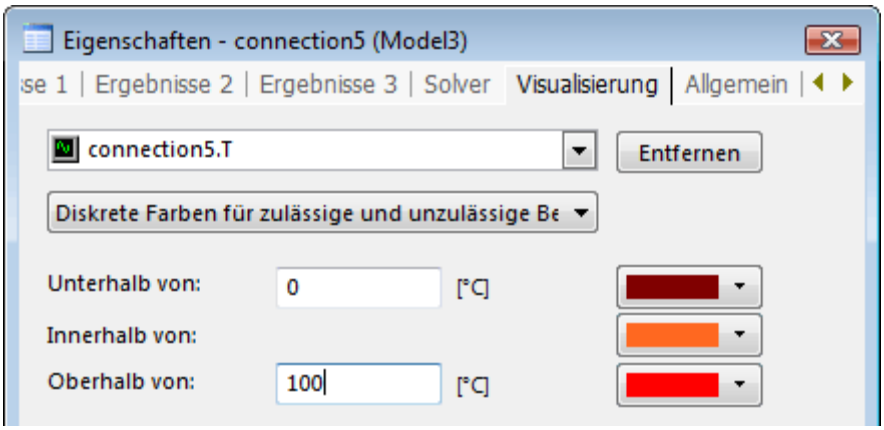
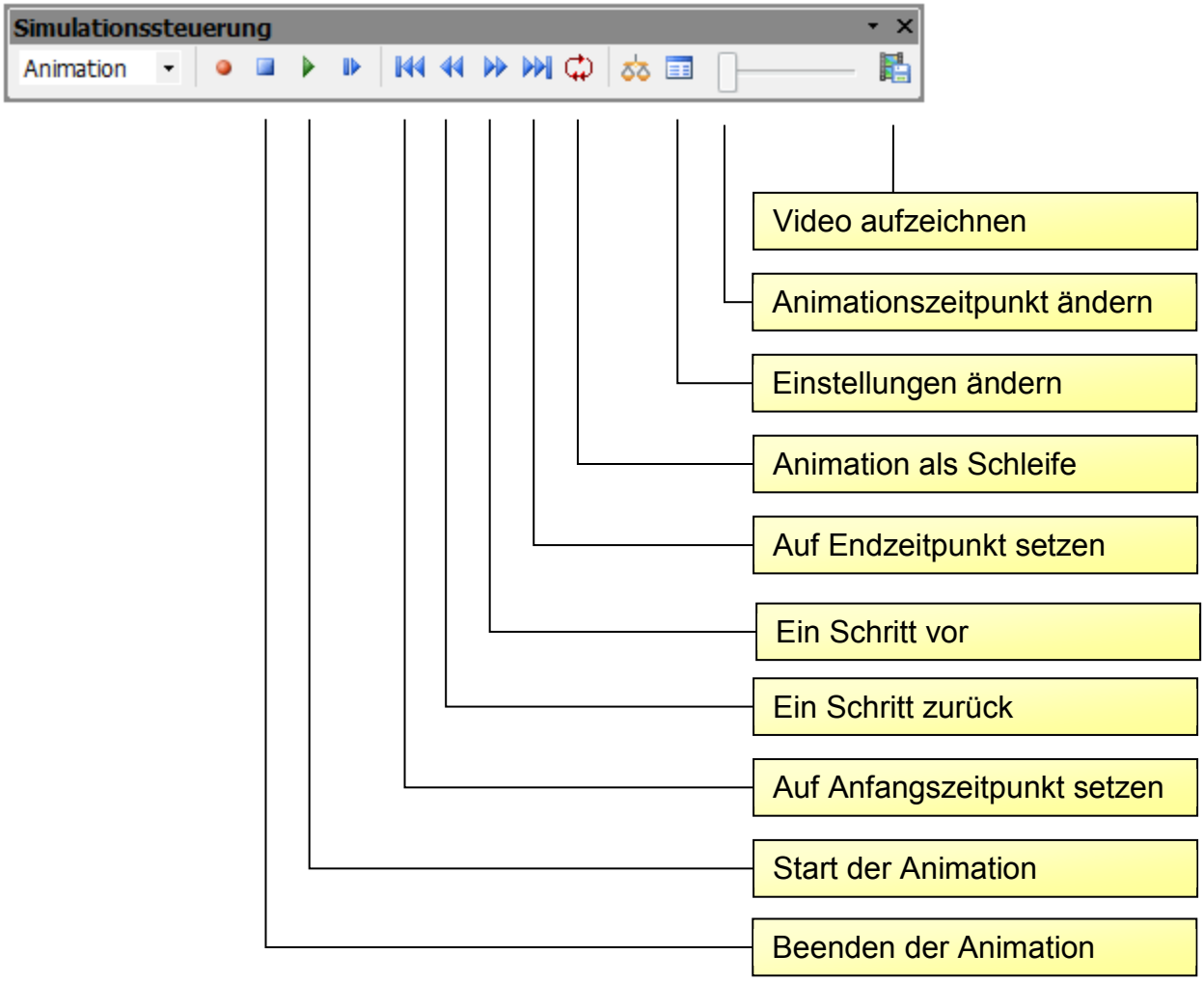



**Bild 6.24:** Einstellungen für die Visualisierung

Die Animation einer 3D-Ansicht kann als Videodatei gespeichert und mittels externer Player (z.B. Windows Media Player) abgespielt, weitergegeben oder in Präsentationen bzw. Internetseiten eingebunden werden.

Die Animation kann mit folgenden Befehlen gesteuert und angepasst werden:




### 6.2.1 Animation aufzeichnen

Um Berechnungsergebnisse in Echtzeit zu animieren, ist es zunächst erforderlich, die Ergebnisse einer transienten Simulation aufzuzeichnen. Betätigen Sie dazu vor Beginn der Rechnung die Schaltfläche Aufzeichnen in der Symbolleiste ***Simulationssteuerung***. Mit Beendigung der Berechnung zum Zeitpunkt `tStop` wird die Auswahlbox für die Analyseart automatisch auf ***Animation*** gesetzt. Ist diese Verhalten nicht gewünscht, können Sie im Menü ***Extras*** den Dialog ***Optionen*** öffnen und dort auf der Dialogseite ***Allgemein*** die Einstellung *Nach Ende der Aufzeichnung automatisch zu Animation wechseln* deaktivieren.

### 6.2.2 Animation wiedergeben

Stellen Sie zunächst sicher, dass die Symbolleiste Simulationssteuerung auf **Animation** eingestellt ist. Sollten die Schaltflächen für die Steuerung der Animation nicht freigegeben sein, stehen keine aufgezeichneten Berechnungs-ergebnisse für die Wiedergabe einer Animation zur Verfügung. Zeichnen Sie in diesem Fall zunächst die Ergebnisse einer transienten Simulation auf.

### 6.2.3 3D-Animation für externe Wiedergabe speichern

Die Animation kann auch für eine Wiedergabe mit einem externen Player im AVI-, oder WMV-Format abgespeichert werden. Mit der Schaltfläche in der Toolbar starten Sie die Aufzeichnung des Videos. Beim Abspeichern können neben dem Dateinamen und dem Speicherort noch weitere Einstellungen vorgenommen werden (siehe **Bild 6.25**)

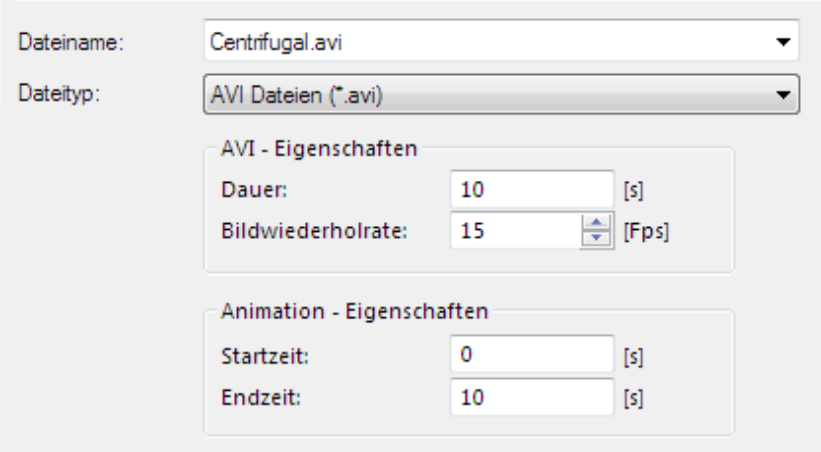



**Bild 6.25:** Video-Einstellungen

Um Speicherplatz zu sparen, sollte bei der Speicherung als AVI-Datei eine Videokomprimierung angewandt werden. Über ein Auswahlfeld (**Bild 6.26**) können Sie ein auf Ihrem Rechner vorhandenes Komprimierungsformat einstellen.

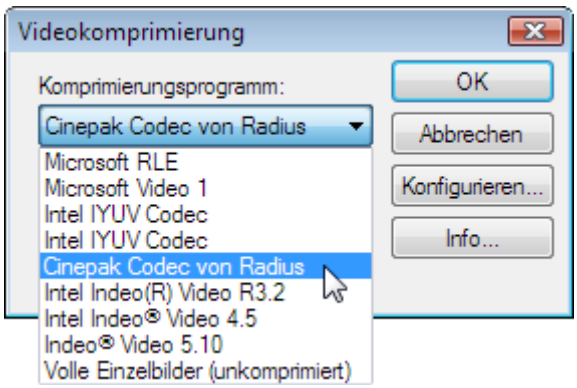



**Bild 6.26:** Auswahl der Videokomprimierung

## 6.3 Stationäre Simulation im Frequenzbereich

Die stationäre Simulation dient der Berechnung **periodisch eingeschwungener Zustände** nichtlinearer und linearer Systeme in Abhängigkeit einer Bezugsgröße (wie z.B. der mittleren Winkelgeschwindigkeit einer Drehmasse).
Wichtige Anwendungsgebiete sind:

- die Schwingungsanalyse bei Antriebssträngen mit Verbrennungsmotoren
- Bestimmung der Beschreibungsfunktion von nichtlinearen dynamischen Blöcken in Regelungssystemen

- Bestimmung aharmonischer Verzerrungen von elektronischen Verstärkern und Filtern in Abhängigkeit der Aussteuerung
- hydraulische und pneumatische Prüfstände für oszillierende Dauerbelastung

Zur Berücksichtigung der mit der Zeit wachsenden Winkelstellungen bei der **Schwingungsanalyse von rotierenden Antriebssträngen** wird der periodische Ansatz um einen mit der Zeit linear veränderlichen Anteil ergänzt.

Für die berechneten Ergebnisgrößen kann man sich im **Frequenzbereich** die **Amplituden der Spektralkomponenten**, die **Ungleichförmigkeitsgrade**, die **Erregungen**, die **Phasenverläufe** sowie die **Real-** und **Imaginärteile** des komplexen Spektrums anzeigen lassen. Für jede dieser Größen werden die **Summenkruve**, der **Mittelwert** und die einzelnen **Spektralkomponenten** (**Ordnungen**) dargestellt. Weiterhin sind für den periodischen Signalanteil der **Signalverlauf** über einer Periode und die **Ungleichförmigkeit** als typische **Zeitbereich**sergebnisse verfügbar.

Bei der Stationären Simulation können auch interne **Verhaltensbeschreibungen im Frequenzbereich** berücksichtigt werden, die keine Entsprechung im Zeitbereich haben.

So wird beim Feder-Dämpfer-Spiel-Element zur Beschreibung der Materialdämpfung die frequenzabhängige Dämpfung genutzt.

Die frequenzabhängige Dämpfung ist auch bei folgenden Elementen vorhanden:

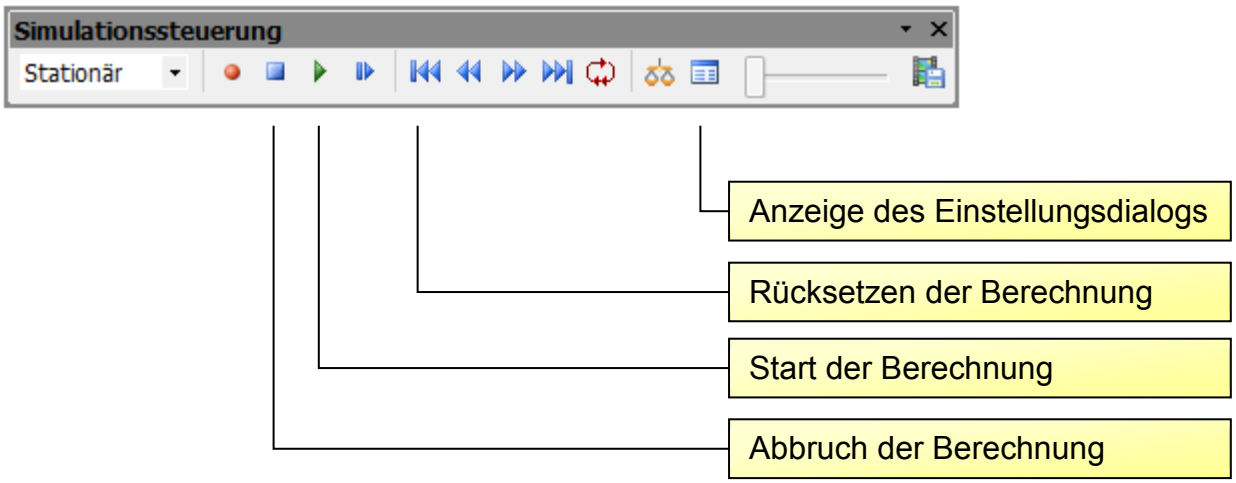
- Elastische Reibstelle (Mechanics.Rotation.ElasticFriction)
- Elastische Kupplung (PowerTransmission.Couplings.Coupling)
- Schaltkupplung (PowerTransmission.Couplings.DiscClutch)
- Zahnradstufe (PowerTransmission.Transmissions.Gear)
- Riemenstufe (PowerTransmission.Transmissions.BeltDrive)

Jedes dieser Elemente besitzt eine eigene Parameterseite für die stationäre Simulation, auf der die Dämpfungsart ausgewählt und die spektrale Leistung als stationäre Ergebnisgröße abgegriffen werden kann (der Realteil der spektralen Leistung ist die umgesetzte Verlustleistung).

Die Totzeit mit konstanter Verzögerungszeit wurde ebenfalls im Frequenzbereich implementiert, so dass diese jetzt bei der stationären Simulation für zeitkontinuierliche Eingangsignale die richtigen Ausgangssignale liefert.

Die Steuerung der Simulation im Frequenzbereich wird über die Schaltflächen der Symbolleiste **Simulationssteuerung** vorgenommen. Kontrollieren Sie vor Beginn der Rechnung, ob in der Auswahlbox für die Analyseart **Stationär** eingestellt ist.

Für die Steuerung der stationären Simulation hält die Symbolleiste Simulationssteuerung folgende Befehle bereit:




## 6.3.1 Grundlagen der stationären Simulation

### 6.3.1.1 Stationär eingeschwungener Zustand eines Systems

Ein dynamisches System, wie zum Beispiel ein Antriebsstrang, wird durch ein Algebro-Differentialgleichungssystem

$$\mathbf{0} = \mathbf{f}(\mathbf{x}(t), \dot{\mathbf{x}}(t)) \tag{6.8}$$

mit dem zeitabhängigken Vektor $\mathbf{x}(t)$ der Zustandsgrößen und dem zugehörigen Vektor $\dot{\mathbf{x}}(t)$ der Zeitableitungen und eine Gleichung

$$\mathbf{y} = \mathbf{g}(\mathbf{x}(t), \dot{\mathbf{x}}(t)) \tag{6.9}$$

zur Berechnung der Ergebnisgrößen $\mathbf{y}(t)$ beschrieben. Bei einem korrekten Modell stimmt die Anzahl $n$ der Zustandsgrößen und der Gleichungen in **(6.8)** überein, $\mathbf{x}(t)$ und $\mathbf{f}$ haben also die gleiche Anzahl von Komponenten.

Beispielsweise sind bei den meisten Antriebssträngen im Vektor $\mathbf{x}$ der Zustandsgrößen die Winkel und Winkelgeschwindigkeiten der Drehmassen des Strangs zusammengefasst und in **(6.5)** stehen die Newton'schen Bewegungsgleichungen für die Drehmassen im Strang und die Gleichungen, die festlegen, dass die Winkelgeschwindigkeiten die Zeitableitungen der Drehwinkel sind.
Im Rahmen der stationären Analyse werden Lösungen $\mathbf{x}(t)$ des Systems ermittelt, die sich als Überlagerung

$$\mathbf{x}(t) = \mathbf{x}_P \frac{t}{T} + \hat{\mathbf{x}}[0] + \tilde{\mathbf{x}}(t) \tag{6.10}$$

einer gleichförmigen (linearen) Bewegung $\mathbf{x}_P \frac{t}{T}$, eines konstanten Anteils $\hat{\mathbf{x}}[0]$ und eines mittelwertfreien **periodischen Signals** $\tilde{\mathbf{x}}(t)$ darstellen lassen.

Die periodische Bewegung wird dabei durch eine endliche harmonische Summe

$$\tilde{\mathbf{x}}(t) := \sum_{k=1}^{N} \hat{\mathbf{x}}_{\mathrm{R}}[k]\cos(\omega k t) - \hat{\mathbf{x}}_{\mathrm{I}}[k]\sin(\omega k t) \qquad \textbf{(6.11)}$$

approximiert. Die Indizes R und I beziehen sich dabei auf den Real- und Imaginärteil der komplexen Amplituden $\underline{\hat{\mathbf{x}}}[k] = \hat{\mathbf{x}}_{\mathrm{R}}[k] + \mathrm{j}\hat{\mathbf{x}}_{\mathrm{I}}[k]$ in der äquivalenten Darstellung

$$\tilde{\mathbf{x}}(t) = \sum_{k=1}^{N} \mathrm{Re}\left(\underline{\hat{\mathbf{x}}}[k]\exp(\mathrm{j}\omega k t)\right). \qquad \textbf{(6.12)}$$

Die in diesen Gleichungen auftretenden Größen sind:

| | |
|---|---|
| $\mathbf{x}_{\mathrm{P}}$ | ein konstanter Periodenvektor mit der Dimension des Zustandsvektors (näher erläutert in Abschnitt 6.3.1.2) |
| $T$ | die Periodendauer des Schwingungsvorganges |
| $\omega$ | die Phasengeschwindigkeit $\omega = \frac{2\pi}{T}$ des Schwingungsvorganges |
| $\underline{\hat{\mathbf{x}}}[k]$ | komplexe Amplitude der $k$-ten Spektralkomponente |
| $\hat{\mathbf{x}}[0]$ | der konstante Signalanteil (im Weiteren auch als Mittelwert bezeichnet, was streng genommen nur für $\mathbf{x}_P = 0$ gilt) |
| $\hat{\mathbf{x}}_{\mathrm{R}}[k], \hat{\mathbf{x}}_{\mathrm{I}}[k]$ | Realteil und Imaginärteil der komplexen Amplitude der $k$-ten Spektralkomponente des Zustandsvektors |
| $N$ | Anzahl der Spektralkomponenten die zur Darstellung von $\tilde{\mathbf{x}}(t)$ benutzt werden |

Den Ergebnisgrößen $\mathbf{y}$ werden ebenfalls der Mittelwert $\hat{\mathbf{y}}[0]$ und die komplexen Spektralkomponenten $\underline{\hat{\mathbf{y}}}[k]$ mit $k = 1,\ldots,N$ zugeordnet und in die verschiedenen Anzeigegrößen umgerechnet. Die Berechnungsgrundlagen für die Anzeigegrößen sind detailliert im **Kapitel 6.3.5** erläutert.

### 6.3.1.2 Periodenvektor, Periodenvariable und Periodenermittlung

Den Sinn des Periodenvektors $\mathbf{x}_{\mathrm{P}}$ verdeutlicht man sich am besten anhand des stationär eingeschwungenen Zustandes bei einem frei drehbaren Antriebsstrang mit einem Verbrennungsmotor. Der Motor verursacht ein drehwinkelabhängiges Drehmoment, das im gesamten Strang Schwingungen anregt. Im Rahmen der stationären Analyse untersucht man diese Schwingungen üblicherweise bei vorgegebenen mittleren Motordrehzahlen ungleich null.
Der Winkel einer im Strang liegenden Drehmasse wird sich innerhalb einer Schwingungsperiode um das Produkt der mittleren Drehgeschwindigkeit der Drehmasse mit der Periodendauer der Schwingung weiterdrehen.
Dieses Weiterdrehen wird im Ansatz durch den Periodenvektor $\mathbf{x}_{\mathrm{P}}$ berücksichtigt. Die zu den Drehwinkeln der Drehmassen gehörigen Komponenten des

Periodenvektors haben gerade den Wert des innerhalb einer Schwingungsperiode zurückgelegten Winkels. Liegt zum Beispiel zwischen zwei Drehmassen eine Getriebeübersetzung, so ist das Verhältnis der zu den Drehmassen gehörigen Drehwinkelkomponenten des Periodenvektors gerade gleich dem Übersetzungsverhältnis des Getriebes.
Für diejenigen Zustandsgrößen des Systems, die einen rein periodischen Zeitverlauf haben, sind die zugehörigen Komponenten des Periodenvektors gleich null. Beim Beispiel eines Antriebsstranges ist das für die Winkelgeschwindigkeiten der Fall.
Für eine als **Periodenvariable** bezeichnete Zustandsvariable des Systems muss der Anwender den Wert für die Periode selber festlegen. Zum Beispiel beim Antriebsstrang mit 1-Zylinder-4-Takt-Verbrennungsmotor wiederholt sich der Erregungsverlauf nach zwei Umdrehungen der Kurbelwelle. Es ist also naheliegend, in diesem Falle den Kurbelwellenwinkel als Periodenvariable und zwei Umdrehungen (bzw. $4\pi$ im Bogenmaß) als zugehörigen Wert für die Periode zu wählen.
Die Werte der anderen Komponenten des Periodenvektors werden in einem separaten als **Periodenermittlung** bezeichneten Rechenschritt, am Anfang der stationären Analyse von SimulationX ermittelt.

1. Das für die Periodenermittlung eingesetzte quasi-statische Berechnungsverfahren erfordert, dass Zwangsbedingungen in den Zustandsgrößen selber formuliert sein müssen und nicht in den Zeitableitungen der Zustandsgrößen. Zum Beispiel für Antriebsstränge bedeutet das, dass Zwangsbedingungen auf **Wegebene** und nicht auf Geschwindigkeitsebene zu formulieren sind.

2. Die Modellgleichungen sollten stetig sein. Dies ist insbesondere an den Stellen wesentlich, an denen die Periodenvariable gleich null oder gleich der vorgegebenen Periode ist und alle Geschwindigkeitsgrößen null sind.

#### 6.3.1.3 Harmonische Balance, Parametrische Analyse, Bezugsgröße und Ausgleichsparameter

Nach der Periodenermittlung werden die noch unbekannten Mittelwerte $\hat{\mathbf{x}}[0]$, die Spektralkomponenten $\underline{\hat{\mathbf{x}}}[1],\ldots,\underline{\hat{\mathbf{x}}}[N]$ und die Phasengeschwindigkeit $\omega$ im Ansatz **(6.10), (6.11)** so abgeglichen, dass das Gleichungssystem **(6.8)** zumindest für den Mittelwert $\hat{\mathbf{f}}[0]$ und die ersten $N$ Spektralkomponenten $\underline{\hat{\mathbf{f}}}[1],\ldots,\hat{\mathbf{f}}_{\mathrm{I}}[N]$ der rechten Seite erfüllt ist.

Der Algorithmus für diesen spektralen Abgleich wird auch als **harmonische Balance** bezeichnet.

Die harmonische Balance von SimulationX ist als parametrische Analyse ausgelegt. Das heißt, der stationär eingeschwungene Zustand wird in Abhängigkeit einer **Bezugsgröße** $x_{\mathrm{Ref}}$ ermittelt, die innerhalb eines vom Anwender vorgegebenen Bereiches durchgestimmt wird.

Als Bezugsgröße kann der Anwender einen **Parameter** oder den **Mittelwert einer Zustandsvariable** auswählen.
Eine Komponente des konstanten Anteils $\hat{\mathbf{x}}[0]$ im Ansatz **(6.8)** für die harmonische Balance ist frei wählbar. Um eine eindeutige stationäre Lösung zu erhalten, wird bei SimulationX deshalb der Mittelwert der Periodenvariablen gleich null gesetzt. Damit ist die Anzahl der Unbekannten und Gleichungen bei der harmonischen Balance ausgeglichen.

**Fall 1: Parameter als Bezugsgröße**

Nutzt der Anwender einen **Parameter** als Bezugsgröße, wird aus dem Algebro-Differentialgleichungssystem **(6.8)** einfach ein parametrisiertes Algebro-Differentialgleichungssystem

$$0 = \mathbf{f}\left(\mathbf{x}, \dot{\mathbf{x}}; x_{\mathrm{Ref}}\right) \qquad \textbf{(6.13)}$$

bei dem für jeden vorgegebenen Wert der Bezugsgröße $x_{\mathrm{Ref}}$ die Anzahl der Gleichungen und Variablen übereinstimmt.

**Fall 2: Variable als Bezugsgröße**

Der Anwender hat auch die Möglichkeit, den **Mittelwert einer Zustandsvariable** $\hat{x}_r[0]$ als Referenzgröße $x_{\mathrm{Ref}}$ zu wählen.
Zum Beispiel bei einem Antriebsstrang liegt es nahe, den Mittelwert der Winkelgeschwindigkeit einer Drehmasse als Bezugsgröße zu nutzen.
Die Relation $\hat{x}_r[0] = x_{\mathrm{Ref}}$ kann als zusätzliche Gleichung für die harmonische Balance interpretiert werden. Damit die Anzahl der unabhängigen Variablen und Gleichungen bei der harmonischen Balance wieder übereinstimmen, muss ein Parameter des Systems als unabhängige Variable freigegeben werden. Dieser freizugebende Parameter wird bei SimulationX als Kompensationparameter $x_{\mathrm{Comp}}$ bezeichnet.

Das Algebro-Differentialgleichungssystem **(6.8)** nimmt dann also die Form

$$0 = \mathbf{f}\left(\mathbf{x}, \dot{\mathbf{x}}, x_{\mathrm{Comp}}\right), \quad \hat{x}_r[0] = x_{\mathrm{Ref}} \qquad \textbf{(6.14)}$$

an.

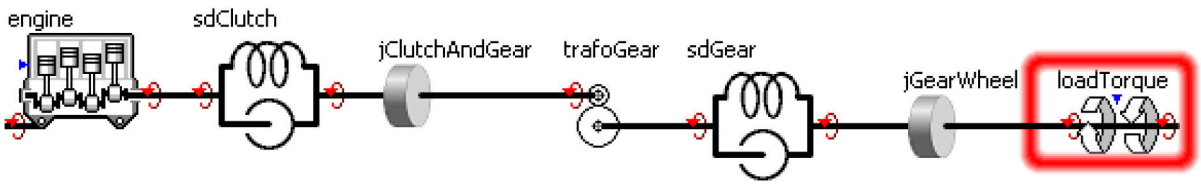


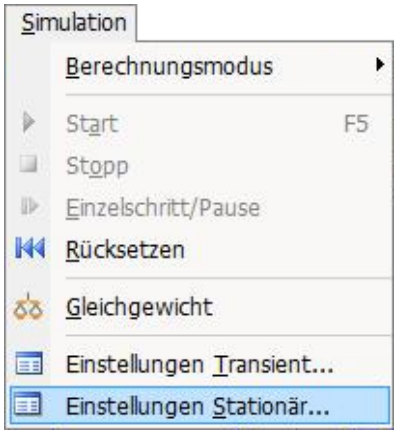

**Bild 6.27:** Antriebsstrang, bei dem zur Kompensation des mittleren Motormoments das Lastmoment *loadTorque* ergänzt wurde.

Die Rolle der Kompensationsvariable lässt sich wieder einfach am Beispiel eines **Antriebsstranges** (siehe **Bild 6.27**) verdeutlichen. Ein lastfreier Antriebsstrang wird beschleunigt und bleibt nicht stationär auf einer gewünschten Drehzahl, wenn er mit Vollgas durch einen Verbrennungsmotor angetrieben wird.

Erst durch Aufbringen eines passenden Lastmoments am Ende des Stranges, welches das mittlere Motormoment kompensiert, lässt sich die mittlere Motordrehzahl auf dem gewünschten Wert halten (siehe **Bild 6.27**). Beim Antriebsstrang ist es also naheliegend, ein Lastmoment zu ergänzen und den Wert des Moments als Ausgleichsparameter zu wählen.

## 6.3.2 Der Einstellungsdialog für die stationäre Simulation

Vor der ersten stationären Simulation für ein SimulationX-Modell, sind systemspezifische Parametereinstellungen erforderlich. Die zugehörigen Eingabefelder befinden sich auf der Seite *System* des Einstellungsdialogs für die Stationäre Simulation.




Den Einstellungsdialog können Sie über den Punkt ***Einstellungen Stationär*** des Menüs ***Simulation*** öffnen.

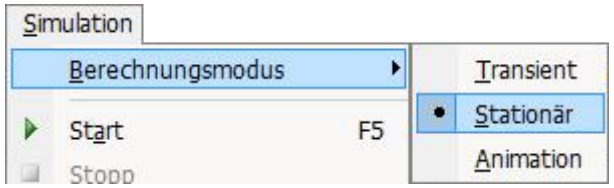
Wird versucht, die stationäre Simulation bei fehlenden oder falschen Parametereinstellungen zu starten, so öffnet sich nach einer Fehlermitteilung ebenfalls der Einstellungsdialog.

### 6.3.2.1 Dialogseiten System und Verfahren

Mit den am oberen Rand des Dialoges für die stationäre Simulation befindlichen Reitern

wechseln Sie die angezeigte Dialogseite. Eine detaillierte Hilfe zu den Dialogseiten finden sie in den Abschnitten **6.3.3** (System-Seite) und **6.3.4** (Verfahrens-Seite).

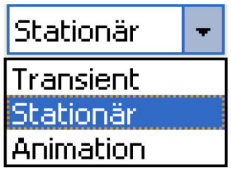
### 6.3.2.2 Start der stationären Simulation




Nachdem Sie alle erforderlichen Einstellungen vorgenommen haben, können Sie die stationäre Simulation starten.

Wählen Sie dazu im Menü ***Simulation*** den Berechnungsmodus ***Stationär*** und klicken Sie auf das Startsymbol oder nutzen Sie die Taste F5.

Den Berechnungsmodus können Sie auch mit Hilfe der Auswahlbox




in der Symbolleiste wechseln.

## 6.3.3 System-Seite des Einstellungsdialoges

Auf der Seite *System* des Einstellungsdialoges für die stationäre Simulation sind die folgenden Größen festzulegen:

1. die Bezugsgröße (Parameter oder Variable)
2. die Periodenvariable und ihre Periode
3. die Ordnungen
4. der Ausgleichsparameter

### 6.3.3.1 Festlegung der Bezugsgröße

Die stationäre Analyse ist als parametrische Analyse ausgelegt. Dies ermöglicht die bequeme Untersuchung der periodischen Lösung eines Systems in Abhängigkeit einer vom Nutzer festgelegten **Bezugsgröße** des Systems innerhalb eines vom Nutzer vorgegebenen Bereiches.
Die Bezugsgröße wird aus der Baumansicht im Abschnitt *Vorgabe Bezugsparameter/-variable* auf der System-Seite ausgewählt (siehe auch **Bild 6.28**). Sie kann eine Systemvariablen oder ein Parameter sein.

Die Eingabefelder *Start* und *Stopp* legen die Grenzen fest, innerhalb derer die Bezugsgröße bei der stationären Analyse variiert wird. Die Werte für *Start* und *Stopp* unterliegen keinen Einschränkungen. So kann *Stopp* größer, gleich oder auch kleiner als *Start* eingestellt werden. Wählt man für beide Einstellungen den gleichen Wert, so wird die stationäre Lösung nur für diesen Parameterwert berechnet.

In dem Beispiel aus **Bild 6.28** wird die Winkelgeschwindigkeit der Drehmasse `jClutchAndGear` per Mausklick gewählt. Die Drehmasse und die ausgewählte Variable sind durch rote Kreise gekennzeichnet.

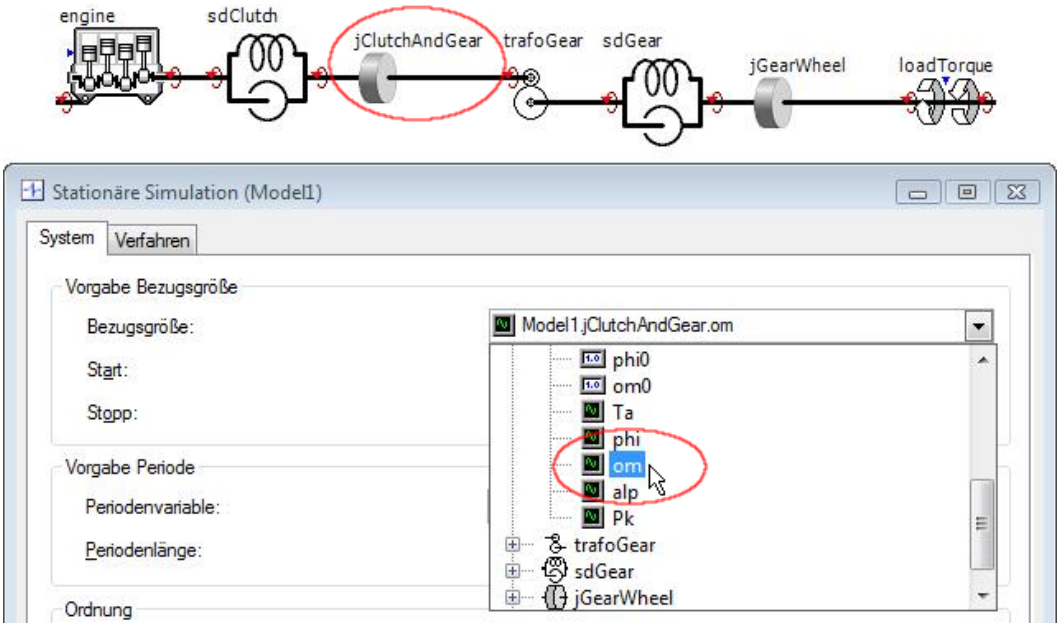



**Bild 6.28:** Auswahl der Bezugsgröße

Legt man wie im Beispiel (siehe **Bild 6.28**) eine **Variable als Bezugsgröße** fest, so wird während der stationären Simulation der **Mittelwert** dieser Variable variiert. Bei der Wahl einer Variablen als Bezugsgröße muss der Anwender auch einen **Ausgleichsparameter** angeben (siehe Abschnitt **6.3.3.4**). Der Algorithmus für die harmonische Balance kann dann den Wert dieses Ausgleichsparameters anpassen, um den Mittelwert der Bezugsgröße auf einen gewünschten Wert abzugleichen.

Falls man als Bezugsgröße einen Parameter wählt, so entfällt die Festlegung eines Ausgleichsparameters.

### 6.3.3.2 Festlegung der Periodenvariable und der Periode

Wie im Grundlagenabschnitt **6.3.1** beschrieben ist, setzen sich die Zeitverläufe der Zustandsgrößen aus einem periodischen Signal und einem linear mit der Zeit veränderlichen Signal zusammen. Durch den linearen Signalanteil unterscheidet sich bei einigen Zustandsgrößen der Wert zum Anfangszeitpunkt von dem nach einer Schwingungsdauer (Beispiel: bei einem Antriebsstrang mit Erregung durch einen Einzylinder-Viertaktmotor wird sich der Kurbelwellenwinkel innerhalb einer Erregungsperiode um $4\pi$ ändern).

Eine dieser Zustandsgrößen ist über die Auswahlbox ***Variable*** im Feld ***Vorgabe Periode*** als **Periodenvariable** auszuwählen (z.B. die Winkelstellung der Kupplung, wie im Beispiel, oder der Kurbelwellenwinkel). Für die Periodenvariable ist die **Periodenlänge** anzugeben. Das ist der Wert, um den sich die Periodenvariable innerhalb einer vollen Schwingung ändert (z.B. $4\pi$ für den Kurbelwellenwinkel). Die Periodendauer des Schwingungsvorganges wird dann aus diesen Angaben von der stationären Simulation selber berechnet.

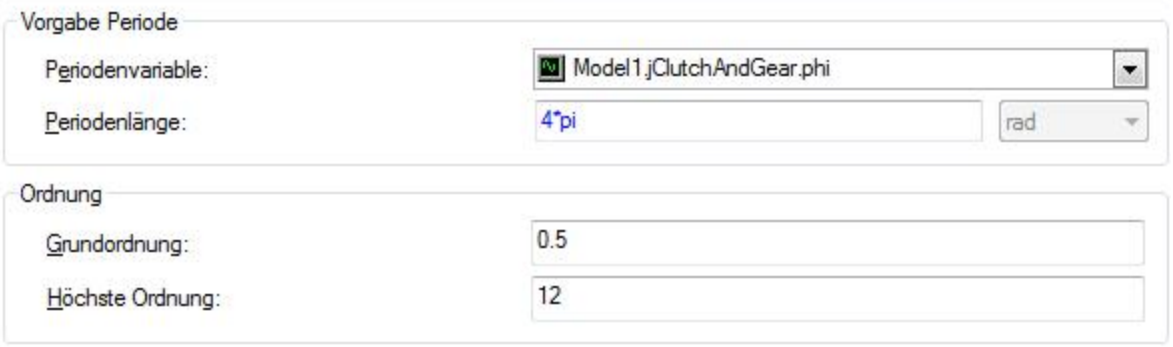



**Bild 6.29:** Vorgabe der Periode, der Grundordnung und der höchsten Ordnung für einen Viertaktmotor

Detaillierte Informationen zu Periodenvariable und Periodenlänge finden Sie auch in Abschnitt **6.3.1**.

#### 6.3.3.3 Festlegung der Ordnungen

Die Periodendauer der niedrigsten berechneten Spektralkomponente (abgesehen vom Mittelwert) wird durch die Periodenlänge für die Periodenvariable festgelegt. Dieser Spektralkomponente wird die vom Anwender angegebene **Grundordnung** zugeordnet, der zweiten Spektralkomponente das Doppelte der Grundordnung und so weiter bis zur vom Nutzer spezifizierten **höchsten Ordnung**.
Die Anzahl der zur Berechnung herangezogenen Spektralkomponenten ergibt sich dann aus dem Quotienten der Höchsten Ordnung und der Grundordnung.
Beim stationären Betrieb von Antriebssträngen mit Viertaktmotor sind die Schwingungsvorgänge periodisch mit zwei Umdrehungen der Kurbelwelle. Es ist jedoch üblich, der Spektralkomponente die mit einer Umdrehung der Kurbelwelle periodisch ist, die Ordnung eins zuzuordnen. Dazu setzt man in diesem Anwendungsgebiet üblicherweise die Grundordnung auf 0.5, wie das auch bei der Voreinstellung von SimulationX der Fall ist.

#### 6.3.3.4 Festlegung des Kompensationsparameters

Falls als Bezugsgröße eine Variable gewählt wurde (siehe Abschnitt **6.3.3.1**), so ist der Kompensationsparameter ebenfalls aus einer Baumansicht, die alle Modellparameter anzeigt, auszuwählen.
Der Wert des Kompensationsparameters sollte (direkt oder indirekt) den Mittelwert der Bezugsgröße beeinflussen, so dass der Algorithmus der harmonischen Balance die Möglichkeit hat, durch Änderung dieses Wertes die Bezugsgröße auf einen gewünschten Wert einzustellen.

Bei dem in **Bild 6.27** dargestellten Antriebsstrang wurde zum Beispiel der Wert des Lastmoments *Model1.loadTorque.Ti* als Kompensationsparameter gewählt.

### 6.3.4 Verfahren-Seite des Einstellungsdialoges

Auf der Seite ***Verfahren*** des **Einstellungsdialog**es für die **stationäre Simulation** sind die wesentlichen Verfahrensparameter für die Periodenermittlung und die har-

monische Balance verfügbar. In vielen Anwendungsfällen können die Standardeinstellungen auf dieser Dialogseite beibehalten werden.

### 6.3.4.1 Relative Größen

Oftmals interessieren in den Simulationsergebnissen auch Resonanzspitzen von Spektralkomponenten höherer Ordnung, die oft wesentlich kleiner sind, als diejenigen niedrigerer Ordnung. Damit betragsmäßig kleine Komponenten genauso fein abgefahren werden, wie betragsmäßig große, werden zur Festlegung der Rechengenauigkeit und der Dichte der Rechenpunkte entlang der Ergebniskurve nur **relative Maße** benutzt. D.h. zur Bestimmung dieser Maße, wie z.B. der relativen Schrittweite, werden alle Komponenten, bei denen die Differenz zwischen Maximum und Minimum der bisher berechneten Werte eins überschreitet mit dem **Skalierungsfaktor 1/(Maximum-Minimum)** multipliziert. In Richtung der Bezugsgröße wird zur Bestimmung der relativen Schrittweite mit **1/|StopWert-StartWert|** skaliert, falls StartWert und StopWert voneinander verschieden sind.

### 6.3.4.2 Schrittweitenparameter

Die relativen Grenzen für die Schrittweite bestimmen wesentlich die Feinheit mit der die Kurven über der Bezugsgröße berechnet und angezeigt werden. Standardmäßig kommt ein Kurvenverfolgungsalgorithmus mit Schrittweitensteuerung zur Berechnung dieser Kurven zum Einsatz. Dieser Algorithmus ermöglicht die Berechnung von Umkehrpunkten in Frequenzgängen von nichtlinearen Systemen mit Kipperscheinungen sowie ein feines "Ausfahren" von Resonanzspitzen in Frequenzgängen linearer Systeme.

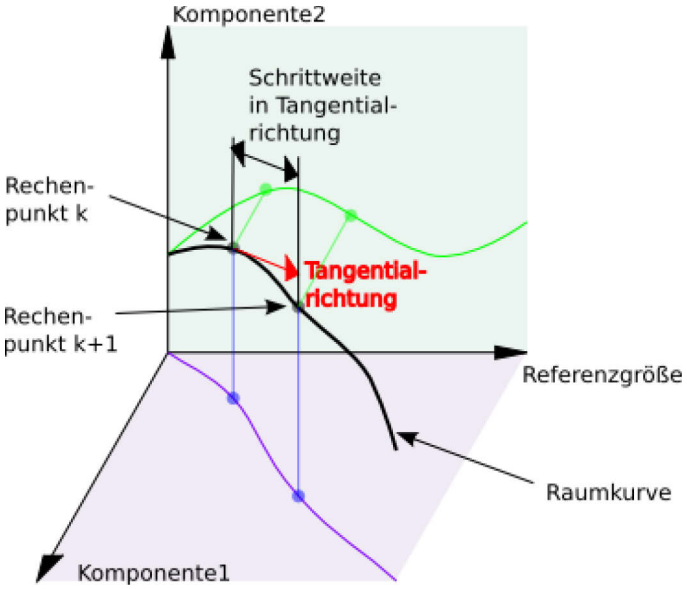



**Bild 6.30:** Schrittweite tangential zur Kurve

Dabei wird die **Schrittweite** nicht in Richtung der Bezugsgröße sondern **tangential zur berechneten Kurve** im Raum der Spektralkomponenten für die Zustandsgrößen festgelegt (siehe **Bild 6.30**). Dies ist vor allem bei Umkehrpunkten unerlässlich (siehe **Bild 6.31**).

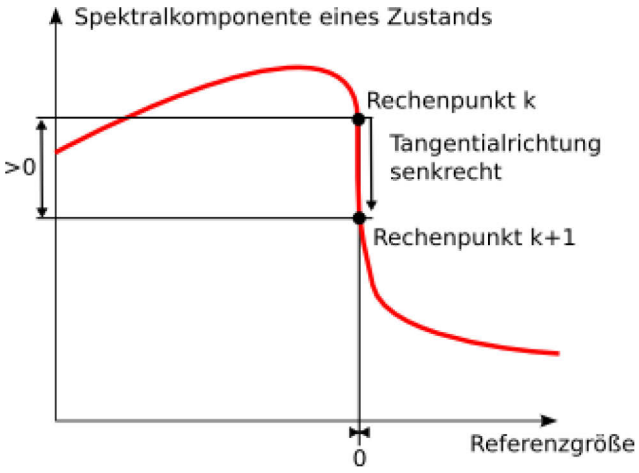



**Bild 6.31:** Frequenzgänge nichtlinearer Systeme können Umkehrpunkte aufweisen, in denen die Tangentialrichtung an die Kurve senkrecht zur Richtung der Bezugsgröße liegt. In solchen Punkten ist die Schrittweite in Richtung der Bezugsgröße gleich null.

Die Kurvenkrümmung und das Konvergenzverhalten des Newton-Algorithmus werden ebenfalls bei der Schrittweitenregelung berücksichtigt. Diese Schrittweitenregelung lässt sich über die Optionen auf der Seite ***Weitere Parameter…*** beeinflussen.

### 6.3.4.3 Relative Toleranz

Die relative Toleranz ist ein Maß für die Genauigkeit, mit der die Spektren berechnet werden. Der numerische Fehler wird für die skalierten Spektralkomponenten der Zustandsgrößen geschätzt (siehe Abschnitt **6.3.4.1**).

---

**Hinweise:**
Da die exakte Lösung nicht bekannt ist, kann auch der numerische Fehler nicht exakt bestimmt werden. Als ein Maß für den numerischen Restfehler der Spektren der Zustandsgrößen wird die Länge des aktuellen Newton-Schrittes benutzt.
Neben dem am Ende der Newton-Iteration verbleibenden Restfehler, der durch die relative Toleranz beschränkt wird, spielt auch noch der durch die zeitliche Abtastung verursachte Fehler eine Rolle. Dieser Fehler kann nur durch eine Überabtastung reduziert werden (siehe Abschnitt **6.3.4.4**).

Beim linearen Verfahren ist die relative Toleranz bedeutungslos, denn bei diesem Verfahren wird genau ein Newton-Schritt ausgeführt, unabhängig davon, ob damit die gewünschte Genauigkeit erreicht wird (siehe auch Abschnitt **6.3.4.5**).

---

### 6.3.4.4 Überabtastung

Bei der harmonischen Balance müssen Funktionen, die nichtlinear von den Zustandsgrößen abhängen, im Zeitbereich ausgewertet werden. Für den Abgleich der Residuen werden die Funktionswerte mittels schneller Fourieranalyse (**FFT** = Fast Fourier Transformation) in den Frequenzbereich transformiert. Die **Überab-**

**tastung** bestimmt, wie viele Abtastpunkte im Zeitbereich für diese Fourieranalyse benutzt werden.

Ist $N$ die Anzahl der Spektralkomponenten für die harmonische Balance, so werden für die FFT mindestens

$$\text{Überabtastung} \cdot 2N$$

Abtastpunkte im Zeitbereich genutzt. Da die FFT bei Potenzen von zwei am effektivsten arbeitet, wird die Zahl der Abtastpunkte jedoch noch bis auf die nächste Zweierpotenz erhöht.

#### 6.3.4.5 Verfahren

Mit der Auswahlbox **„*Verfahren*“** kann eins der folgenden Verfahren für die harmonische Balance selektiert werden:

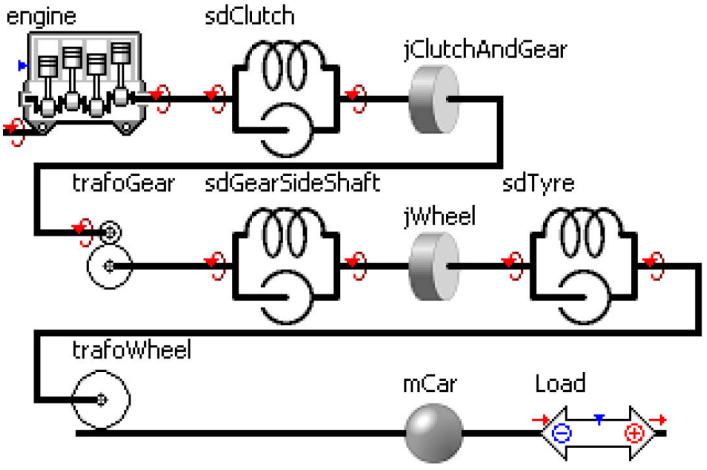
1. **Nichtlineares Verfahren** (Newton, GMRES, Jacobi-Präkond.)
   Beim nichtlinearen Verfahren wird das harmonische Newton-Verfahren auf die Systemgleichungen angewandt. Dabei werden so viele Newton-Iterationen ausgeführt, bis der anhand eines Newton-Schrittes geschätzte relative Fehler in den Spektralkomponenten der Zustandsgrößen kleiner als die vorgegebene Fehlerschranke ist. Es ist zu beachten, dass der Abtastfehler nicht in die Fehlerschätzung eingeht. Die Überabtastung muss also vom Anwender hoch genug gewählt werden, so dass der Abtastfehler das Gesamtergebnis nicht zu stark verfälscht.
   Der harmonische Newton-Algorithmus erfordert die Lösung eines großen linearen Gleichungssystems in jedem Iterationsschritt. Die Erweiterung GMRES im Namen verweist auf den iterativen Algorithmus zur Lösung dieses linearen Systems. Der GMRES-Algorithmus kann durch einen Vorkonditionierer beschleunigt werden. Dies ist eine Approximation der Jacobimatrix für die harmonischen Balance-Gleichungen. Im Fall des Jacobi-Vorkonditionierers werden die Jacobimatrizen des Systems **(6.5)** über eine Schwingungsperiode gemittelt und dann für jede einzelne Frequenzkomponente genutzt. Daher werden keine Frequenzmischungseffekte durch den Vorkonditionierer zu berücksichtigt. GMRES kann das lineare System dennoch genau lösen. Jedoch können für die richtige Lösung mehr GMRES-Iterationen notwendig sein.
2. **Lineares Verfahren**
   Beim linearen Verfahren wird in jedem Punkt der Frequenzgangkurve nur ein Newton-Schritt des harmonischen Newton-Verfahrens ausgeführt. Damit ist das lineare Verfahren im Normalfall wesentlich schneller als das nichtlineare Verfahren. Jedoch kann bei diesem Verfahren der Abbruchsfehler des Newton-Algorithmus nicht kontrolliert werden (dazu wären mindestens zwei Newton-Schritte notwendig).
3. **Nichtlineares Verfahren** (Newton, GMRES, approxim. Jacobian)
   Diese Methode unterscheidet sich nur durch die Vorkonditionierung von der

nichtlinearen Methode mit Jacobi-Vorkonditionierer (siehe oben). Alle Einträge des Joacobi-Vorkonditionierers werden auch für die Approximation der harmonischen Jacobimatrix genutzt. Darüber hinaus werden stark mischende Matrixelemente in diesem Vorkonditionierer berücksichtigt. Dies kann die Konvergenz des GMRES-Algorithmus verbessern. Auf der anderen Seite werden die harmonischen Jacobimatrizen größer und strukturell komplizierter, was zu größeren Rechenzeiten führt. So kann diese Methode als Alternative gesehen werden, wenn das nichtlineare Verfahren mit Jacobi-Vorkonditionierer fehlschlägt.

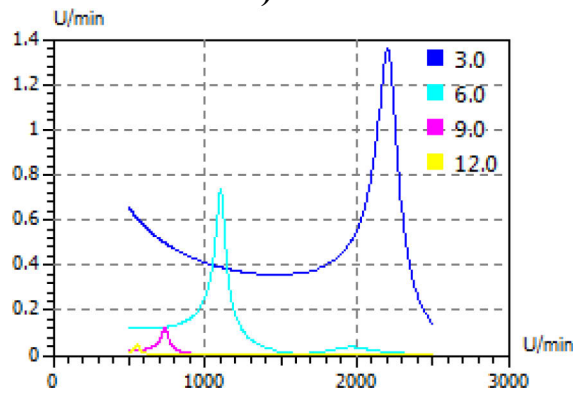
4. **Lineares Verfahren mit Interpolation**
   Dieses Verfahren ist eine beschleunigte Version des linearen Verfahrens. Für die meisten leicht nichtlinearen Modelle ermöglicht es eine zehnmal schnellere Berechnung als die lineare Methode ohne Interpolation. Die hauptsächliche Beschleunigung der Rechnung beim interpolierenden Verfahren resultiert aus der besseren Ausnutzung der Schrittweitensteuerung für die Referenzgröße. In Powertrain-Anwendungen wird in der Regel die Drehzahl des Motors als Referenzgröße ausgewählt. Oft haben die verschiedenen Schwingungsordnungen Resonanzen bei unterschiedlichen Geschwindigkeiten. Daher variieren die Amplituden der Ordnungen sehr stark in verschiedenen Bereichen der Referenzgröße. In diesen Bereichen wird die Referenzschrittweite verkleinert, um die schnellen Amplitudenänderungen zu erfassen. Ohne Interpolation gibt es nur ein gemeinsames Referenzgrößengitter. Der Mittelwert und sämtliche Ordnungen werden in jedem Punkt dieses Gitters berechnet. Dies verursacht einen hohen Rechenaufwand.
   Beim Verfahren mit Interpolation werden die Mittelwerte und die Schwingungs-Ordnungen getrennt mit jeweils eigener Schrittweitensteuerung und eigenem Gitter berechnet. Dies ermöglicht ein gröberes Gitter und geringeren numerischen Aufwand ohne Verlust an Präzision. Nur die Ergebnisse werden auf einem kombinierten Gitter interpoliert.

Die linearen Verfahren eignen sich für Systeme, bei denen der Konstantanteil und die linearer Zustandsentwicklung gegenüber dem oszillierenden Anteil dominieren. Wichtige Beispiele für solche Systeme sind Antriebsstränge mit Verbrennungsmotoren, dessen Pleuel starr modelliert sind.

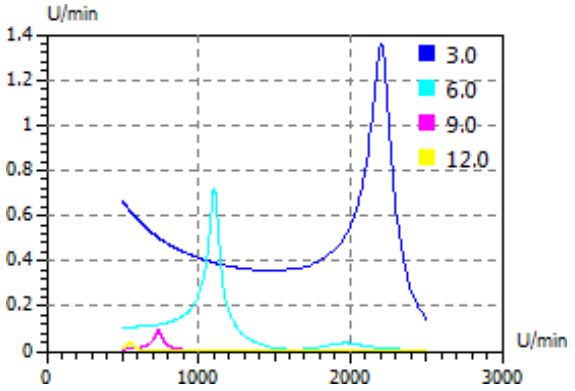
In **Bild 6.32** werden Simulationsergebnisse für einen solchen Strang verglichen, die einmal mit dem nichtlinearen und einmal mit dem linearen Verfahren gewonnen wurden. Wie man sieht, stimmen diese Ergebnisse gut überein. Allgemein kann man sagen, dass bei Antriebssträngen mit starrem Pleuel die mit den linearen Verfahren erreichten Genauigkeiten meistens vollkommen ausreichend sind.




**a)** Modell




**b)** Ergebnis der nichtlinearen Rechnung



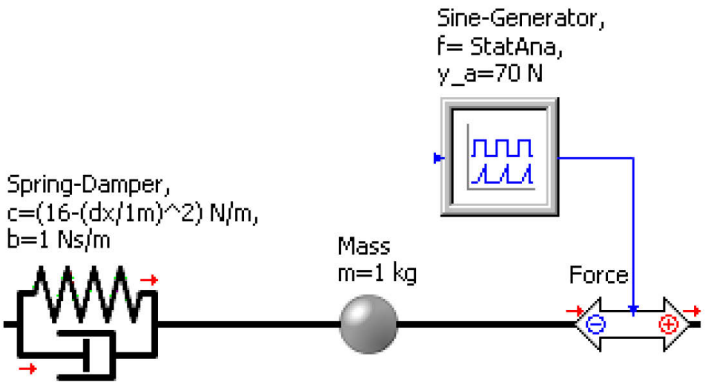

**c)** Ergebnis der linearen Rechnung

**Bild 6.32:** Vergleich der Ergebnisse der nichtlinearen und der linearen Rechnung bei einem Antriebsstrang mit 6-Zylindermotor. Dargestellt sind die Amplituden der in der Farblegende aufgeführten Ordnungen von `jClutchAndGear.om`

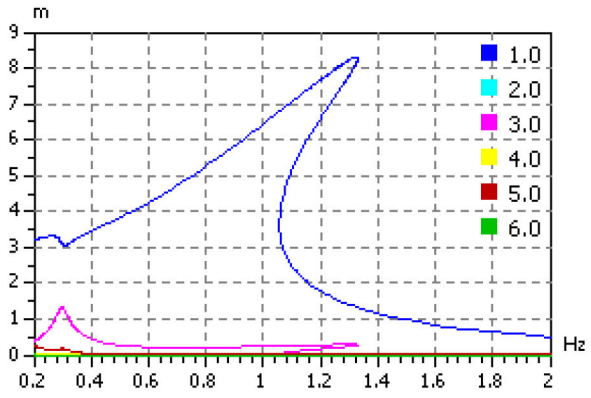
Für lineare Systeme die durch einen Generator erregt werden, dessen Frequenz als Bezugsgröße und dessen Phase als Periodenvariable gewählt wurden, sind die linearen Verfahren exakt.

Bei Systemen deren Verhalten wesentlich durch die Nichtlinearitäten beeinflusst wird, können die Ergebnisse des linearen Verfahrens jedoch mit großen Fehlern behaftet sein. Ein Beispiel dafür ist der Feder-Masse-Schwinger mit nichtlinearer Federcharakteristik in **Bild 6.33**.

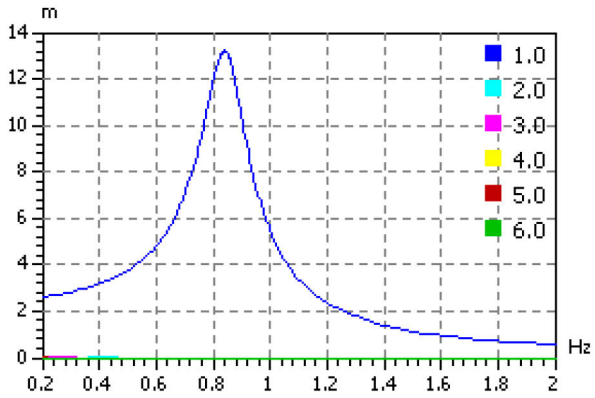
Die Unterscheidung zwischen linearem und nichtlinearem Verfahren betrifft vorrangig die Ermittlung der Verläufe der Zustandsgrößen in Abhängigkeit der Phase. Aus den Verläufen der Zustandsgrößen werden die Ergebnisverläufe berechnet. Bei den nichtlinearen Verfahren und beim linearen Verfahren ohne Interpolation werden die entsprechenden Zuweisungsgleichungen im Zeitbereich ausgewertet. Ergebnisgrößen, die stark nichtlinear vom oszillierenden Anteil des Zustandsgrößenverlaufes abhängen, werden dadurch auch korrekt berechnet.




**a)** Modell (Federsteife `c` von Auslenkung `dx` abhängig)




**b)** Realistisches Ergebnis der nichtlinearen Rechnung

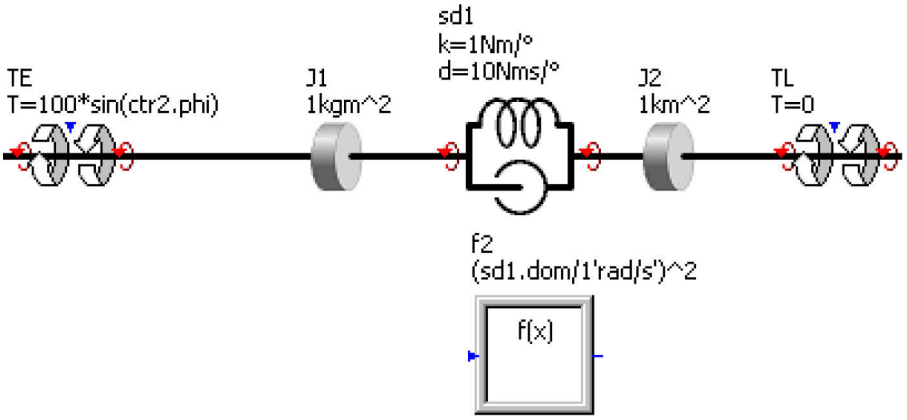



**c)** Stark abweichendes Ergebnis der linearen Rechnung

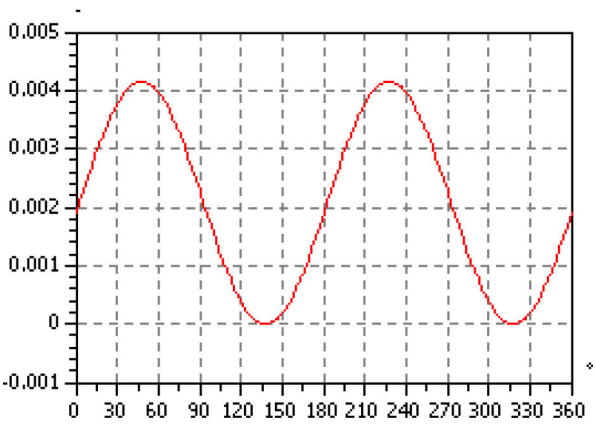
**Bild 6.33:** Vergleich der Ergebnisse der nichtlinearen und der linearen Rechnung bei einem Feder-Masse-Schwinger mit nichtlinearer Federsteifigkeit.

Das lineare Verfahren mit Interpolation ist für höhere Rechengeschwindigkeit ausgelegt. Bei ihm werden die Ergebnisverläufe nur für die große affin-lineare Zustandsbewegung des Systems im Zeitbereich ermittelt. Der durch die kleine Oszillation der Zustände hervorgerufenen Änderungen in den Ergebnisverläufen werden nachträglich linear überlagert (als Übertragungsmatrix wird dabei die über eine Periode gemittelte Jacobi-Matrix der Ergebnisgrößen nach den Zuständen genutzt). Dies kann zu Ungenauigkeiten in den Ergebnissen führen, wenn Ergebnisgrößen stark nichtlinear vom oszillierenden Signalanteil der Zustände abhängen. Ein Extrembeispiel dafür ist in **Bild 6.34** zu sehen. Dort sind die Winkelgeschwindigkeiten der Drehmassen J1 und J2 Zustandsgrößen und die Ergebnisgröße `f2.y` hängt quadratisch vom von der Differenzwinkelgeschwindigkeit ab.

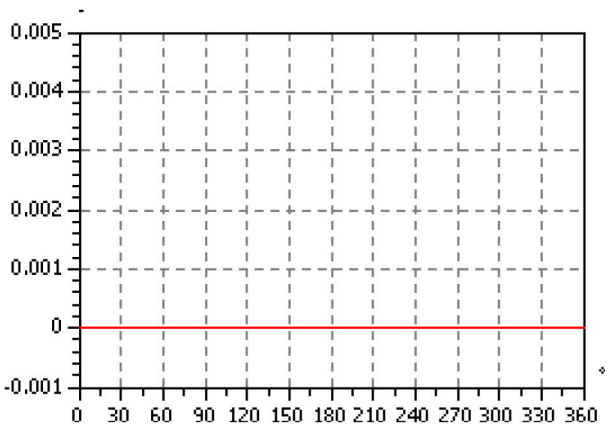
Die bei Antriebsstranganwendungen gewünschten Ergebnisgrößen sind meist linear oder schwach nichtlinear von den Oszillationen der Zustandsgrößen abhängig und die spektrale Leistung `PSpec` wird im Frequenzbereich ermittelt, so dass dieser Effekt normalerweise keine Rolle spielt. Sollte er sich doch einmal störend bemerkbar machen, so kann man mit Hilfe des erweiterten Parameters `bResultInterpol` die Ergebnisinterpolation beim linearen Verfahren mit Interpolation auf Kosten der Rechengeschwindigkeit abschalten.




**a)** Modell (Ergebnisgröße `f2.y` stark nichtlinear von `sd1.dom` abhängig)




**b)** Realistisches Ergebnis für `f2.y` ohne Ergebnisinterpolation




**c)** Stark abweichendes Ergebnis für `f2.y` mit Ergebnisinterpolation

**Bild 6.34:** Vergleich der Ergebnisse mit und ohne Ergebnisinterpolation.

## 6.3.5 Ergebnisfenster für die stationäre Simulation

Mit den folgenden Schritten kann man sich die Ergebnisfenster für die stationäre Analyse anzeigen lassen:

1. Aktivieren der Protokollattribute der Ergebnisgrößen
2. Öffnen der Ergebnisfenster
3. Spektren-Darstellung
4. Verlaufs-Darstellung

### 6.3.5.1 Aktivieren der Protokollattribute der Ergebnisgrößen

Aktivieren Sie vor der Simulation die Protokollattribute derjenigen Ergebnisgrößen, die sie sich anzeigen lassen möchten. Dabei gehen Sie analog der Bedienung bei der Transientenanalyse vor. Klicken Sie im Eigenschaftsdialog des entsprechenden Modellelements auf der Seite *Ergebnisgrößen* das Protokollattribut der gewünschten Ergebnisgröße an ( → ).

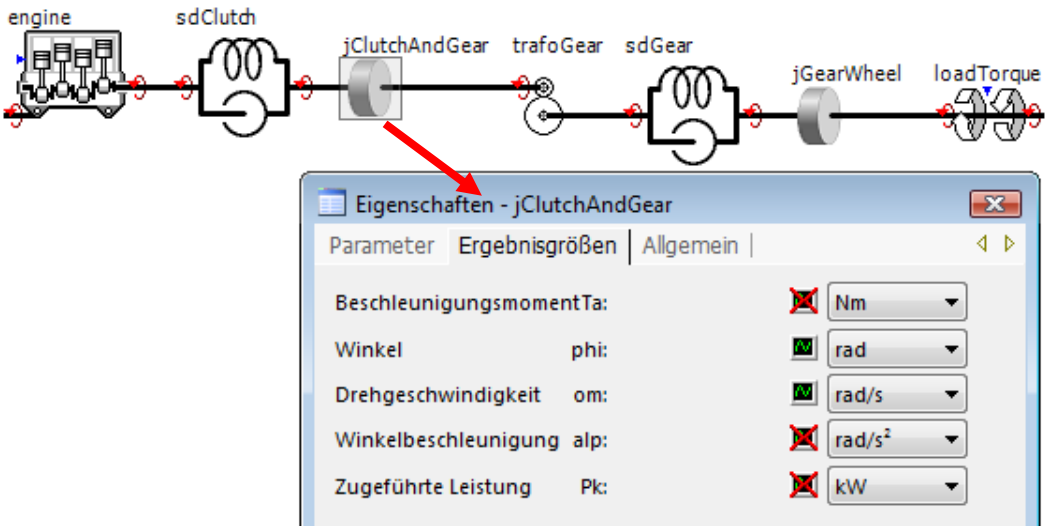



**Bild 6.35:** Aktivieren des Protokollattributs einer Ergebnisgröße über den Eigenschaftsdialog eines Modellelements (hier jClutchAndGear)

Öffnen der Ergebnisfenster

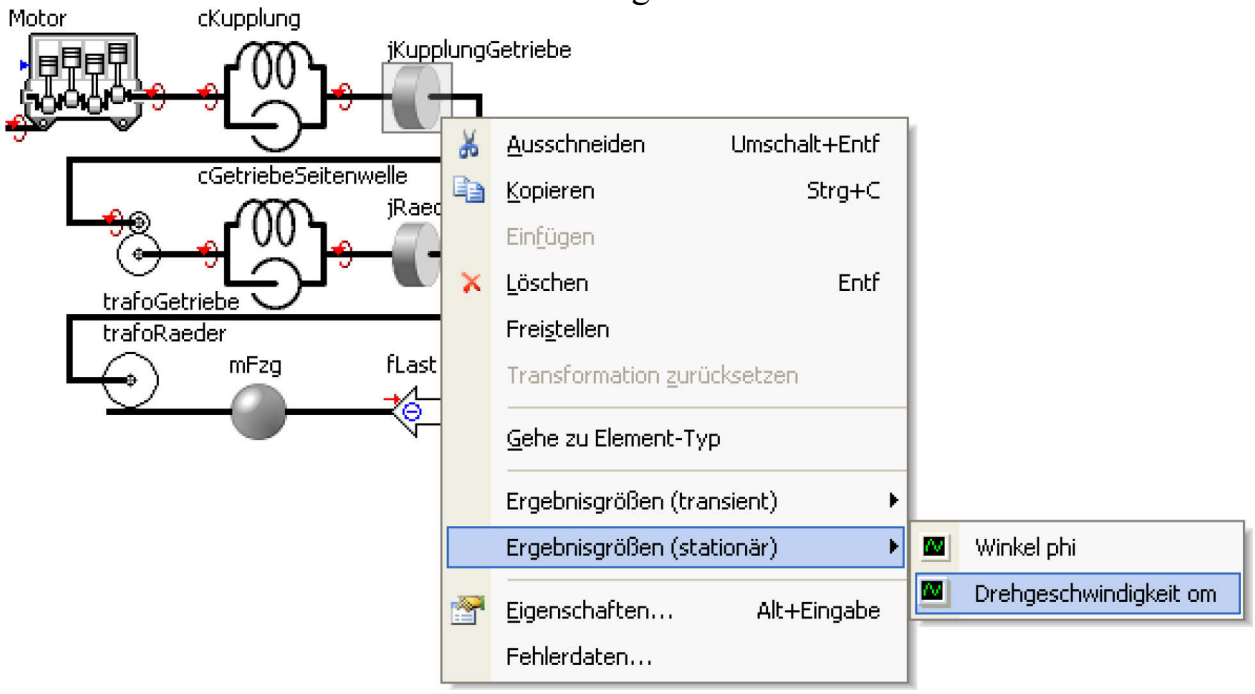



**Bild 6.36:** Öffnen des Ergebnisfensters für die stationäre Simulation

Zum Anzeigen des Ergebnisfensters einer protokollierten Ergebnisgröße wählen Sie im Kontextmenü (**Bild 6.36**) die entsprechenden ***Ergebnisgrößen (stationär)*** aus. Es öffnet sich dann das zu dieser Ergebnisgröße gehörige Ergebnisfenster für die Stationäre Simulation (siehe **Bild 6.37**).
Mit den rechts unten im Ergebnisfenster sichtbaren Reitern ***Spektren*** und ***Verlauf*** kann man zwischen der Darstellung

- der Spektren über der Referenzgröße (wie z.B. der mittleren Drehzahl) und
- des Verlaufs der Zeitsignals über der Periodenvariablen (wie z.B. die Schwankung der Drehgeschwindigkeit über dem Drehwinkel)

wählen.

Beim Öffnen eines Ergebnisfensters werden zuerst die Spektren angezeigt.

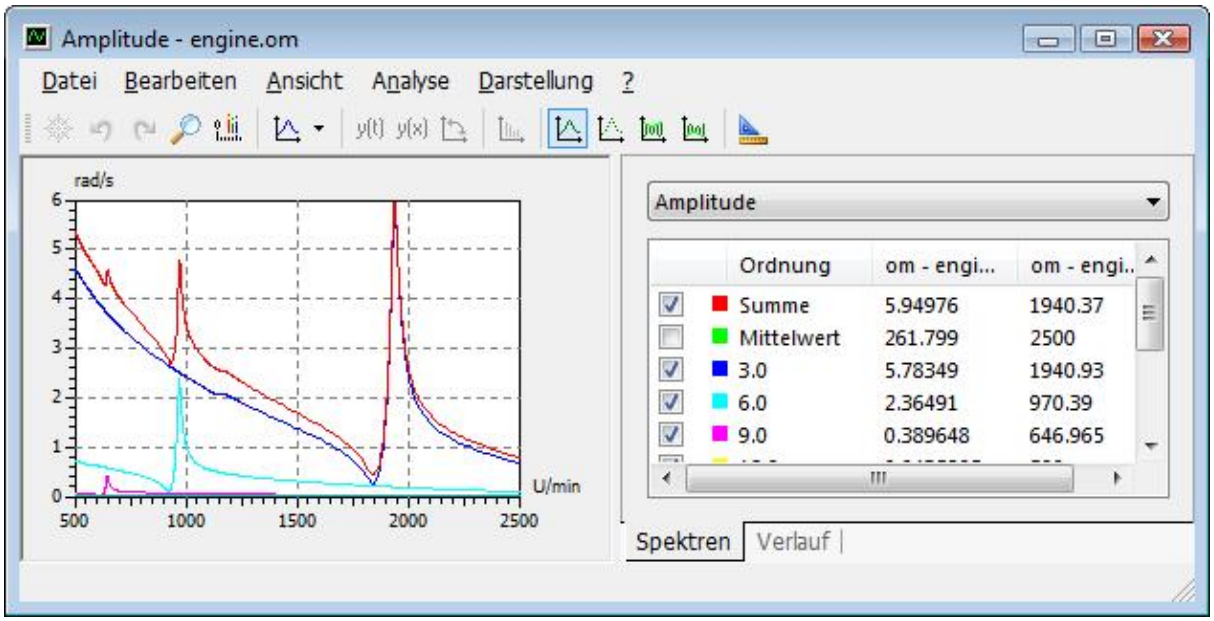



**Bild 6.37:** Ergebnisfenster für die stationäre Simulation

Für diese Ergebnisfenster stehen die Funktionen ***Einfrieren*** und ***Drag & Drop*** nicht zur Verfügung!

### 6.3.5.2 Spektren-Darstellung

Hat man die Spektren-Darstellung gewählt, stehen für die Ergebnisgröße ein Spektrum einschließlich Mittelwert und eine Summengröße zur Anzeige zur Verfügung. Mit der Auswahlbox rechts oben im Ergebnisfenster (siehe **Bild 6.38**) kann man festlegen, wie diese Größen berechnet werden sollen.

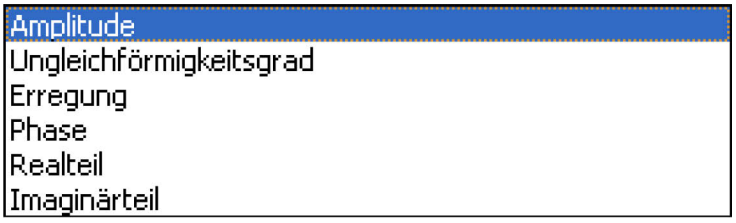



**Bild 6.38:** Auswahlbox für die Berechnung der Spektralkomponenten

| | Summen-größe | Mittel-wert | Spektral-komponenten (Ordnungen) | Bemerkungen/ Beispiele |
|---|---|---|---|---|
| **Amplitude** | $\max_{t\in[0,T]} \lvert \tilde{y}(t) \rvert$ | $\hat{y}[0]$ | $\hat{y}[k] = \sqrt{\hat{y}_R^2[k] + \hat{y}_I^2[k]}$ | Zum Beispiel beim Winkel einer Drehmasse wird als Summengröße die maximale Abweichung von der gleichförmigen Bewegung (ohne Schwingungserregung) dargestellt. |
| **Ungleichförmigkeitsgrad** | $\frac{\tilde{y}_{max} - \tilde{y}_m}{\hat{y}[0]}$<br>Für $\hat{y}[0] = 0$ wird diese Größe gleich null gesetzt. | 0 | $\frac{2\hat{y}[k]}{\hat{y}[0]}$<br>Für $\hat{y}[0] = 0$ wird diese Größe gleich null gesetzt. | Der Ungleichförmigkeitsgrad ist vor allem bei Drehmassen eines Antriebsstranges ein übliches Maß für die Schwankung der Winkelgeschwindigkeit um ihren Mittelwert. Er ist bei der Winkelgeschwindigkeit von Drehmassen als Berechnungsgrundlage voreingestellt. |
| **Erregung** | Bei der Berechnung der Erregergrößen $y$ wird für die Zustandsgrößen nur der Ansatz<br>$x(t) := x_P \frac{t}{T} + \hat{x}[0]$<br>genutzt.<br>Das heißt, das mittelwertfreie periodisch oszillierende Signal $\tilde{x}(t)$ wird bei der Berechnung der Erregergrößen null gesetzt.<br>Bei Ergebnisgrößen $y$, die linear von den Zustandsgrößen abhängig sind, sind die Erreger-Spektralkomponenten $\hat{y}[k]$ $(k = 1,\ldots,N)$ gleich null und werden nicht mit angezeigt. | | | Bei einer Drehmasse, die direkt an eine Schubkurbel mit starrem Pleuel und starrer Kurbelwelle angeschlossen ist, ist die Erregung für das Beschleunigungsmoment das in der Antriebstechnik übliche Erregermoment. |
| **Phase** | 0 | 0 | $\arctan(\hat{y}_I[k], \hat{y}_R[k])$ | Die Phasenlage bezieht sich auf die Nullphase der Periodenvariablen. |

| | | | | |
|---|---|---|---|---|
| **Realteil** | 0 | $\hat{y}[0]$ | $\hat{y}_R[k]$ | Der Realteil ist der Koeffizient zur Basisfunktion $\cos(\omega kt)$ im Ansatz. |
| **Imaginärteil** | 0 | 0 | $\hat{y}_I[k]$ | Der Imaginärteil ist der Koeffizient zur Basisfunktion $-\sin(\omega kt)$ im Ansatz. |

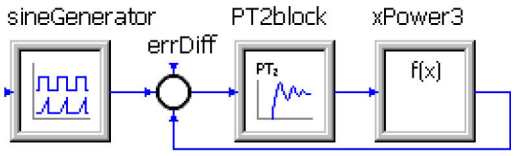
**Tabelle 6.2:** Auswahl der Spektrendarstelung

Im Panel der spektralen Darstellung sind die Summenkurve, der Mittelwert und die einzelnen Spektralkomponenten aufgelistet (siehe **Bild 6.40**).

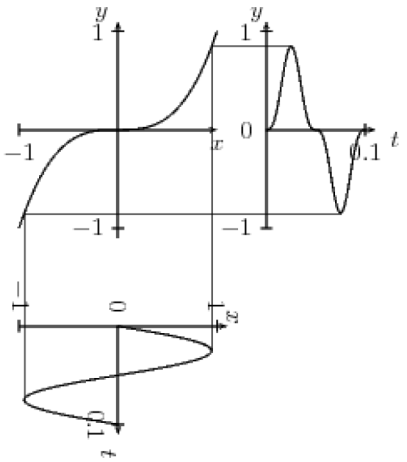
1. Mit den Schaltflächen in der ersten Spalte des Panels kann man die Anzeige jeder dieser Komponenten separat ein- und ausschalten.
2. In der zweiten Spalte des Panels sind die Ordnungen der Spektralkomponenten aufgelistet. Das sind die ganzzahlig Vielfachen der im System-Dialog eingestellten Grundordnung.
3. In der dritten Spalte ist für jede Spektralkomponente das Maximum aufgelistet.
4. In der vierten Spalte ist für jede Spektralkomponente aufgelistet, wo innerhalb des vom Anwender eingestellten Intervalls für die Bezugsgröße die Spektralkomponente ihr Maximum annimmt.

Bei einem vorgegebenen Referenzwert ergibt sich der Wert der Summenkurve aus dem maximalen Betrag des Zeitverlaufes der Auslenkung (Darstellung *Verlauf* im Ergebnisfenster für die stationäre Simulation).

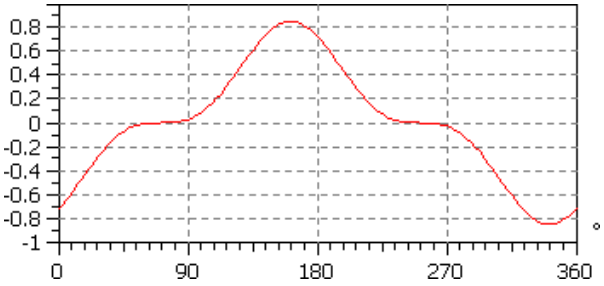
Im Beispiel von **Bild 6.39** die Grundordnung und die höchste Ordnung sind beide auf 1 eingestellt. Damit werden die Verläufe der Zustandsgrößen jeweils nur durch die Grundwelle approximiert. Das Ausgangssignal des nichtlinearen Signalgliedes *xPower5* ist so verzerrt, dass das Maximum des Verlaufes die Amplitude der Grundwelle überschreitet. Damit verläuft die Summenkurve oberhalb des Amplitudenganges für die Grundwelle.



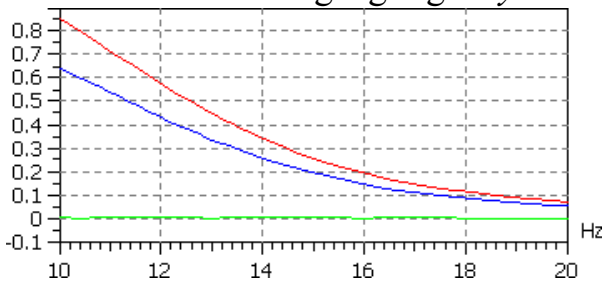

a) Modell




b) Verzerrung des sinusförmigen Signals x durch das nichtlineare Signalglied *xPower3* sowie das resultierende Ausgangssignal *y*




c) Verlauf des Ausgangssignals *y* von *xPower3* bei einer Erregerfrequenz von 100 Hz




d) Die Summenkurve (rot) liegt höher als die Amplitude der Grundwelle (blau)

**Bild 6.39:** Grundordnung und höchste Ordnung sind in diesem Beispiel beide auf 1 gesetzt. Trotzdem verläuft die Summenkurve deutlich oberhalb des Amplitudenganges der Grundwelle

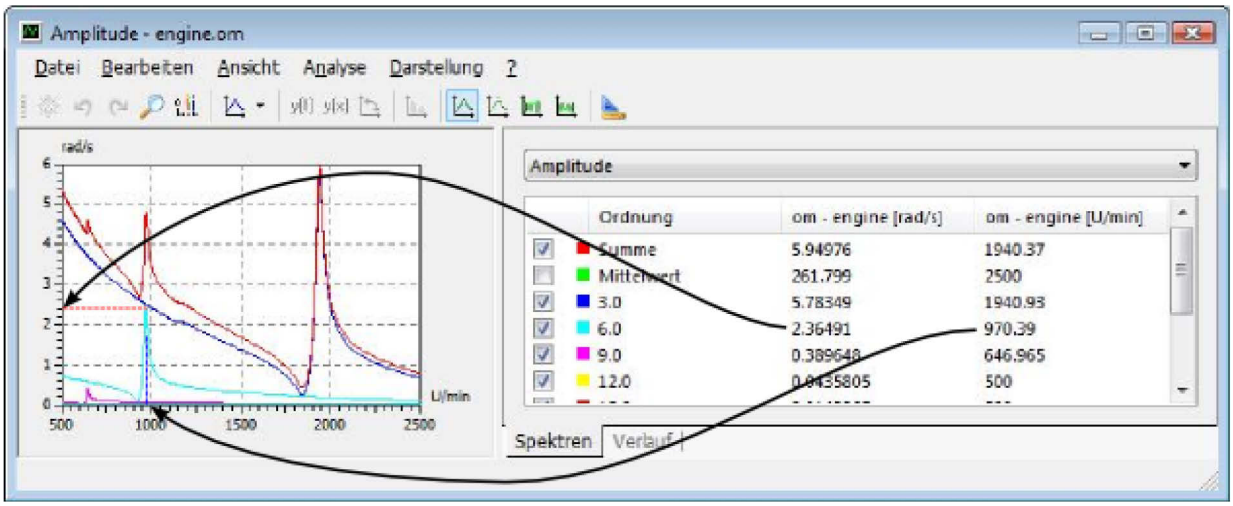



**Bild 6.40:** Das Panel in der spektralen Darstellung der Ergebnisse der stationären Analyse

Für die sechste Ordnung (hellblaue Kurve) ist exemplarisch dargestellt, an welcher Stelle die zugehörige Kurve ihr Maximum annimmt (blaue Markierung) und wie groß dieses Maximum ist (rote Markierung).

### 6.3.5.3 Verlaufs-Darstellung

**Bild 6.41** zeigt das Ergebnisfenster für die stationäre Analyse für den Fall, dass man die Verlaufsdarstellung gewählt hat.

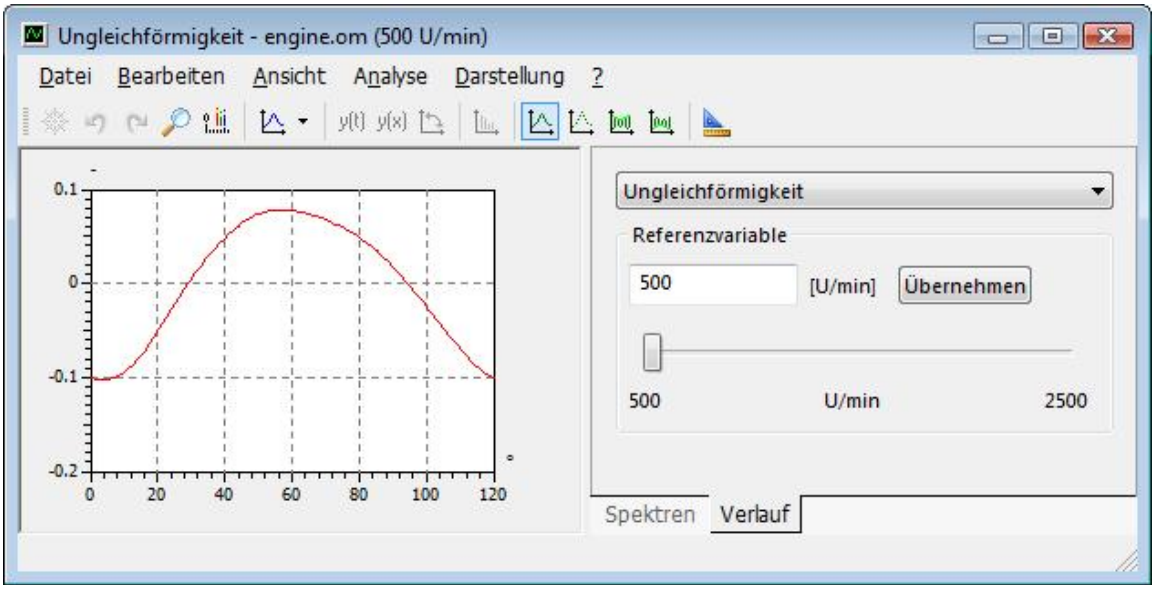



**Bild 6.41:** Ergebnisfenster für die Verlaufsdarstellung der Ergebnisse der stationären Analyse

In dem Ergebnisfenster ist die gewünschte Ergebnisgröße über einer Periode der Periodenvariablen aufgetragen.
In welcher Weise die Werte für die Ergebnisgröße berechnet werden, kann über die Auswahlbox in der rechten oberen Ecke des Ergebnisfensters festgelegt werden.

**Bild 6.42:** Auswahlbox für die Berechnung der Ergebnisgröße im Zeitbereich

Zur Auswahl stehen:

| | **Berechnungs-vorschrift** | **Bemerkungen/Beispiele** |
|---|---|---|
| **Auslenkung** | $\tilde{y}(t)$ | Oftmals ist der oszillierende Anteil einer Ergebnisgröße wesentlich kleiner als ihr Mittelwert. In diesem Fall ist es günstig, den oszillierenden Anteil separat darzustellen. Z.B. beim Winkel einer Drehmasse wird hier direkt die Abweichung |

| | | |
|---|---|---|
| | | des momentanen Winkels von dem bei der gleichförmigen Bewegung (ohne Schwingungserregung) angezeigt. |
| **Auslenkung +Mittelwert** | $\tilde{y}(t) + \hat{y}[0]$ | Für Größen, deren Komponente im Periodenvektor gleich null ist, ist dies das vollständige Zeitsignal (aufgetragen über der Periodenvariable). |
| **Ungleichförmigkeit** | $\frac{\tilde{y}(t)}{\hat{y}[0]}$ | Die Ungleichförmigkeit gibt an, wie groß die momentane relative Abweichung einer Größe von ihrem Mittelwert ist.<br>Bei Antriebssträngen wird oftmals für die Winkelgeschwindigkeiten die Ungleichförmigkeit angegeben. |

Den Wert der Bezugsgröße bei dem die Zeitbereichsergebnisse dargestellt werden sollen, stellt man entweder mit dem Schieberegler ein oder man gibt ihn direkt in das Eingabefeld ein und bestätigt die Eingabe mit ***Übernehmen***. Der Zeitverlauf wird dann für diesen Wert aus den protokollierten Spektren berechnet.

## 6.4 Gleichgewichtszustand (statisch/stationär)

Für bestimmte Untersuchungen ist es sinnvoll, ein Modell nicht aus einem durch die Anfangswerte vorgegebenen Zustand, sondern aus einem bestimmten Gleichgewichtszustand heraus zu untersuchen. Beispiele dafür sind:

- **Elektrotechnik**:
  DC-Arbeitspunkt einer Schaltung
- **Hydraulik**:
  Stationärer Zustand hydraulischer Antriebe
- **Mechanik:**
  Eingefedertes, elastisch gelagertes System, oder Fahrzeug fährt mit konstanter Geschwindigkeit und startet dann ein Manöver

Ein System befindet sich in einem Gleichgewichtszustand, wenn sich seine Zustandsgrößen nicht mehr ändern, d.h. wenn die Ableitungen der Zustandsgrößen Null sind. In der Mechanik sind jedoch auch „Gleichgewichtszustände“ interessant, bei denen sich das System mit konstanter Geschwindigkeit bewegt. Dann ist nur die Beschleunigung (also die höchste Ableitung) Null.

Systeme können:

- genau einen oder keinen Gleichgewichtszustand oder
- mehrere Gleichgewichtszustände (Pendel haben eine stabile, sowie eine instabile Gleichgewichtslage), eventuell sogar unendlich viele (z. B. Ball auf ebener Unterlage) haben.

Mit dem Menüpunkt ***Simulation/Gleichgewicht*** oder der Schaltfläche versetzen Sie das Modell in den Gleichgewichtszustand.

Der Berechnungsalgorithmus hält dabei die höchsten Ableitungen der Zustandsgrößen auf null und versucht unter diesen Voraussetzungen einen konsistenten Zustand herzustellen. Das entstehende nichtlineare Gleichungssystem wird mit einem speziellen iterativen Verfahren gelöst. Die eingegebenen Anfangswerte werden dabei als Startwerte für die Iteration genutzt.

Wird kein Gleichgewichtszustand gefunden, kann der Lösungsprozess durch eine Variation der Anfangswerte unterstützt werden. Es gibt jedoch auch Modelle, die keinen Gleichgewichtszustand besitzen, wie **Bild 6.43** zeigt.

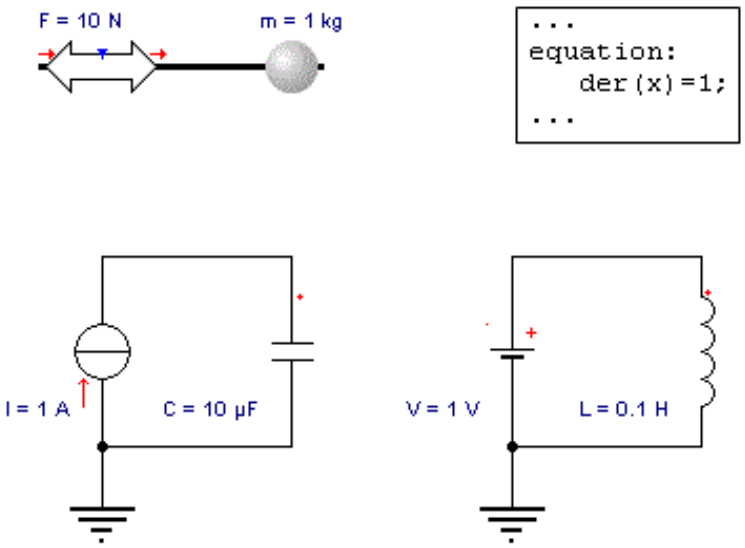



**Bild 6.43:** Einfache Modelle, die keinen Gleichgewichtszustand besitzen

Das Berechnungsverfahren kann nicht unterscheiden, ob ein Modell keinen Gleichgewichtszustand hat oder dieser nicht gefunden werden kann, weil z. B. die Startwerte ungünstig gewählt wurden.

Die Ergebnisse der Gleichgewichtsberechnung werden im Modellexplorer (für alle Variablen), in Ergebnisfenstern (erster Wert des Ergebnisverlaufes) oder mittels Momentananzeigen angezeigt.
Aus dem Gleichgewichtszustand heraus kann eine Simulation im Zeitbereich oder eine lineare Systemanalyse gestartet werden.

Bevor die Meldung „Der stationäre Zustand (statisches Gleichgewicht) konnte nicht gefunden werden!“ erscheint, wird wie bei Berechnung der konsistenten Anfangswerte zum Startzeitpunkt versucht, den Gleichgewichtszustand mit einer Einbettung zu berechnen, falls dies definiert ist( siehe **6.1.2.3)**.

## 6.5 Lineare Systemanalyse

Zusätzlich zur Berechnung im Zeitbereich stehen dem Anwender weitere Analysefunktionen zur Verfügung. Dazu gehören:

- Eigenfrequenzen und Schwingformen und das
- Übertragungsverhalten.

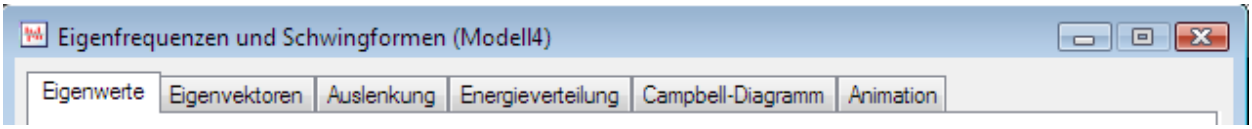
Bei der linearen Systemanalyse findet eine Linearisierung des Modells um den aktuellen Arbeitspunkt statt. Die Ergebnisse gelten also bei nichtlinearen Modellen nur innerhalb einer (kleinen) Umgebung um diesen Arbeitspunkt.

Die lineare Systemanalyse kann ausgelöst werden, wenn keine Simulationsrechnung läuft. Damit ist es beispielsweise möglich, ein nichtlineares Modell bis zu einem bestimmten Arbeitspunkt rechnen zu lassen und dort eine Analyse durchzuführen.

### 6.5.1 Eigenfrequenzen und Schwingformen

Die Eigenfrequenzen und Schwingformen des Systems werden aus den Jacobimatrizen des Modells berechnet. Es werden alle Zustandsgrößen berücksichtigt. Die berechneten Werte gelten also für das gesamte Modell unabhängig von der physikalischen Domäne.

Eine Eigenfrequenzanalyse lösen Sie über ***Analyse/Eigenfrequenzen und Schwingformen*** oder den Button aus. Die Berechnung wird durchgeführt, danach öffnet sich ein Fenster zur Anzeige der Werte in Tabellenform.




Zur Anzeige aller Werte ist das Fenster in seiner Größe veränderbar. Bei weiteren Bedienhandlungen kann es geöffnet bleiben. Damit können Sie komfortabel den Einfluss einzelner Parameter auf bestimmte Eigenwerte beobachten. Verändern Sie einfach den Parameter und berechnen Sie die Eigenwerte mit der Schaltfläche ***Aktualisieren*** neu.

Im Fenster sind folgende Schaltflächen vorhanden:

| | |
|---|---|
| Druckansicht | Betätigen Sie diese Schaltfläche, um die Ergebnisse der Eigenfrequenzanalyse zu drucken oder als Datei zu exportieren. In der Druckansicht können Sie nicht interessierende Ergebnisse ausblenden, sowie verschiedene Formateinstellungen vornehmen. |

Aktualisieren — Diese Schaltfläche löst eine erneute Berechnung aus. Die Aktualisierung ist erforderlich, wenn Parameter geändert wurden und der Einfluss dieser Parameter auf bestimmte Eigenwerte untersucht werden soll.

Schließen — Betätigen Sie diese Schaltfläche um das Fenster für die Berechnung der Eigenfrequenzen und Schwingformen zu schließen.

Hilfe — Über diese Schaltfläche gelangen Sie zur Hilfe für die Arbeit mit diesem Fenster.

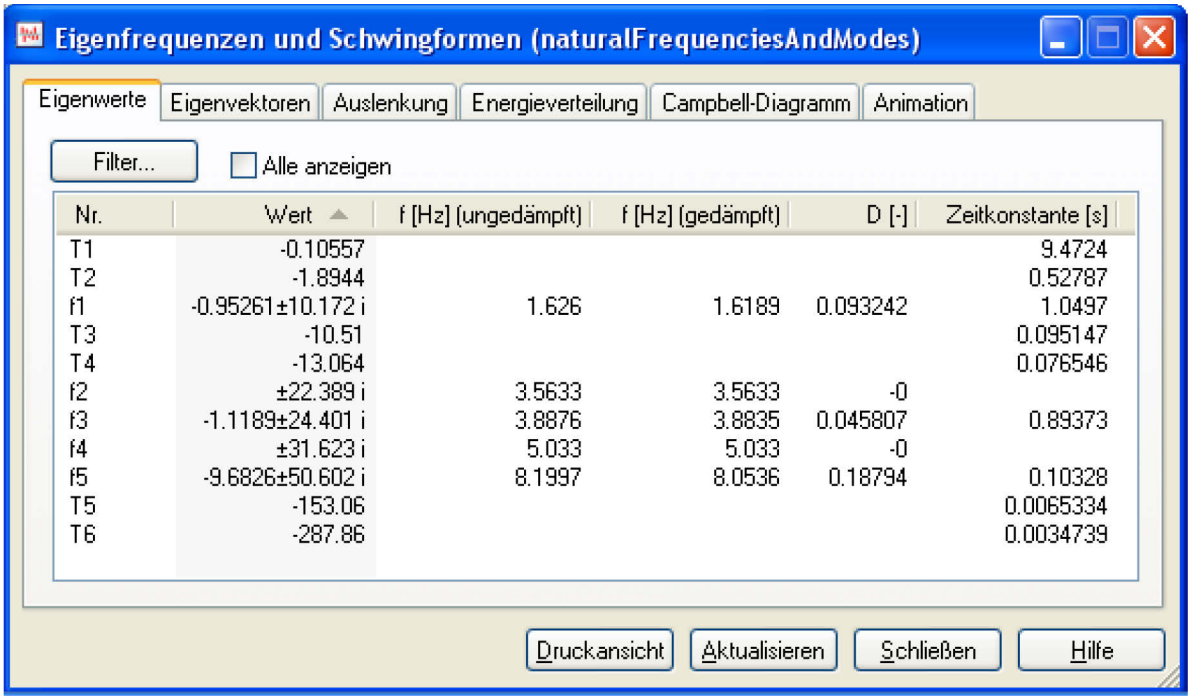
Die Eigenfrequenzen und Schwingformen werden auf den folgenden Dialogseiten angezeigt.

***Filter***

Mit der Filterfunktion können Sie Wertebereiche für **f** und **T** , sowie Werte von Null und Unendlich ausblenden.

### 6.5.1.1 Eigenwerte

Auf dieser Dialogseite werden die komplexen Eigenwerte, die gedämpften und ungedämpften Eigenfrequenzen, die modalen Dämpfungsgrade D, sowie die Zeitkonstanten für das aktive Modell dargestellt.




| Nr. | Wert | f [Hz] (ungedämpft) | f [Hz] (gedämpft) | D [-] | Zeitkonstante [s] |
|---|---|---|---|---|---|
| T1 | -0.10557 | | | | 9.4724 |
| T2 | -1.8944 | | | | 0.52787 |
| f1 | -0.95261±10.172 i | 1.626 | 1.6189 | 0.093242 | 1.0497 |
| T3 | -10.51 | | | | 0.095147 |
| T4 | -13.064 | | | | 0.076546 |
| f2 | ±22.389 i | 3.5633 | 3.5633 | -0 | |
| f3 | -1.1189±24.401 i | 3.8876 | 3.8835 | 0.045807 | 0.89373 |
| f4 | ±31.623 i | 5.033 | 5.033 | -0 | |
| f5 | -9.6826±50.602 i | 8.1997 | 8.0536 | 0.18794 | 0.10328 |
| T5 | -153.06 | | | | 0.0065334 |
| T6 | -287.86 | | | | 0.0034739 |

**Bild 6.44:** Eigenwerte, Eigenfrequenzen, Dämpfungsgrade und Zeitkonstanten

Für einen gegebenen Eigenwert λ bestimmen sich die Einträge der entsprechenden Tabellenzeile wie folgt:

1. Die Zeile ist in der ersten Spalte als Zeitkonstante T markiert, wenn **λ** reell ist und als Frequenz f, wenn λ komplex ist.
2. Komplexe Eigenwerte werden mit Hilfe der imaginären Einheit ί dargestellt. Komplex konjugierte Eigenwerte werden jeweils innerhalb einer Zeile kombiniert. Dies wird durch das Zeichen ± vor dem Imaginärteil angedeutet.

3. Die ungedämpfte Eigenfrequenz ergibt sich zu

$$\frac{|\lambda|}{2 \cdot \pi} \qquad \textbf{(6.15)}$$

   wobei |λ| der Betrag von λ ist. Sie wird nur angezeigt, wenn λ komplex ist.

4. Die gedämpfte Eigenfrequenz ergibt sich

$$\frac{|im(\lambda)|}{2 \cdot \pi} \qquad \textbf{(6.16)}$$

   wobei *im*|λ| der Betrag von λ ist. Sie wird nur angezeigt, wenn λ komplex ist.

5. Der modale Dämpfungsfaktor D ergibt sich zu

$$\frac{-re(\lambda)}{|\lambda|} \qquad \textbf{(6.17)}$$

   Er wird nur angezeigt, wenn λ komplex ist.

6. Die Zeitkonstante ergibt sich zu

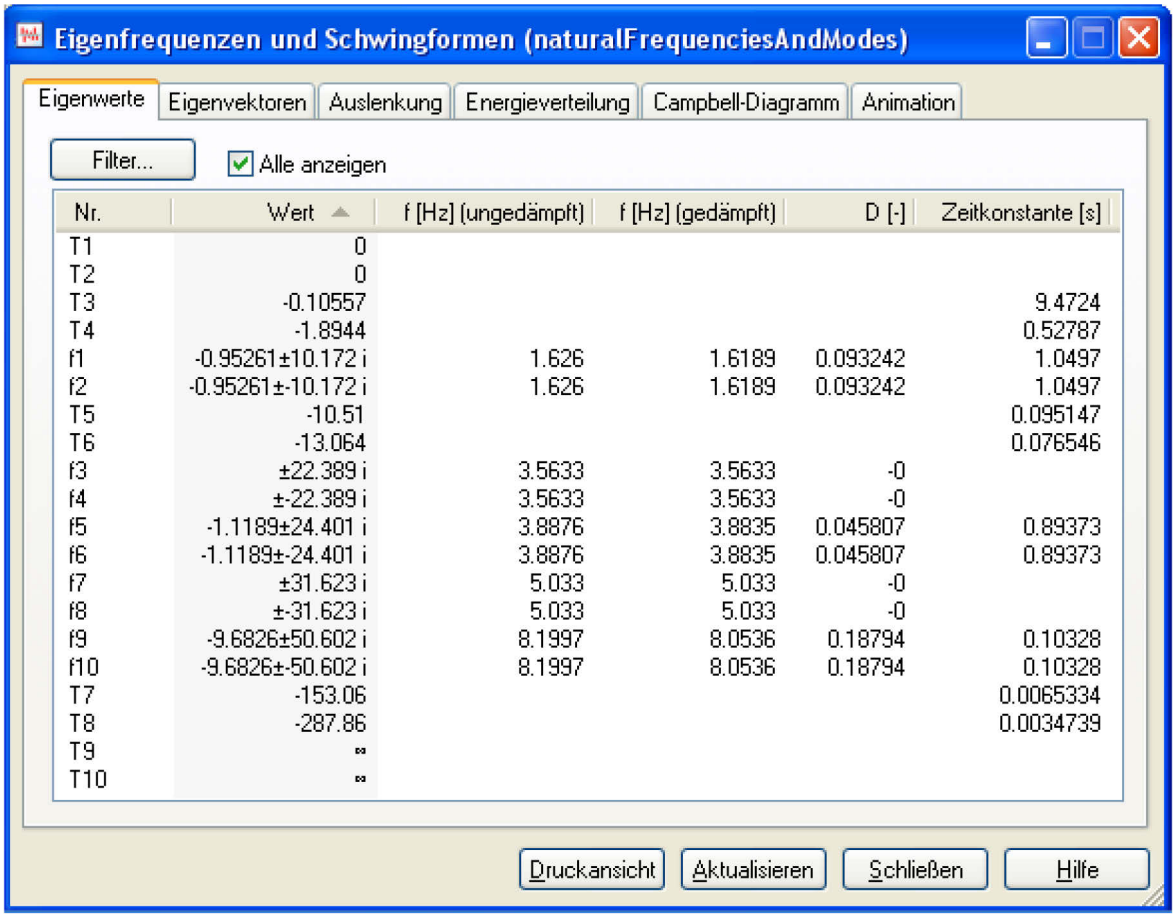
$$-\frac{1}{\lambda} \qquad \textbf{(6.18)}$$

   Sie wird nur angezeigt, wenn λ reell und nicht 0 ist.

Ist ***Alle anzeigen*** aktiviert, so werden auch unendliche Eigenwerte und Eigenwerte gleich null angezeigt (Siehe **Bild 6.45**) und belegen Zeilen der Tabelle.

Durch Klicken auf den Spaltenkopf werden die Eigenwerte entsprechend der selektierten Spalte sortiert. Bei jedem erneuten Klick auf den Spaltenkopf wird die Sortierreihenfolge zwischen aufsteigend und absteigend umgeschaltet.

Um den Inhalt einer Tabelle im Textformat in die Zwischenablage zu kopieren, wählen Sie im Kontextmenü der Tabelle den Befehl ***Tabelle kopieren***.




| Nr. | Wert | f [Hz] (ungedämpft) | f [Hz] (gedämpft) | D [-] | Zeitkonstante [s] |
|---|---|---|---|---|---|
| T1 | 0 | | | | |
| T2 | 0 | | | | |
| T3 | -0.10557 | | | | 9.4724 |
| T4 | -1.8944 | | | | 0.52787 |
| f1 | -0.95261±10.172 i | 1.626 | 1.6189 | 0.093242 | 1.0497 |
| f2 | -0.95261±-10.172 i | 1.626 | 1.6189 | 0.093242 | 1.0497 |
| T5 | -10.51 | | | | 0.095147 |
| T6 | -13.064 | | | | 0.076546 |
| f3 | ±22.389 i | 3.5633 | 3.5633 | -0 | |
| f4 | ±-22.389 i | 3.5633 | 3.5633 | -0 | |
| f5 | -1.1189±24.401 i | 3.8876 | 3.8835 | 0.045807 | 0.89373 |
| f6 | -1.1189±-24.401 i | 3.8876 | 3.8835 | 0.045807 | 0.89373 |
| f7 | ±31.623 i | 5.033 | 5.033 | -0 | |
| f8 | ±-31.623 i | 5.033 | 5.033 | -0 | |
| f9 | -9.6826±50.602 i | 8.1997 | 8.0536 | 0.18794 | 0.10328 |
| f10 | -9.6826±-50.602 i | 8.1997 | 8.0536 | 0.18794 | 0.10328 |
| T7 | -153.06 | | | | 0.0065334 |
| T8 | -287.86 | | | | 0.0034739 |
| T9 | ∞ | | | | |
| T10 | ∞ | | | | |

**Bild 6.45:** Darstellung der Eigenwerte mit ***Alle anzeigen*** aktiv

### 6.5.1.2 Eigenvektoren

Darstellung aller Eigenvektoren und der Namen der entsprechenden Zustandsgröße. Durch Klicken auf eine Zeile wird das zugehörige Element bzw. die zugehörige Verbindung in der Strukturansicht sowie im Modellexplorer selektiert. Um den Inhalt einer Tabelle im Textformat in die Zwischenablage zu kopieren, wählen Sie im Kontextmenü der Tabelle den Befehl ***Tabelle kopieren***.

### 6.5.1.3 Auslenkung

Darstellung eines Eigenvektors als Wert sowie in Form eines Balkendiagramms. Durch Klicken auf eine Zeile wird das zugehörige Element bzw. die zugehörige Verbindung in der Strukturansicht und im Modellexplorer selektiert.
Um den Inhalt einer Tabelle im Textformat in die Zwischenablage zu kopieren, wählen Sie im Kontextmenü der Tabelle den Befehl ***Tabelle kopieren***.

Wählen Sie hier den darzustellenden Eigenvektor aus. Die Reihenfolge der Einträge in der Combobox entspricht der Sortierung auf der Dialogseite Eigenwerte.

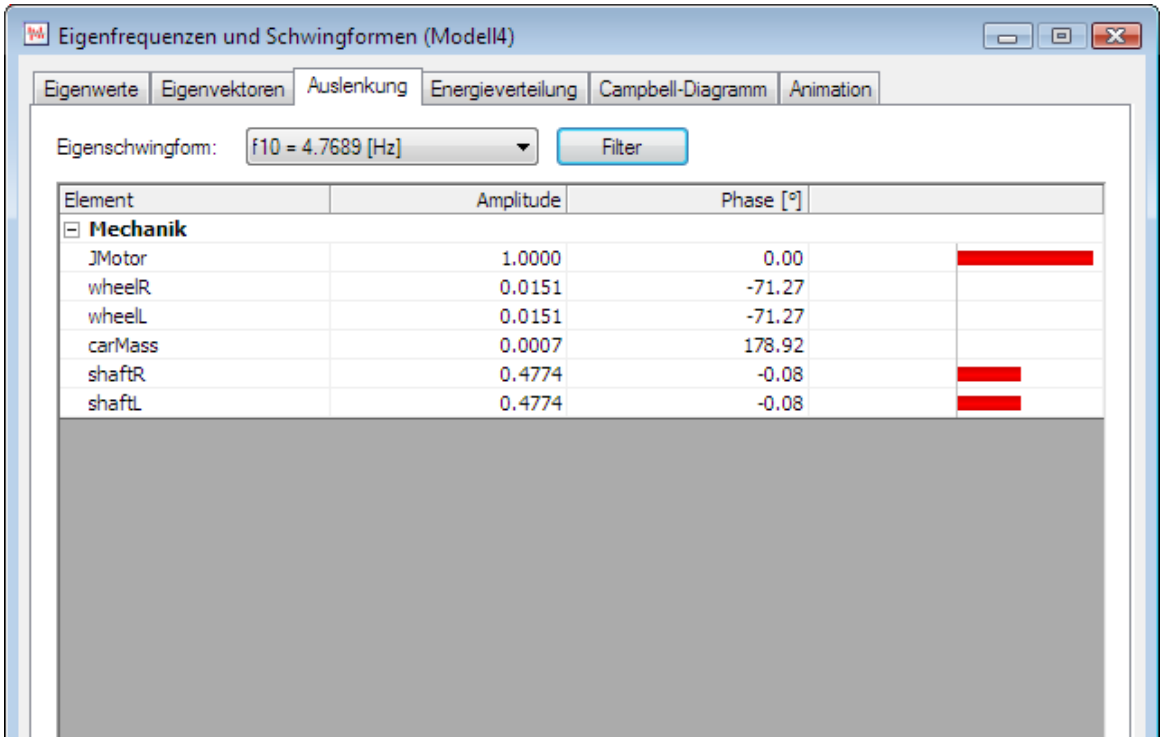



**Bild 6.46:** Ansicht Auslenkung

## ***Filter***

Die Normierung der im Balkendiagramm dargestellten Auslenkungen erfolgt nach der Regel:

$$\max(|y|) = 1 \qquad \textbf{(6.19)}$$

Bei der Normierung werden nur die aktuell dargestellten Zustandsgrößen berücksichtigt.

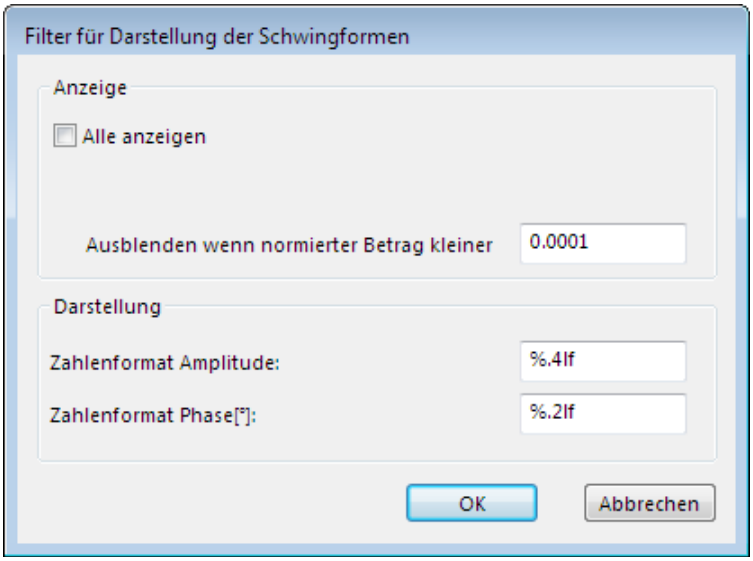



**Bild 6.47:** Filter für die Darstellung der Schwingformen

### 6.5.1.4 Energieverteilung

Auf dieser Dialogseite erfolgt die Darstellung der Energieverteilung bei einer Eigenschwingform. Damit ist der Einfluss bestimmter Elemente auf die jeweilige Eigenfrequenz ablesbar. Durch die Auswahl einer Tabellenzeile wird das betreffende Element in den geöffneten Strukturansichten sowie im Modellexplorer markiert. Damit ein Element bei der Energieberechnung berücksichtigt wird, müssen in der Verhaltensbeschreibung spezielle Variablen berechnet werden. Um den Inhalt einer Tabelle im Textformat in die Zwischenablage zu kopieren, wählen Sie im Kontextmenü der Tabelle den Befehl Tabelle kopieren.

Die Energieverteilung ist Bestandteil der Druckausgabe. Wählen Sie hier die Eigenschwingform, für die die Energieverteilung dargestellt werden soll. Die Reihenfolge der Einträge in der Combobox entspricht der Sortierung der Eigenwerte auf der Dialogseite ***Eigenwerte***.

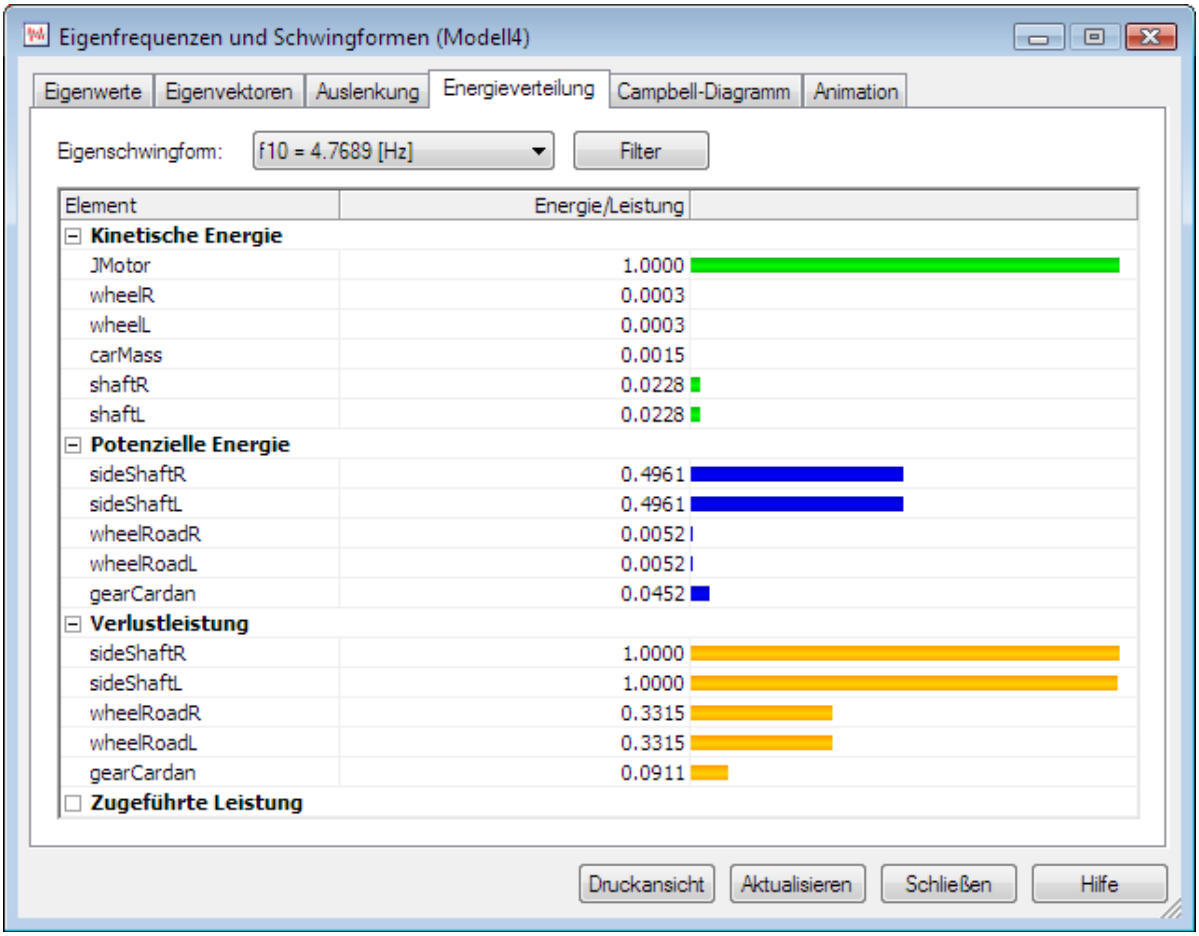



**Bild 6.48:** Ansicht Energieverteilung

***Kinetische Energie***

In dieser Tabelle wird die normierte Verteilung der kinetischen Energie $E_{kin}$ für die aktuell gewählte Eigenschwingform dargestellt. Die Tabelle enthält alle Elemente, die kinetische Energie speichern können (z.B. Element Masse, Bibliothek Mechanik).

***Potenzielle Energie***
In dieser Tabelle wird die normierte Verteilung der potenziellen Energie $E_{pot}$ für die aktuell gewählte Eigenschwingform dargestellt. Die Tabelle enthält alle Elemente die potenzielle Energie speichern können (z.B. Element Feder, Bibliothek Mechanik).

***Verlustleistung***
In dieser Tabelle wird die normierte Verteilung der Verlustleistung $E_{loss}$ für die aktuell gewählte Eigenschwingform dargestellt. Die Tabelle enthält alle Elemente die Verlustleistung abgeben können (z.B. Element Dämpfer, Bibliothek Mechanik).

Die Normierung der dargestellten Energieformen erfolgt nach den Regeln:

$$\sum E_{kin} = \sum E_{pot} \tag{6.20}$$

$$\max\left(E_{kin}, E_{pot}\right) = 1 \tag{6.21}$$

$$\max\left(E_{loss}\right) = 1 \tag{6.22}$$

***Filter***

Betätigen Sie diese Schaltfläche, um über einen Auswahldialog nicht interessierende Größen auszublenden sowie weitere Einstellungen vorzunehmen.

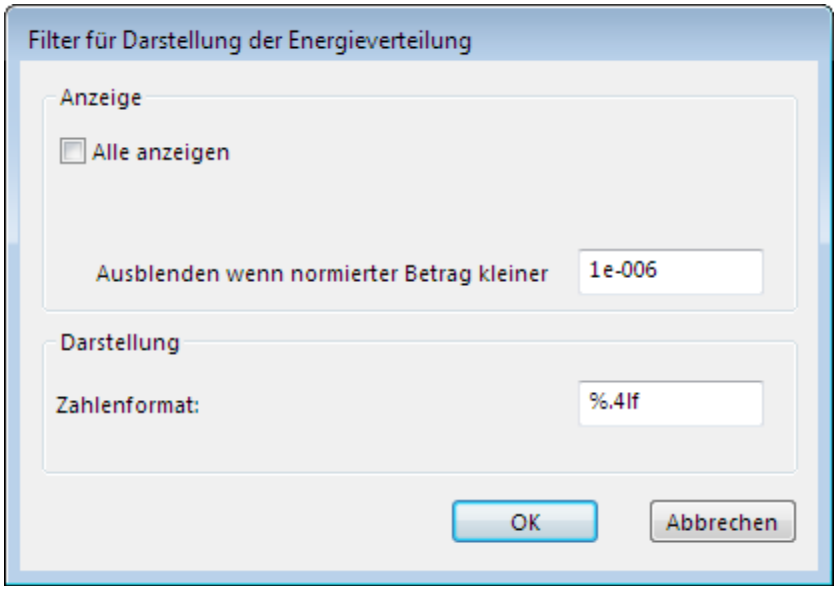



**Bild 6.49:** Filter für die Darstellung der Energieverteilung

### 6.5.1.5 Campbell-Diagramm

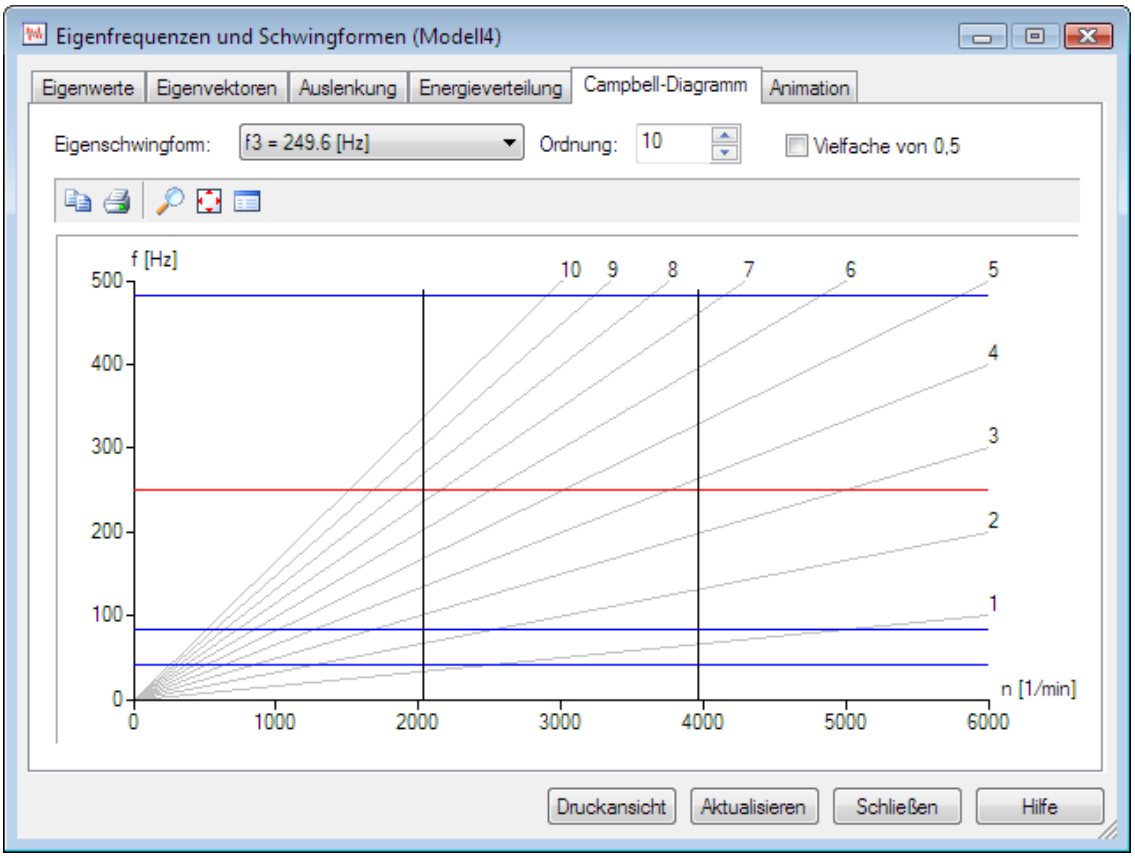



**Bild 6.50:** Campbell-Diagramm

Auf dieser Dialogseite wird das Campbell-Diagramm (oder auch Drehzahlschaubild) dargestellt. Es zeigt an, bei welchen Drehzahlen mit Resonanzen zu rechnen ist. Mögliche Resonanzstellen ergeben sich an Schnittpunkten der Eigenfrequenzen (waagerechte Linien) mit den Ordnungen (schräge Linien). Ob und wie stark die Resonanzen an diesen Stellen in Erscheinung treten, hängt von weiteren Faktoren ab (z. B. Art der Erregung, Größe der Dämpfung) und ist Ergebnis der Simulationsrechnung.

Die Schaltflächen haben folgende Bedeutung:

Kopiert das Campbell-Diagramm in die Zwischenablage

Druckt das Campbell-Diagramm

Lupenfunktion zur Auswahl eines Drehzahlbereiches

Anzeige des vollständigen Campbell-Diagramm

Die Schnittstellen können ausgemessen werden (mit der Maus auf die Schnittstelle klicken). Ein interessierender Drehzahlbereich ist mit zwei Markern (senkrechte Linien) markierbar.

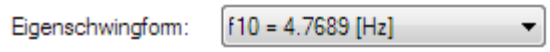
Die Marker können grafisch mit der Maus oder wertmäßig, nach Betätigung nebenstehender Schaltfläche, positioniert werden. Der

darzustellende Bereich lässt sich mit der Lupenfunktion oder wertmäßig gleichfalls über nebenstehende Schaltfläche angeben.

#### 6.5.1.6 Animation

Die Animation der Eigenschwingformen gestattet in der 3D-Ansicht die räumliche Zuordnung der Eigenfrequenzen sowie die Lokalisierung der entsprechenden Elemente.

***Eigenschwingform***




Wählen Sie hier die Eigenschwingform, die animiert werden soll. Die Reihenfolge der Einträge in der Combobox entspricht der Sortierung der Eigenwerte auf der Dialogseite **Eigenwerte**. Die Auswahl der Eigenschwingform kann auch während der laufenden Animation erfolgen.

***Verstärkung***

Mit dem Schieberegler können Sie einen Verstärkungsfaktor für die Auslenkung der animierten Eigenschwingform einstellen. Die Verstärkung kann während der laufenden Animation geändert werden.

***Animationsfrequenz***

Mit dem Schieberegler können Sie die Animationsgeschwindigkeit beeinflussen. Bei einer Animationsfrequenz von 1 wird eine Schwingungsperiode pro Sekunde wiedergegeben. Die Animationsfrequenz kann während der laufenden Animation geändert werden.

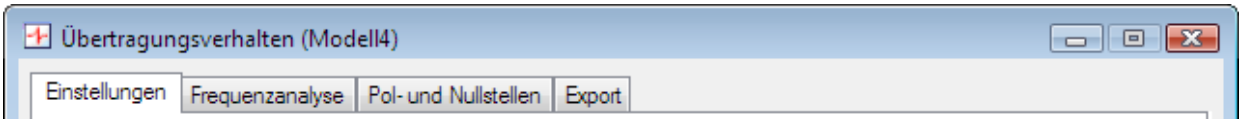
***Start***
Betätigen Sie diese Schaltfläche um die Animation zu starten. Während der Animation können Sie Verstärkung und Animationsfrequenz einstellen sowie die aktive Eigenschwingform umschalten.

***Stopp***
Betätigen Sie diese Schaltfläche um die laufende Animation zu beenden.

***Speichern***
Betätigen Sie diese Schaltfläche, um die Animation für die aktive Eigenschwingform als Video-Datei zu speichern. Ist mehr als eine 3D-Ansicht geöffnet, müssen Sie zunächst die gewünschte Ansicht aus einer Liste auswählen.

## 6.5.2 Übertragungsverhalten

Bei der Analyse des Übertragungsverhaltens wird das Modell als ein System mit Ein- und Ausgängen interpretiert. Nach Festlegung der Ein- und Ausgänge durch den Nutzer werden die Modellgleichungen im aktuellen Arbeitspunkt linearisiert.




Zur Anzeige aller Werte ist das Fenster in seiner Größe veränderbar. Das Fenster kann bei weiteren Bedienhandlungen geöffnet bleiben. Damit können Sie komfortabel den Einfluss einzelner Parameter untersuchen.

Aktualisieren — Betätigen Sie diese Schaltfläche, um die lineare Systemanalyse erneut durchzuführen. Die Aktualisierung ist erforderlich, wenn Parameter geändert wurden und der Einfluss dieser Parameter auf die Berechnungsergebnisse untersucht werden soll.

Schließen — Betätigen Sie diese Schaltfläche, um das Fenster für das Übertragungsverhalten zu schließen.

Hilfe — Über diese Schaltfläche gelangen Sie zur Hilfe für die Arbeit mit diesem Fenster.

### 6.5.2.1 Einstellungen

Die Dialogseite **Einstellungen** gestattet die Auswahl von Ein- und Ausgängen zur Durchführung einer linearen Systemanalyse. Als Eingang lassen sich freie Signaleingänge festlegen. Als Ausgang können sowohl Signalausgänge als auch Ergebnisgrößen gewählt werden.

***Bearbeiten***
Betätigen Sie die jeweilige Schaltfläche, um Ein- bzw. Ausgänge hinzuzufügen oder zu löschen. Es wird ein Dialog angezeigt, der die Auswahl von zulässigen Ein- bzw. Ausgängen aus dem aktiven Modell gestattet. Aus der Baumansicht des Modells können die Größen per *Drag & Drop* oder *Doppelklick* übernommen werden. Dabei dienen Signaleingänge des Modells als zulässige Eingangsgrößen und Ergebnisgrößen als zulässige Ausgangsgrößen der Analyse des Übertragungsverhaltens.

#### 6.5.2.2 Frequenzanalyse

Die Frequenzanalyse gestattet die Darstellung des Modellverhaltens über der Frequenz. Um eine Frequenzanalyse durchführen zu können, sind folgende Arbeitsschritte notwendig:

1. Auswahl der Ein- und Ausgänge auf der Dialogseite Einstellungen
2. Beschreibung der Erregung für jeden Eingang
3. Festlegung des Frequenzbereichs sowie von Anzahl und Verteilung der Samples zur Durchführung der Analyse
4. Ausführung der Analyse durch Betätigung der Schaltfläche Aktualisieren

***Erregungen***

Für jeden definierten Eingang muss eine Erregung vorgegeben werden. Dazu ist der betreffende Eingang in der Tabelle zu markieren. Für die Beschreibung der Erregung gibt es verschiedene Möglichkeiten:

- Betrag und Phase als Konstante (Standardeinstellung mit Betrag 1 und Phase 0)
- Real- und Imaginärteile als Konstante
- Betrag und Phase als Kennlinie über der Frequenz
- Real- und Imaginärteile als Kennlinie über der Frequenz
- Periodisches Zeitsignal

***Analyse***

Vor Durchführung einer Analyse ist der zu untersuchende Frequenzbereich durch folgende Parameter zu beschreiben:

- Minimale Frequenz in Hz
- Maximale Frequenz in Hz
- Anzahl der Samples
- Teilung der Frequenzachse (linear bzw. logarithmisch)

***Ergebnisse***

Die Tabelle Ergebnisse enthält für jeden festgelegten Ausgang die Systemantwort in drei verschiedenen Darstellungsarten:

- Real und Imaginärteil
- Betrag und Phase
- Nyquist-Plot

Mit *Drag & Drop* oder *Doppelklick* können die Symbole aus der Ergebnistabelle als Ergebnisfenster in den SimulationX-Arbeitsbereich gezogen werden. Die so geöffneten Ergebnisfenster werden mit dem Modell gespeichert und bei einer weiteren Frequenzanalyse automatisch aktualisiert.

Aktualisieren

Betätigen Sie diese Schaltfläche, um nach Parameteränderungen im Modell oder nach Änderung von Einstellungen für die Frequenzanalyse die Berechnungsergebnisse zu aktualisieren. Bereits geöffnete Ergebnisfenster mit Frequenzverläufen werden dabei automatisch aktualisiert.

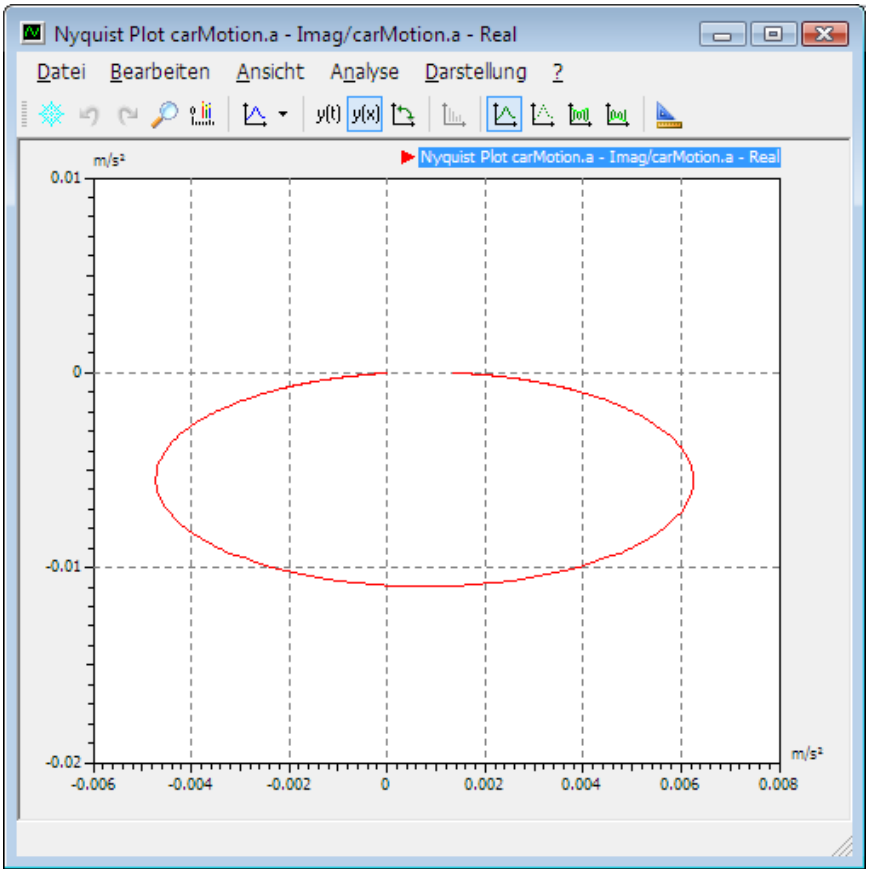



**Bild 6.51:** Ergebnisdarstellung als Nyquist-Plot

### 6.5.2.3 Pol- und Nullstellen

Auf dieser Dialogseite erfolgt die Darstellung der Pole und Nullstellen des Übertragungssystems als Tabelle sowie in grafischer Form als Pol-/Nullstellenplan. Zunächst müssen auf der Dialogseite ***Einstellungen*** die Ein- und Ausgänge des Systems bestimmt werden.
Pole und Nullstellen können sowohl in der Tabelle als auch innerhalb der grafischen Darstellung mit einem Mausklick markiert werden. Mehrfachselektion ist bei gedrückter *Strg*-Taste möglich. Markierte Pole bzw. Nullstellen werden in der Grafik farblich hervorgehoben.

Die Symbolleiste der Dialogseite beinhaltet folgende Schaltflächen:

Kopiert sowohl die Tabelle im Textformat als auch die grafische Darstellung in die Zwischenablage. Über den Befehl „*Inhalt einfügen*“ können diese Darstellungen so in andere Programme übernommen werden.

Öffnet die Druckansicht. Damit können die Tabelle sowie die grafische Darstellung ausgedruckt oder exportiert werden.

Zeigt eine Lupe an. Durch Markieren eines rechteckigen Bereichs innerhalb des Pol-/Nullstellenplans kann ein Ausschnitt der Grafik vergrößert dargestellt werden.

Setzt die Ausschnittsvergrößerung zurück. Es werden alle darstellbaren Pole und Nullstellen angezeigt.

Aktualisieren

Betätigen Sie diese Schaltfläche, um nach Parameteränderungen im Modell die Berechnungsergebnisse zu aktualisieren. Geöffnete Ergebnisfenster mit Frequenzkurven werden automatisch aktualisiert.

Schließen

Betätigen Sie diese Schaltfläche, um das Fenster für das Übertragungsverhalten zu schließen.

Hilfe

Über diese Schaltfläche gelangen Sie zur Hilfe für die Arbeit mit diesem Fenster.

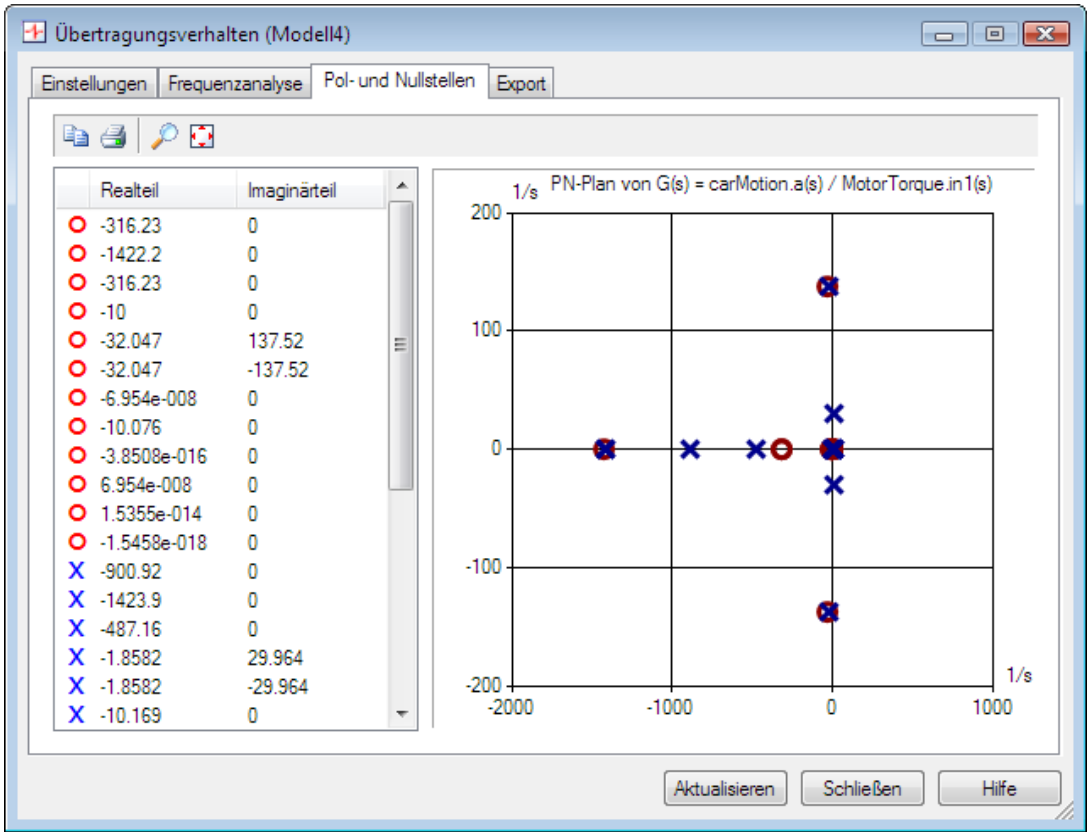



**Bild 6.52:** Pol- und Nullstellen Darstellung

### 6.5.2.4 Export

Diese Dialogseite gestattet den Export des aktiven Modells im aktuellen Arbeitspunkt in Zustandsraumdarstellung, falls diese vorliegt:

$$\dot{x} = A \cdot x + B \cdot u$$
$$y = C \cdot x + D \cdot u \qquad \textbf{(6.23)}$$

oder in Deskriptorform:

$$E \cdot \dot{x} = A \cdot x + B \cdot u$$
$$y = C \cdot x + D \cdot u \qquad \textbf{(6.24)}$$

mit:
x ... Zustandsgrößen
u ... Eingangsgrößen
y ... Ausgangsgrößen

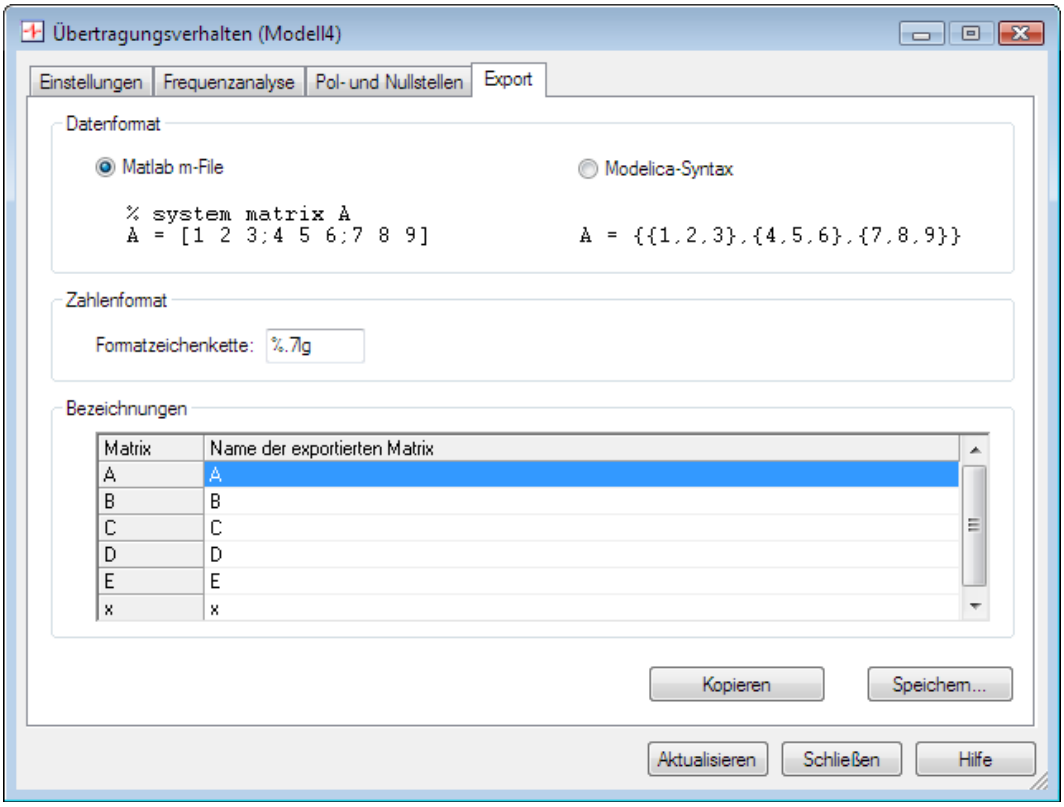



**Bild 6.53:** Export der Systemmatrizen

Zunächst müssen auf der Dialogseite ***Einstellungen*** die Ein- und Ausgangsgrößen des Systems bestimmt werden.

***Datenformat***
Bestimmen Sie hier, in welchem Format die Systemmatrizen exportiert werden sollen. Es werden folgende Formate unterstützt

- Matlab m-File

- Modelica-Syntax .

***Zahlenformat***
Bestimmen Sie hier unter **Formatzeichenkette** Genauigkeit und Format der Zahlendarstellung innerhalb der exportierten Daten. Die hier verwendete Syntax entspricht den für die printf-Funktion in der Programmiersprache C vorgesehenen Regeln.

***Bezeichnungen***
Passen Sie hier die Namen der exportierten Matrizen an. Beachten Sie die Gültigkeit der Bezeichnungen in Abhängigkeit von dem gewählten **Datenformat.**

| | |
|---|---|
| Kopieren | Betätigen Sie diese Schaltfläche, um die Systemmatrizen im gewählten Format in die Zwischenablage zu kopieren. |
| Speichern... | Betätigen Sie diese Schaltfläche, um die Systemmatrizen im gewählten Format als Datei zu speichern. Zur Auswahl von Dateiname und Zielverzeichnis wird ein Dateidialog geöffnet. |
| Aktualisieren | Betätigen Sie diese Schaltfläche, um nach Parameteränderungen im Modell die Berechnungsergebnisse zu aktualisieren. |
| Schließen | Betätigen Sie diese Schaltfläche, um das Fenster für das Übertragungsverhalten zu schließen. |
| Hilfe | Über diese Schaltfläche gelangen Sie zur Hilfe für die Arbeit mit diesem Fenster. |

# 6.6 Variantenrechnung

## 6.6.1 Anwendungsbereich

Häufig soll ein Modell mit verschiedenen Parametersätzen berechnet werden. Der Variantenassistent unterstützt Sie bei der Vereinbarung solcher Parameterstudien. Er ermöglicht die Auswahl der zu variierenden Parameter sowie deren Wertebelegung. Außerdem stellt er verschiedene Ausgabeformate bereit, in denen die Ergebnisse der Variantenrechnung zur weiteren Auswertung gespeichert werden.

Eine so beschriebene Variantenrechnung lässt sich für eine spätere Verwendung auch speichern.

## 6.6.2 Vorbereitung

Laden Sie zunächst das Modell, mit dem Sie eine Parameterstudie durchführen wollen. Um beurteilen zu können, wie lange eine Simulationsrechnung etwa dauert, sollten Sie vor der eigentlichen Variantenrechnung eine einzelne Rechnung durchführen.

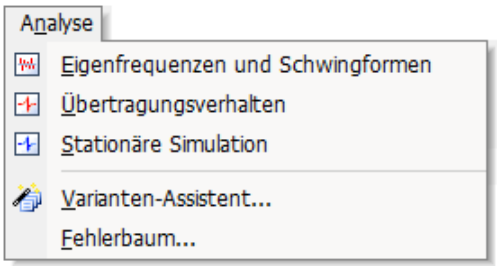



**Bild 6.54**: Menü-Auswahl

Starten Sie nun den Variantenassistenten. Den entsprechenden Befehl finden Sie im Menü ***Analyse*** oder in der Symbolleiste .
Beachten Sie, dass während der Variantenrechnung das Modell zurückgesetzt wird. Vorhandene Berechnungsergebnisse gehen dabei verloren. Speichern Sie deshalb Ihr Modell vor Aufruf des Variantenassistenten. Öffnen Sie bereits jetzt die Ergebnisfenster der Größen, deren Verlauf sie während der Variantenrechnung beobachten wollen. Schalten Sie im Ergebnisfenster ggf. die Option zum automatischen Einfrieren von Ergebnisverläufen vor jeder Neuberechnung ein (siehe **6.6.4**).

## 6.6.3 Der Variantenassistent

### 6.6.3.1 Startseite

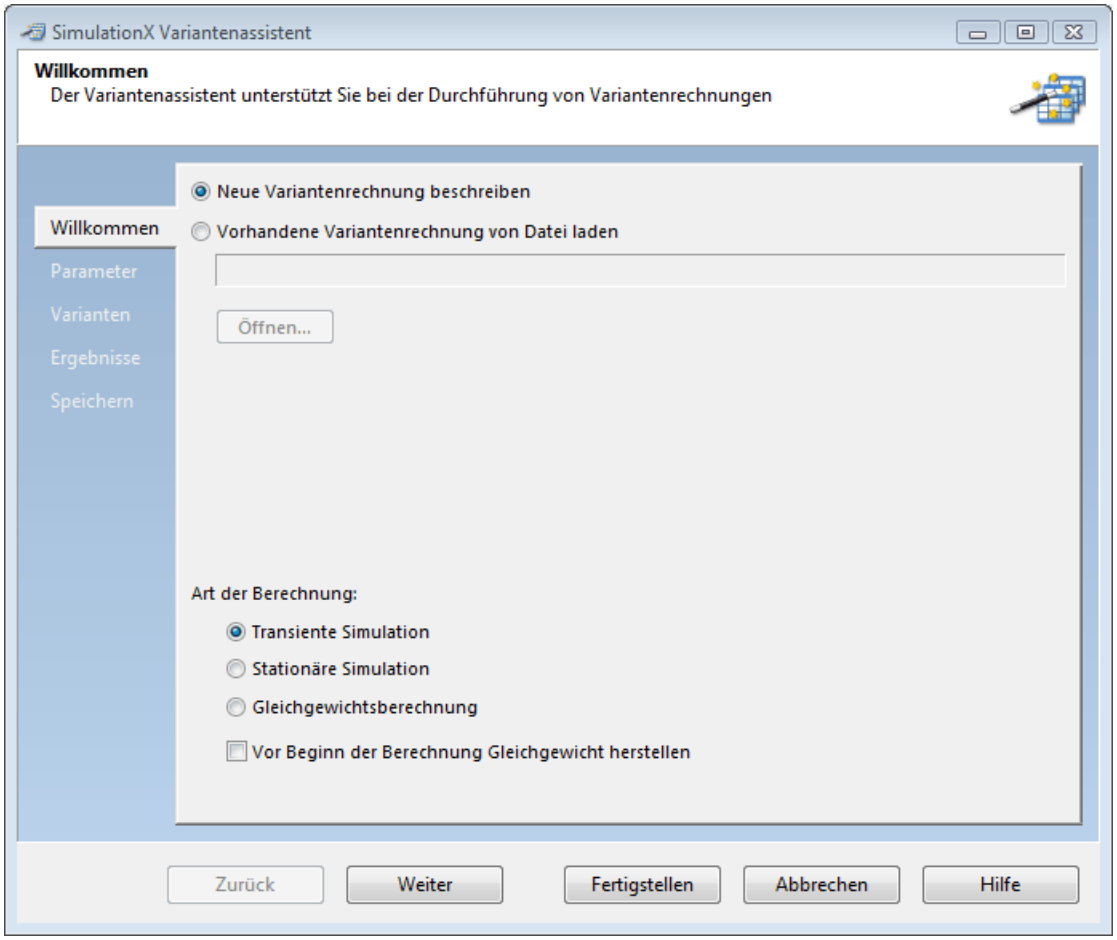



**Bild 6.55:** Variantenassistent: Dialogseite *Willkommen*

Auf der Dialogseite ***Willkommen*** wählen Sie zunächst aus, ob Sie eine neue Variantenrechnung beschreiben wollen oder eine früher bereits einmal gespeicherte Beschreibung einer Variantenrechnung wieder verwenden möchten.

Außerdem legen Sie hier die Berechnungsmethode fest. Sie können zwischen

- transienter Simulation (Standardeinstellung)
- stationärer Simulation
- oder Gleichgewichtsberechnung wählen.

### 6.6.3.2 Parameter

Die Dialogseite ***Parameter*** dient zur Auswahl der zu variierenden Größen. Dabei steht ihnen eine mit dem Modellexplorer vergleichbare Baumansicht zur Verfügung. Die zu erforderlichen Einträge können Sie durch *Drag & Drop* oder Doppelklick in die Liste der zu variierenden Parameter übernehmen.

Parametervariationen werden durch Modelica Array Ausdrücke definiert. Die Angabe von Anfangswert, Schrittweite und Endwert formulieren Sie dabei wie folgt:

```
start:step:stop
```

Aufzählung dreier Werte, die selbst auch Vektoren oder Matrizen sein können, können Sie mit dem folgenden Ausdruck vornehmen:

```
{value1, value2, value3}
```

Einträge in der Liste können sie löschen, indem sie die betreffenden Zeilen zunächst markieren und dann die Schaltfläche betätigen.

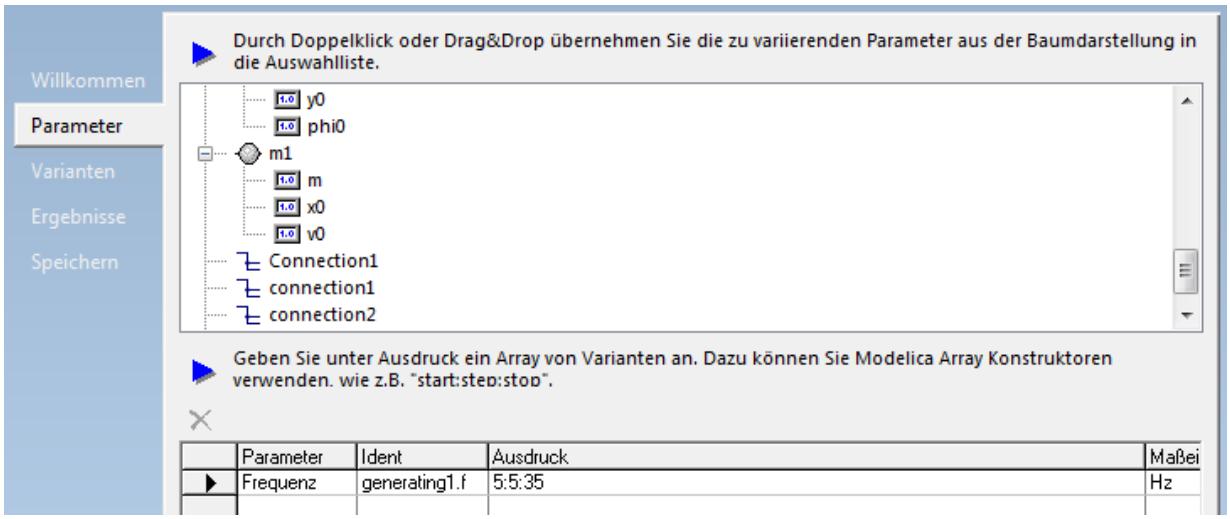



**Bild 6.56:** Dialog *Parameter*

### 6.6.3.3 Varianten

Die Dialogseite ***Varianten*** listet die Parameterbelegungen für alle durchzuführenden Berechnungen in Form einer Tabelle auf. Sie können hier Parametersätze hinzufügen oder markierte Einträge löschen.

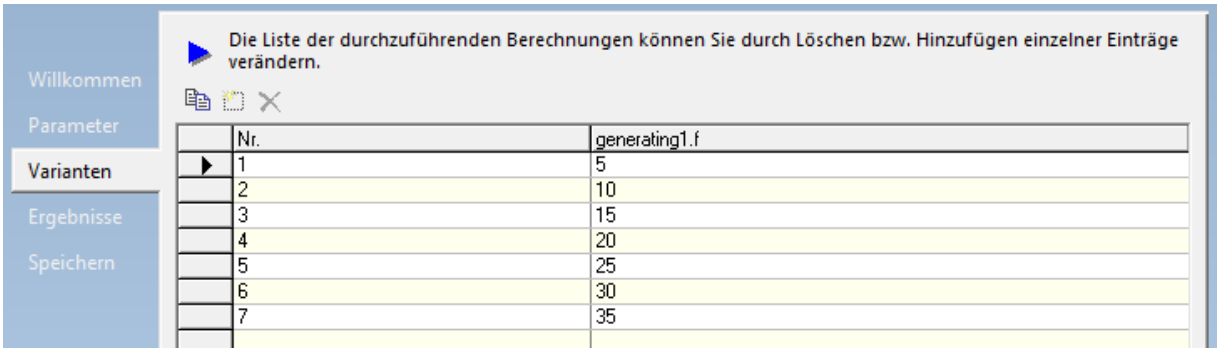



**Bild 6.57:** Dialog *Varianten*

### 6.6.3.4 Ergebnisse

Auf der Dialogseite ***Ergebnisse*** wählen Sie die Ergebnisgrößen aus, die zur Auswertung der Variantenrechnung gespeichert werden sollen.

Im oberen Teil der Ansicht (**Bild 6.58**) sehen Sie wiederum die Baumansicht des aktuellen Modells, aus der Sie mittels *Drag & Drop* oder durch Doppelklick mit der Maus die entsprechenden Einträge in die Liste unterhalb übernehmen können.

Das Protokollattribut für die ausgewählten Ergebnisgrößen wird bei Bedarf automatisch eingeschaltet.

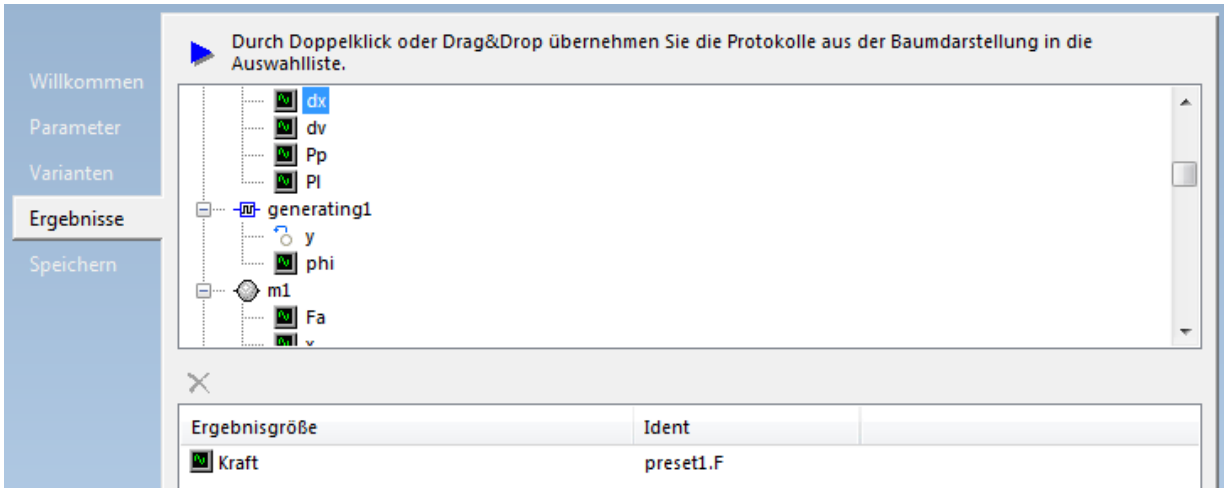



**Bild 6.58:** Dialog Ergebnisse

### 6.6.3.5 Speichern

Wird als Dateiformat eine einzelne Datei gewählt, so kann diese Datei sofort nach der Variantenrechnung mit der für das Dateiformat registrierten Anwendung geöffnet werden (z.B. MS Excel für XML-Kalkulationstabelle).

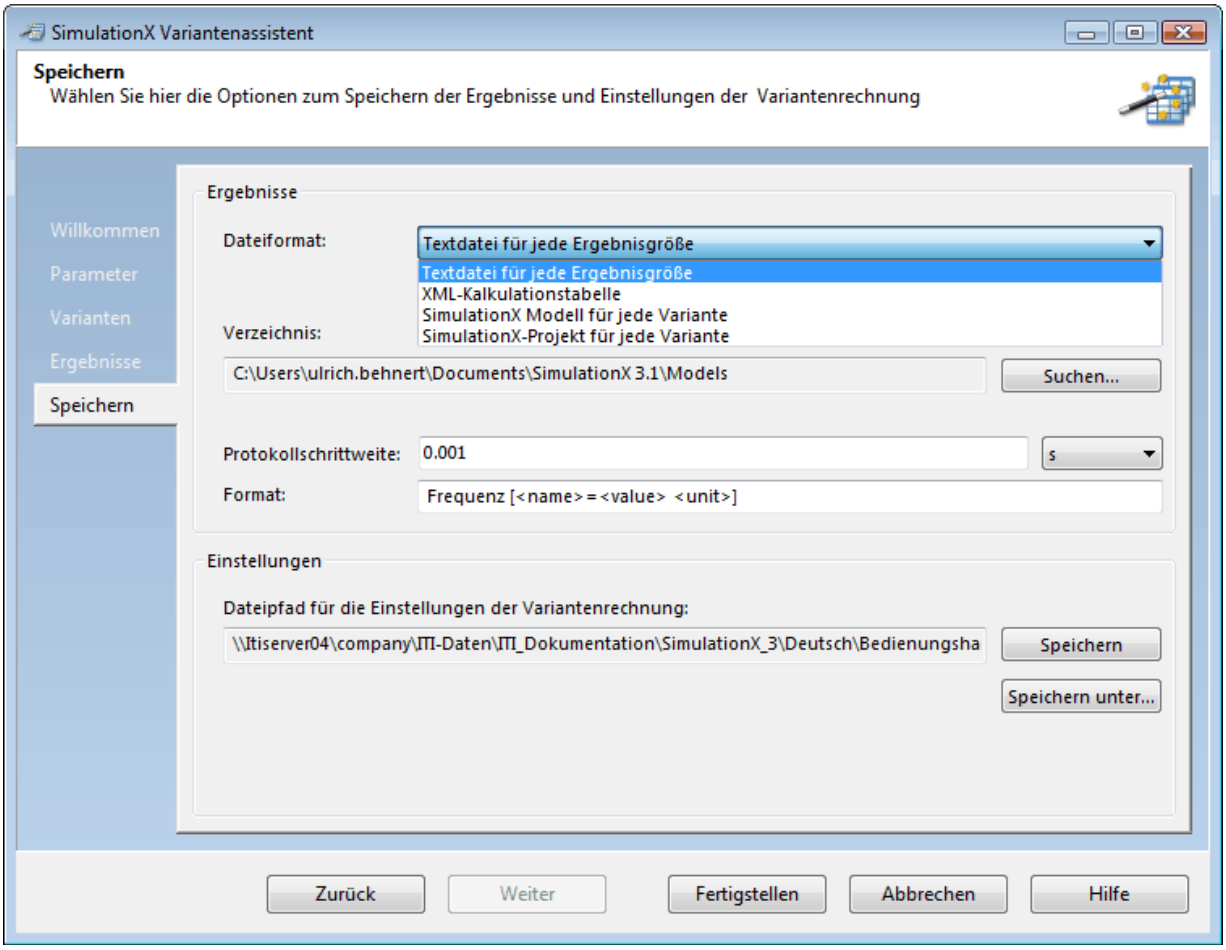



**Bild 6.59:** Dialog Speichern

Für die Speicherung der Berechnungsergebnisse stehen 4 Ausgabeformate zur Verfügung:

**<u>Textdatei für jede Ergebnisgröße (Standardeinstellung)</u>**
Für jede in der Liste vorhandene Ergebnisgröße wird eine Textdatei angelegt. Das Verzeichnis, in dem diese Dateien erzeugt werden, kann über die Schaltfläche „Suchen“ ausgewählt werden. Um die Vergleichbarkeit der einzelnen Varianten untereinander zu vereinfachen, erfolgt die Speicherung der Ergebnisse in einer äquidistanten **Protokollschrittweite**, die im entsprechenden Eingabefeld angepasst werden kann.
Im Folgenden sehen sie ein Beispiel für eine solche Datei.

Kraft F[N]

| Akt. Simulations-zeit [s] | Frequenz f=5 Hz | Frequenz f=10 Hz | Frequenz f=15 Hz |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0.01 | 0.0610679658897106 | 0.115427960730342 | 0.165126435123372 |
| 0.02 | 0.351107779866563 | 0.648306137116845 | 0.857805463509306 |

| | | | |
|---|---|---|---|
| 0.03 | 0.84043513728217 | 1.40536310612747 | 1.52968568804314 |
| 0.04 | 1.26635999100135 | 1.67116295096669 | 1.01977814907407 |
| 0.05 | 1.39485305276926 | 0.909691505499602 | -0.899841642475671 |

## XML-Kalkulationstabelle

Dieses Ausgabeformat gestattet die Speicherung aller Ergebnisse der Varianten-rechnung in einer einzigen Datei. Die Standardisierung dieses Formats gestattet die Auswertung und Weiterverarbeitung der so gespeicherten Daten durch eine wachsende Zahl anderer Programme, wie z.B. Excel. Der Name und Speicherort dieser Datei kann über die Schaltfläche angepasst werden.

Für die **Protokollschrittweite** gilt das gleiche, wie bei der Text-Datei.

Im Folgenden sehen Sie einen Ausschnitt aus einer solchen Datei:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
- <Workbook xmlns="urn:schemas-microsoft-com:office:spreadsheet"
  xmlns:ss="urn:schemas-microsoft-com:office:spreadsheet">
  - <Worksheet ss:Name="Preset1.F">
    - <Table>
      - <Row>
        - <Cell>
            <Data ss:Type="String">Akt. Simulationszeit [s]
              </Data>
          </Cell>
        - <Cell>
            <Data ss:Type="String">Frequenz f=5 Hz</Data>
          </Cell>
        - <Cell xmlns="">
            <Data ss:Type="String">Frequenz f=10 Hz</Data>
          </Cell>
        - <Cell xmlns="">
            <Data ss:Type="String">Frequenz f=15 Hz</Data>
          </Cell>
        - <Cell xmlns="">
            <Data ss:Type="String">Frequenz f=20 Hz</Data>
          </Cell>
        - <Cell xmlns="">
            <Data ss:Type="String">Frequenz f=25 Hz</Data>
          </Cell>
        - <Cell xmlns="">
            <Data ss:Type="String">Frequenz f=30 Hz</Data>
          </Cell>
        - <Cell xmlns="">
            <Data ss:Type="String">Frequenz f=35 Hz</Data>
          </Cell>
```

```
- <Cell xmlns="">
    <Data ss:Type="String">Frequenz f=25 Hz</Data>
  </Cell>
- <Cell xmlns="">
    <Data ss:Type="String">Frequenz f=30 Hz</Data>
  </Cell>
- <Cell xmlns="">
    <Data ss:Type="String">Frequenz f=35 Hz</Data>
  </Cell>
```

**SimulationX-Modell für jede Variante**

Bei diesem Format erfolgt keine vom Modell getrennte Speicherung der Ergebnisse. Vielmehr wird für jeden berechneten Parametersatz das komplette SimulationX-Modell einschließlich aller protokollierten Ergebnisse gespeichert. Dieses Format sollte zur Anwendung kommen, wenn die Auswertung der einzelnen Rechnungen mit SimulationX selbst vorgenommen werden soll.

Im **Feld Format** (siehe **Bild 6.58**) kann die bei der Ausgabe verwendete Beschriftung der berechneten Varianten angepasst werden. Die **Tabelle 6.3** gibt einen Überblick über die hier möglichen Einträge.

| Ausdruck | Bedeutung |
|---|---|
| […] | Die Klammern stehen für eine Liste der variierten Parameter in der Form p1, p2, …, pn |
| <comment> | Kommentar des Parameters pi |
| <name> | Name des Parameters pi |
| <value> | Aktueller Wert des Parameters pi |
| <unit> | Maßeinheit des Parameters pi |
| <ident> | Ident des Parameters pi |
| <iVar> | Nummer der Variante i |
| <numVar> | Anzahl Varianten n |

**Tabelle 6.3:** Formatdefinitionen

Beispiele:

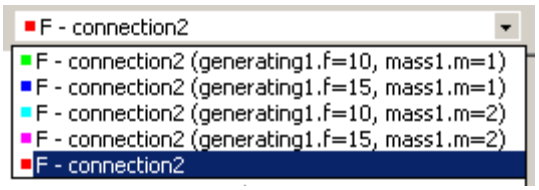
Wenn Sie folgende Parametervarianten testen wollen:

Variants

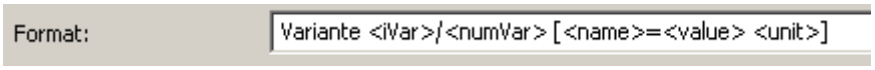
| No. | generating1.f | mass1.m |
|---|---|---|
| 1 | 10 | 1 |
| 2 | 15 | 1 |
| 3 | 10 | 2 |
| 4 | 15 | 2 |

und das folgende Format nutzen:

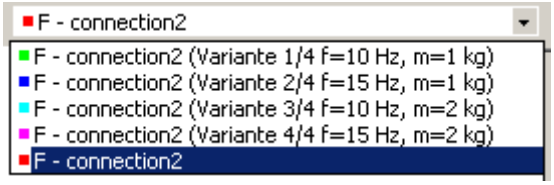
erhalten Sie folgende Ausgabe:




Ein weiteres Format:




führt zu folgender Ausgabe:




**SimulationX-Projekt für jede Variante**

Bei diesem Format erfolgt eine Speicherung der Ergebnisse in einem separaten Projekt. Es wird für jeden berechneten Parametersatz ein komplettes SimulationX-Projekt einschließlich aller protokollierten Ergebnisse gespeichert.

### 6.6.4 Beobachtung der Berechnungsergebnisse während der Variantenrechnung

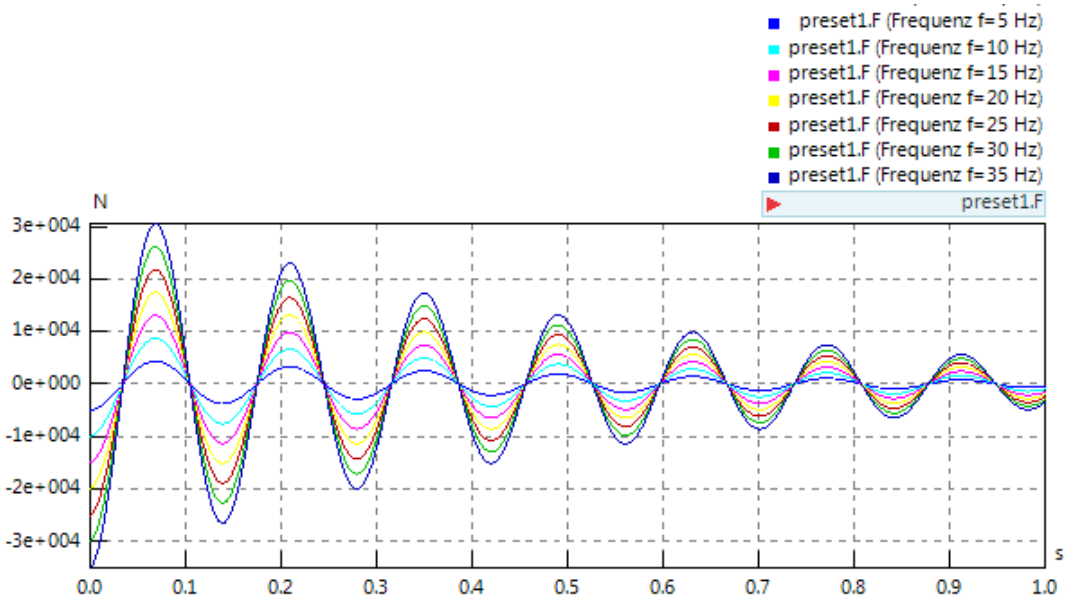



**Bild 6.60:** Beschriftetes Ergebnis der Variantenrechnung

Während der Variantenrechnung können die Berechnungsergebnisse beobachtet werden. Dazu müssen die erforderlichen Ergebnisfenster vor dem Aufruf des

Variantenassistenten geöffnet werden. Verwenden Sie die Option zum automatischen Einfrieren von Ergebnisverläufen. Gehen Sie dazu wie folgt vor:

1. Öffnen sie ein **Ergebnisfenster** für die zu beobachtende Größe
2. Öffnen sie im Ergebnisfenster den Eigenschaftsdialog und wählen sie die ***Eigenschaftsleiste***
3. Passen Sie die Einstellung für X- und Y-Achse manuell den zu erwartenden Ergebnissen an (Minimum, Maximum, Teilung)
4. Aktivieren sie die Option „***Automatisch Einfrieren beim Rücksetzen“***

Zur Beschriftung der Kurven in der Legende wird das auf der Dialogseite ***Ergebnisse*** des Variantenassistenten vorgegebene Format verwendet.

## 6.7 Ordnungsanalyse

Die Ordnungsanalyse ist eine Methode zur Frequenzanalyse von Antriebssystemen. Mit ihr können Resonanzen übersichtlich dargestellt und leicht erkannt werden. Die Ordnungsanalyse wird beispielsweise zur Untersuchung von verbrennungsmotorischen Antrieben oder Antrieben mit ungleichförmiger Übersetzung genutzt.
Mit ITI-ORD steht ein Werkzeug zur effizienten Ordnungsanalyse zur Verfügung. Die enge Anbindung an die Simulationssoftware SimulationX erleichtert das Einlesen der Quelldaten. Vielfältige Einstellmöglichkeiten sowie eine automatische Parameterwahl erlauben ein flexibles und schnelles Arbeiten. Die Ergebnisdiagramme können ausgedruckt oder über die Zwischenablage exportiert werden. Die Daten, Einstellungen und Ergebnisse lassen sich in einer Datei speichern und später wieder verwenden.

### 6.7.1 Arbeitsweise

Anwendungsbeispiel für eine Ordnungsanalyse ist der berechnete oder gemessene Hoch- oder Runterlauf eines Antriebssystems.
Eingangsdaten sind das Analysesignal $A(t)$ (z. B. Drehmoment einer Welle oder Beschleunigung an einem Punkt) und ein Basissignal $\omega(t)$ (z.B. Drehzahlsignal) über einer gemeinsamen Zeitbasis $t$.

#### 6.7.1.1 Basissignal über Frequenz (Berechnung über der Zeit)

Das Analysesignal wird in Zeitabschnitte der Länge $T$ eingeteilt und jeweils einem Wert im Basissignal zugeordnet. Diese Abschnitte werden einer diskreten (Fast-) Fouriertransformation (FFT) unterzogen.

Die Ergebnisse der Fouriertransformation liefern mittelbar die Koeffizienten einer Fourierreihe, die das Analysesignal über dem betreffenden Zeitabschnitt näherungsweise als Funktion über der Zeit beschreibt:

$$A(t) \approx a_0 + \sum_{k=1}^{N-1} a_k \cos\left(\frac{2\pi}{T} k\, t + \theta_k\right) \qquad \textbf{(6.25)}$$

Die maximal darstellbare Frequenz $f_{\max} = \frac{N/2-1}{T}$ hängt dabei von der Anzahl der Stützstellen der FFT bzw. aus deren Zeitschrittweite ab.

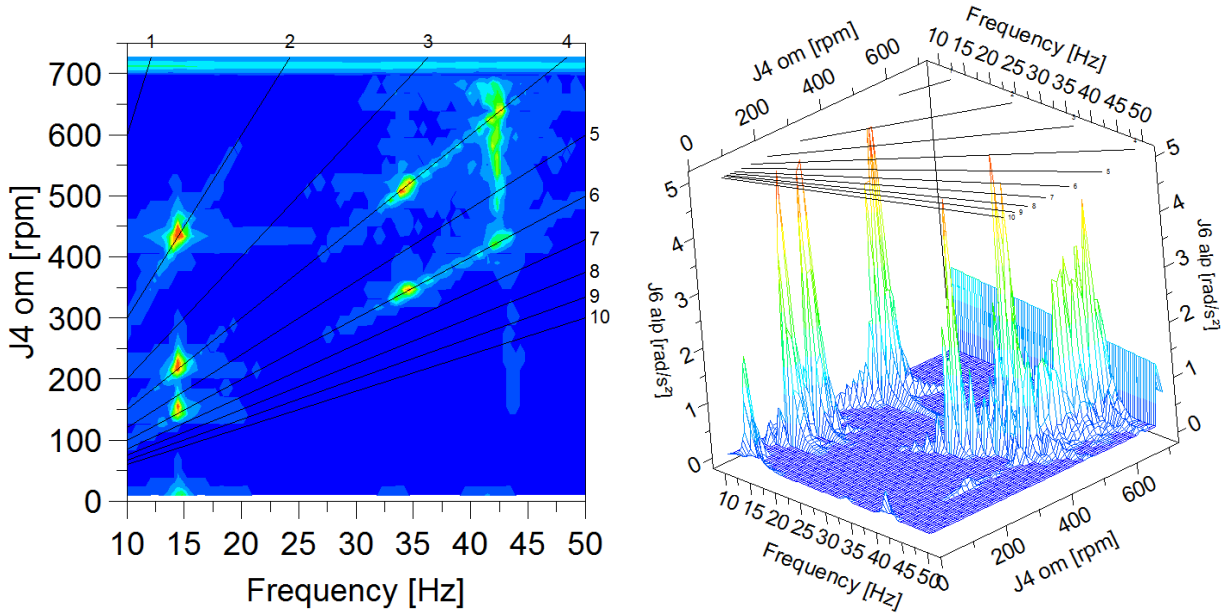



**Bild 6.61:** Sonogramm und Wasserfalldarstellung

Die Ergebnisdarstellung der Koeffizienten $a_k$ erfolgt als Sonogramm oder Wasserfalldiagramm (**Bild 6.61**), mit den Frequenzen $f_k = \frac{k}{T}$ auf der Abszisse und dem Wert des Basissignals als Ordinate.
Eigenfrequenzen erscheinen im Sonogramm als senkrechte Linien (vgl. **Bild 6.61** links bei ca. 14 Hz). Wird eine der Eigenfrequenzen angeregt, erscheint dieser Bereich farbig markiert.

Aus der Darstellungsform der Ergebnisse resultiert eine Anforderung an die zu Grunde liegenden Simulations- oder Messdaten**:** Der gleitende Mittelwert des Basissignals muss im Analyseintervall (siehe **6.7.2.2**) streng monoton sein.

### 6.7.1.2 Basissignal über Ordnung (Berechnung über dem Integral des Basissignals)

Im Gegensatz zur Berechnung über der Zeit wird das Analysesignal nicht mehr in Zeitabschnitte $T$ zerlegt, sondern die gemeinsame Zeitbasis wird zunächst durch das Integral der Basisgröße $\varphi(t) = \int_0^t \omega(\tau) d\tau$ ausgetauscht:

$$A(t) = A(\varphi(t)). \qquad \textbf{(6.26)}$$

Im Fall des erwähnten Anwendungsbeispiels (Hoch- oder Runterlauf eines Antriebsstrangs), bei dem das Basissignal $\omega$ meist der Motordrehgeschwindigkeit entspricht, ist das Integral des Basissignals der Umlaufwinkel $\varphi$ des Motors.

Damit ist es möglich das Analysesignal $A$ in Abschnitte der Breite $P$ über $\varphi$ zu zerlegen und diese Abschnitte wiederum einem Wert des Basissignals $\omega$ zuzuordnen.

Die Ergebnisse der FFT (mit äquidistanten Stützstellen des Basissignalintegrals $\Delta\varphi$) können wiederum den Koeffizienten einer Fourierreihe zugeordnet werden, die das Analysesignal in diesem Fall jedoch als Funktion über dem Integral $\varphi$ des Basissignals beschreibt:

$$A(\varphi) \approx a_0 + \sum_{k=1}^{N-1} a_k \cos\left(\frac{2\pi}{P} k\,\varphi + \theta_k\right). \qquad \textbf{(6.27)}$$

Mit Hilfe der Grundperiode $p$ (entspricht im Anwendungsbeispiel einer Umdrehung des Antriebs, d.h. $2\pi$) lassen sich die Koeffizienten der Fourierreihe den Ordnungen $n$ zuordnen:

$$n_k = \frac{k\,p}{P}. \qquad \textbf{(6.28)}$$

Die Ergebnisdarstellung des Koeffizienten $a_k$ erfolgt als Sonogramm oder Wasserfalldiagramm (**Bild 6.62**), mit den Ordnungen $n_k = \frac{k\,p}{P}$ auf der Abszisse und dem Wert des Basissignals als Ordinate.
Aus der Darstellungsform der Ergebnisse resultiert eine Anforderung an die zu Grunde liegenden Simulations- oder Messdaten**:** Der gleitende Mittelwert des Basissignals muss im Analyseintervall (siehe **6.7.2.2**) streng monoton sein. Ferner bedingt die Berechnung der Ordnungen über dem Integral des Basissignals dessen strenge Monotonie im Analyseintervall.

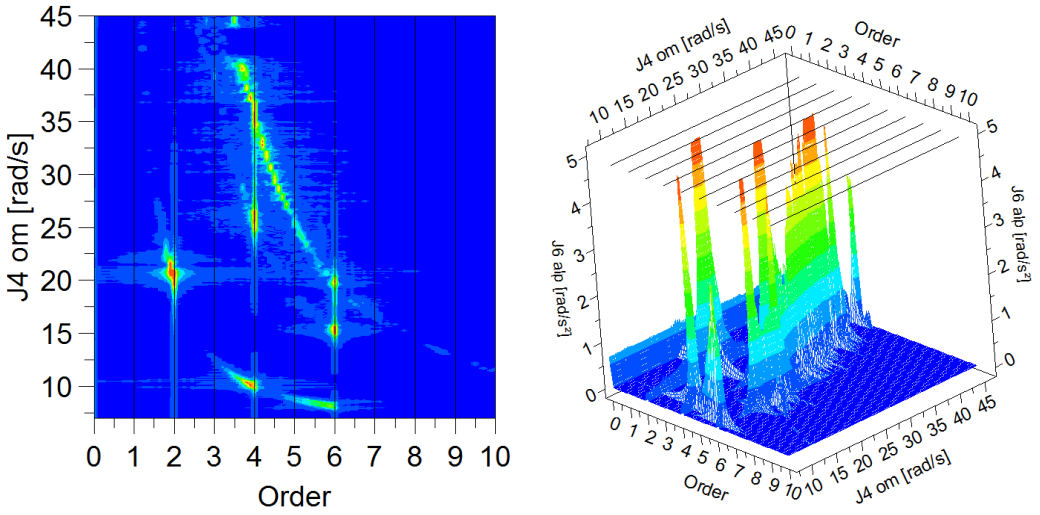



**Bild 6.62:** Sonogramm und Wasserfalldarstellung (*Basissignal über Ordnung*)

### 6.7.1.3 Ordnung über Zeit

Die Berechnung erfolgt ähnlich, wie in **Abschnitt 6.7.1.2** beschrieben, d.h. die Zeitbasis wird gegen das Integral des Basissignals ausgetauscht. Allerdings sind die Abschnitte $P$ nicht länger einem Wert des Basissignals zugeordnet sondern einem Zeitwert. Dies ist insbesondere dann sinnvoll, wenn es sich nicht um einen Hoch- oder Runterlauf des Basissignals handelt, sondern Beispielsweise um Messdaten, die bei konstanter Drehzahl des Triebstrangs ermittelt wurden.

Bei der Darstellung der Ergebnisse $a_k$ in Sonogramm und Wasserfalldarstellung wird als Abszisse die Zeit $t$ verwendet und als Ordinate die Ordnung $n$.

Bei Berechnung der Ordnungen über dem Integral des Basissignals muss dieses im Analyseintervall (siehe **6.7.2.2**) streng monoton sein.

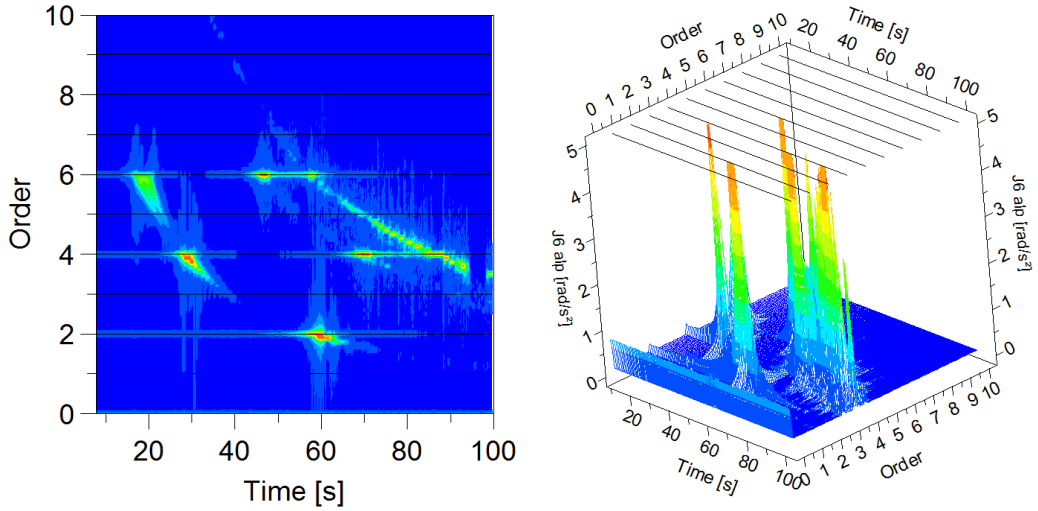



**Bild 6.63:** Sonogramm und Wasserfalldarstellung (*Ordnung über Zeit*)

## 6.7.2 Bedienung

Das Programm ITI-ORD kann über den Menüpunkt ***Analyse/ Ordnungsanalyse*** aus SimulationX heraus oder über das Windows Startmenü gestartet werden.

Das Programmfenster ist zweigeteilt. Im linken Fenster werden die Einstellungen vorgenommen, im rechten Bereich erfolgt die Diagrammanzeige (**Bild 6.64**).

Die Größe der einzelnen Fensterbereiche kann beliebig verändert werden.

Die Bedienung erfolgt in vier Schritten:

1. Import der Quelldaten
2. Auswahl der Darstellung (& Berechnungsart) und Einstellen der Analyseoptionen
3. Einstellen der Anzeigeoptionen
4. Speichern, Drucken, Export der Ergebnisse

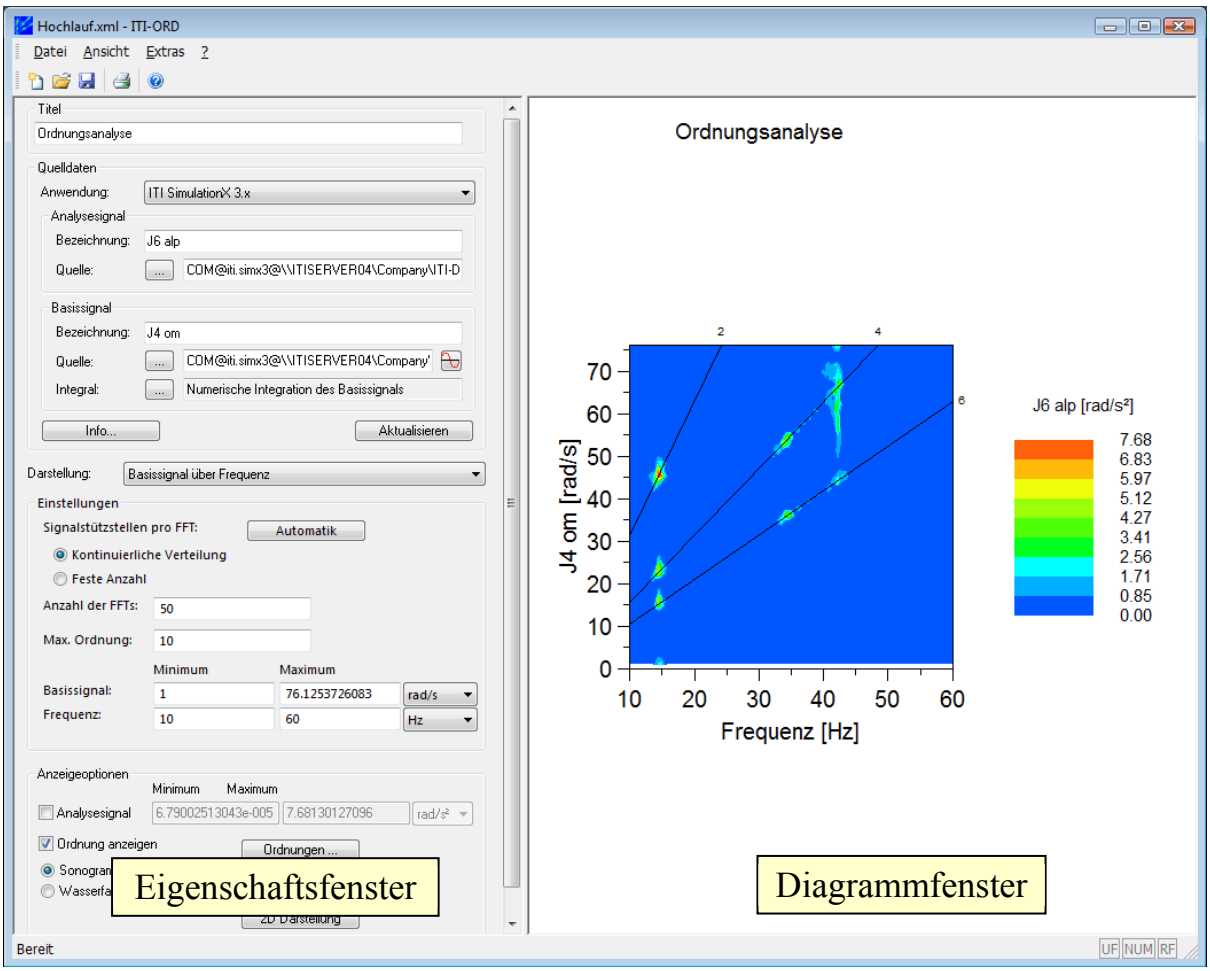



**Bild 6.64:** Programmfenster

### 6.7.2.1 Import der Quelldaten

- **Auswahl Anwendung**

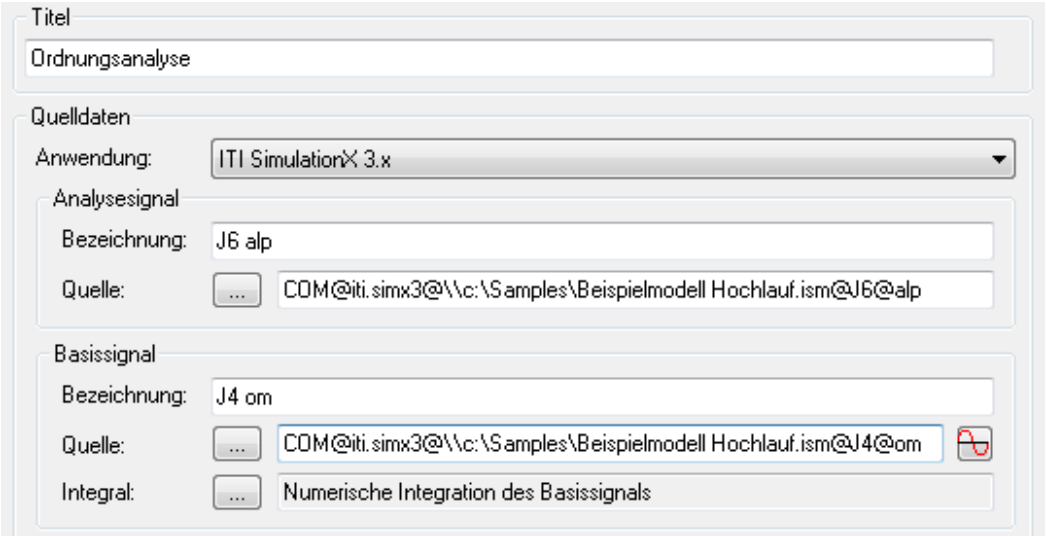



**Bild 6.65:** Import der Quelldaten

Zuerst ist die Anwendung auszuwählen, welche die Quelldaten bereitstellen soll. Im Auswahlfeld werden nur die tatsächlich installierten Anwendungen angeboten.

- **Auswahl Ergebnisgrößen**

Für eine Ordnungsanalyse müssen das Analysesignal A(t) und das Basissignal ω(t) (meist Drehzahlsignal) importiert werden. Bei den entsprechenden Ergebnisgrößen muss das Protokollattribut gesetzt sein. Das Modell muss berechnet und mindestens einmal gespeichert worden sein.

Analyse- und Basissignal sollten aus dem gleichen Modell stammen. Das Basissignal (z.B. Drehzahlsignal) sollte im Analyseintervall (näheres in **6.7.2.2**) monoton wachsend (Hochlauf) oder fallend (Runterlauf) sein. Für die Darstellung Ordnung über Zeit besteht keine Anforderung an die Monotonie des Basissignals. Für die Berechungsarten Ordnung über Zeit und Basissignal über Ordnung ist optional das Integral des Basissignals vorgebbar. Erfolgt keine Angabe, wird das Integral numerisch aus dem Basissignal berechnet.

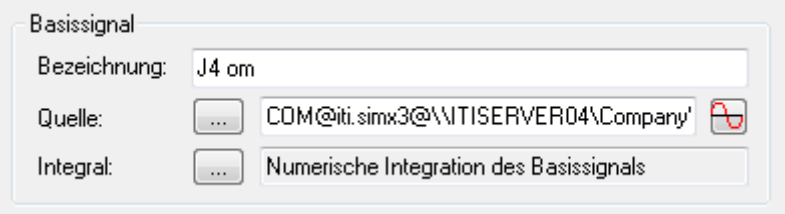



**Bild 6.66:** Numerische Integration des Basissignals

Nach betätigen der Schaltfläche ... neben dem jeweiligen Bezeichner **(Bild 6.65**) öffnet sich ein Dialog mit einer Baumansicht aller derzeit geöffneten und bereits gespeicherten Modelle der ausgewählten Anwendung. In diesem Dialog kann jeweils die entsprechende Ergebnisgröße ausgewählt werden. Die Auswahl ist mit ***OK*** zu bestätigen. Die Verknüpfung ist damit hergestellt. Im Eingabefeld unter der Schaltfläche erscheint ein Text mit folgender Syntax:

*COM@AppID@Modell@Modellobjekt@Ergebnisgröße*
Das Zeichen '@' dient als Trennzeichen zwischen den einzelnen Bezeichnern. Der Bezeichner *COM* verdeutlicht, dass die Verbindung über die COM-Schnittstelle zur Applikation hergestellt wird. Die *AppID* definiert die ausgewählte Applikation. Diese Einträge sollten nicht verändert werden.
Gefolgt werden die Einträge vom kompletten Dateinamen (mit Pfadangabe) des Modells. Die beiden letzten Einträge bezeichnen das Modellobjekt sowie die Ergebnisgröße. Diese drei letzten Einträge können auch von Hand eingegeben werden. Damit ist beispielsweise ein schnelles Verändern der Pfadangabe zum Modell möglich.

Die Eingabefelder Analyse- bzw. Basissignal werden beim Herstellen der Verknüpfung mit dem Namen des Modellobjektes und der Ergebnisgröße belegt. Dieser Text ist editierbar. Er erscheint als Achsenbezeichner im Diagramm.

Um das Integral des Basissignals auswählen zu können, ist vorher der entsprechende Punkt im oberen Teil des Dialogfeldes zu setzen (siehe **Bild 6.67**).

Der eigentliche Import der Quelldaten erfolgt, wenn für Analyse- und Basissignal Verknüpfungen bestehen.
Ein nochmaliger Datenimport (z. B. wenn das Modell erneut simuliert wurde) erfolgt bei Betätigung der Schaltfläche ***Aktualisieren***. Falls die Simulationsanwendung zu diesem Zeitpunkt nicht geöffnet ist, wird sie gestartet und das entsprechende Modell geladen. Ist die Anwendung nicht installiert oder wird das Modell, das Modellobjekt oder die Ergebnisgröße nicht gefunden, so erfolgt eine entsprechende Fehlermeldung.

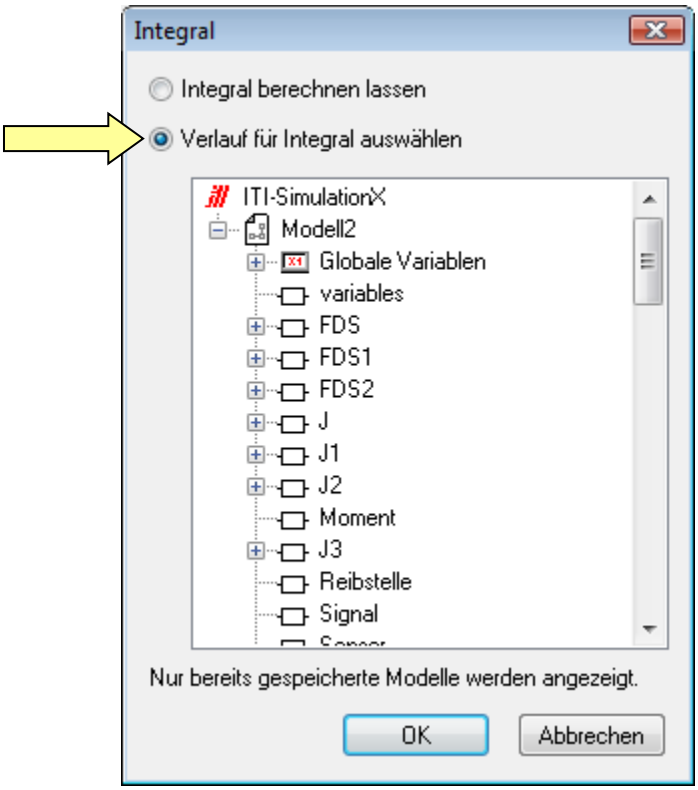



**Bild 6.67:** Auswahl des Integrals des Basissignals

Die Verknüpfungen können jederzeit neu erstellt werden. Das über die Schaltfläche ***Info...*** aufrufbare Dialogfenster zeigt Informationen zu den importierten Verläufen.

- **Periode des Basissignals**

Die Auswahl des Basissignals beinhaltet zusätzlich die Definition der Periodenlänge (**Bild 6.68**).
Wenn die ausgewählte Ergebnisgröße eine Drehzahl ist (z. B. die Drehzahl einer Drehträgheit), dann beträgt die Periode i.A. eine Umdrehung ($2\pi$ [rad]). Falls das Basissignal in einer anderen physikalischen Größe vorliegt, ist die Periodenlänge in der SI Einheit anzugeben. Liegt das Basissignal z.B. in [m/s] vor, so muss die Periodenlänge in Meter angegeben werden.

**Bild 6.68:** Definition Periodenlänge

Die Periodenlänge ist für die korrekte Darstellung der Ordnungslinien wichtig. Außerdem wird die Periodenlänge zur Erzeugung der äquidistanten Verläufe verwendet (siehe **6.7.2.1**).

Die Periodenlänge ist auch nachträglich änderbar.

- **Erzeugung äquidistanter Verläufe**

Für die Berechnung im Zeitbereich werden, äquidistant abgetastete Verläufe über der Zeit erzeugt, falls die Quelldaten es nicht bereits sind. Die Schrittweite (`dt`) wird aus der maximalen Schrittweite im Basissignal ($\Delta Base_{max}$) berechnet:

$$f_{\max} = \frac{\Delta Base_{\max}}{2}$$
$$dt = \frac{1}{2 f_{\max}} \qquad \textbf{(6.29)}$$

Die Werte von Basis- und Analysesignal an den Stützstellen werden durch lineare Interpolation ermittelt.

### 6.7.2.2 Einstellen der Analyseoptionen

Die Analyseoptionen werden im mittleren Bereich des Einstellungsfensters festgelegt (**Bild 6.69**).

- **Analyseoptionen bei Berechnung über der Zeit**

Der Parameter *Anzahl der FFTs* legt fest, wie viele Signalabschnitte gebildet werden, also wie viele FFTs zu berechnen sind. Es stehen zwei Verfahren zur Auswahl, wie diese Abschnitte gebildet werden.

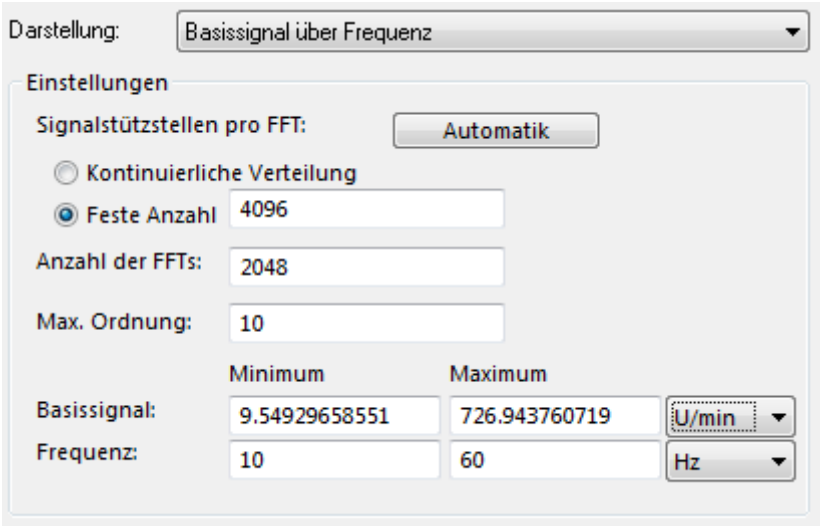



**Bild 6.69:** Einstellungen der Analyseoptionen

- Kontinuierliche Verteilung

Die Signalabschnitte werden nach folgendem Algorithmus gebildet:

- Suche nach dem ersten Auftreten der aktuellen Stützstelle $Base_n$ im Basissignalverlauf (bei Hochlauf von unten nach oben, Runterlauf umgekehrt).
- Der Abschnitt beginnt bei der zum Wert des Basissignals $\frac{Base_{n-1} + Base_n}{2}$ gehörenden Signalstützstelle und endet bei der zum Wert $\frac{Base_n + Base_{n+1}}{2}$ gehörenden Signalstützstelle.

Die Signalabschnitte können also durchaus unterschiedliche Längen besitzen. Der Vorteil dieser Variante ist, dass das Signal komplett verwendet wird, es also keine Bereiche geben kann, die nicht analysiert werden, und auch keine Überschneidungen auftreten.

Es ist zu beachten, dass bei dieser Methode kein direkter Einfluss auf die Länge der einzelnen Signalabschnitte besteht. Bei längeren Signalsequenzen und wenigen Stützstellen im Basissignal entstehen relativ lange Abschnitte. Innerhalb eines solchen langen Abschnittes kann sich die Frequenz stark ändern, wodurch die Ergebnisse der FFT verfälscht werden. Bei kurzen Signalsequenzen und vielen Stützstellen im Basissignal entstehen kurze Signalabschnitte. In Folge dessen ist eine FFT nur mit niedriger Frequenzauflösung möglich.

- Feste Anzahl

Bei diesem Verfahren kann der Anwender festlegen, aus wie vielen Signalstützstellen $N$ ein Signalabschnitt bestehen soll. Ein Signalabschnitt besteht dann immer aus den $\frac{N}{2}$ Werten jeweils vor und nach der Stützstelle im Basissignal.

Hier ist zu beachten, dass insbesondere bei langen Signalsequenzen, wenigen Stützstellen im Basissignal und geringer Anzahl von FFTs, Abschnitte des Signals nicht mit in die FFT einbezogen werden. Dadurch besteht die Gefahr, dass existierende Überhöhungen im Analysesignal nicht sichtbar sind. Analog kann es zu einem Überschneiden von Signalabschnitten kommen. Dadurch werden diese Bereiche mehrfach analysiert und das Ergebnis erscheint verwischt.

Es empfiehlt sich deshalb, die Einstellungen zu variieren und die Ergebnisse dabei kritisch zu analysieren.

Die Angaben zu *Minimum* und *Maximum* legen fest, welcher Abschnitt im Signalverlauf analysiert werden soll, sowie welcher Frequenzbereich interessiert.

Die Schaltfläche ***Automatik*** stellt sinnvolle Parameter für alle Optionen ein. Das sind:

- Kontinuierliche Verteilung der Signalstützstellen
- 50 FFTs
- Minimum und Maximum des Basissignals
- Minimale Frequenz = 0, maximale Frequenz $\frac{1}{2dt}$ ($dt$ ist Schrittweite der Zeitbasis)

Haben die Einstellungen konsistente Werte, kann die Schaltfläche ***Berechnen*** betätigt und so die Ordnungsanalyse durchgeführt sowie das Diagramm aktualisiert werden.

- **Analyseoptionen bei Berechnung über dem Winkel**

Für die Berechnungs- und Darstellungsarten *Basissignal über Ordnung* und *Ordnung über Zeit* können die folgenden Parameter für die Berechnung vorgegeben werden (vgl. **Bild 6.63**):

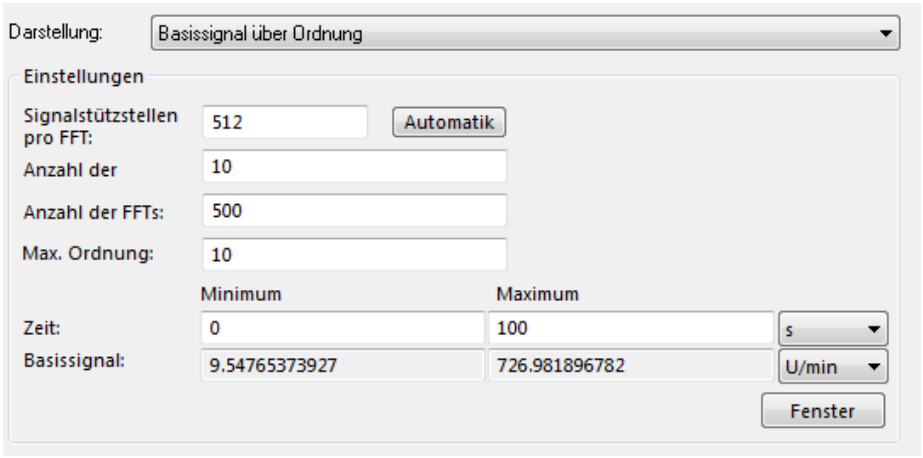



**Bild 6.70:** Dialogteil Einstellungen der Analyseoptionen bei Berechnung über dem Winkel

| Parameter | Bemerkung |
|---|---|
| Signalstützstellen pro FFT | Punktezahl der diskreten Fouriertransformation über einem Fenster (muss eine gerade Zahl aber nicht unbedingt eine Potenz von 2 sein) |
| Anzahl der Perioden | Die Breite eines Fensters P resultiert aus dem Produkt der Anzahl der Perioden und der Periodenlänge. |
| Anzahl der FFTs | Dieser Parameter gibt an, wie viele FFT-Fenster der Breite P über dem gesamten Analyseintervall verteilt werden. |
| Max. Ordnung | die maximal dargestellte Ordnung |
| Zeit - Minimum | Beginn des zu analysierenden Intervalls |
| Zeit -Maximum | Ende des zu analysierenden Intervalls |

**Tabelle 6.4:** Parameter bei Berechnung über dem Integral des Basissignals

- Signalstützstellen pro FFT

Zur Berechnung einer diskreten Fouriertransformation über einem Intervall *P* nach der Gleichung

$$X[j] = \sum_{k=1}^{N-1} x[k] e^{-i\frac{2\pi}{N}k\,j} \qquad \textbf{(6.30)}$$

werden *N* Signalstützstellen *x*[*k*] des Analysesignals verwendet.

Es ist zu beachten, dass die Anzahl der Stützstellen *N* eine gerade Zahl sein muss und die maximal darstellbare Ordnung beeinflusst:

$$n_{\max} = \frac{N/2 - 1}{M} \qquad \textbf{(6.31)}$$

Dabei ist *M* die Anzahl der Perioden (siehe **Bild 6.71**).

- Anzahl der Perioden

Dieser Parameter legt die Länge *P* eines Fensters für eine FFT als geradzahliges vielfaches der Grundperiode fest.

Die Anzahl der Perioden gibt die maximale Auflösung der Ordnungen vor:

$$\Delta n = \frac{1}{M}. \qquad \textbf{(6.32)}$$

Aus diesem Grund muss *M* auch geradzahlig sein, da sonst vielfache der halben Ordnung nicht Teil der Lösungsmenge sind.

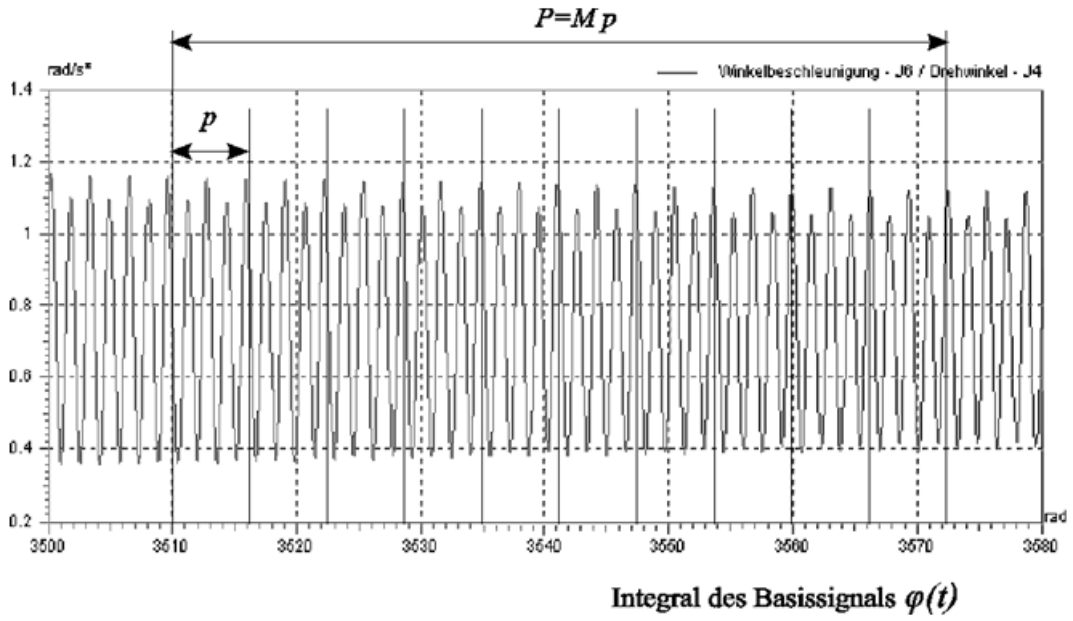



**Bild 6.71:** Anzahl der Perioden *M*

- Anzahl der FFTs

Dieser Parameter gibt an, wie viele FFT-Fenster der Breite *P* auf dem Analyseintervall (*Minimum* bis *Maximum*) verteilt werden. Er beeinflusst daher auch wie stark sich die Fenster gegebenenfalls gegenseitig überdecken. Im Extremfall können diese enger nebeneinander liegen als die Stützstellen des Analysesignals. In diesem Fall erscheint zu Beginn der Berechnung eine Fehlermeldung. Die Stützstellen sind dann deutlich zu reduzieren.

- Maximale Ordnung

Der Parameter *Max. Ordnung* ( $n_{\max}$ ) legt die maximal dargestellte Ordnung in den verschiedenen Diagrammen fest.

Die Anzahl der Stützstellen $N$ (siehe **6.7.2.2** ) muss eine gerade Zahl sein, sie beeinflusst die maximal darstellbare Ordnung:

$$n_{\max} = \frac{N/2 - 1}{M}. \qquad \textbf{(6.33)}$$

Der eingegebene Wert kann daher durch die Berechnung automatisch reduziert werden. Gegebenenfalls ist die *Anzahl der Stützstellen* pro FFT $N$ zu erhöhen um die gewünschte maximale Ordnung darstellen zu können.

- Festlegung des Analyseintervalls

Über die Parameter *Minimum* und *Maximum* erfolgt die Angabe der Zeitgrenzen des Analyseintervalls. Diese Grenzen sind der Beginn des ersten bzw. das Ende des letzten FFT-Fensters (siehe **Bild 6.74**).

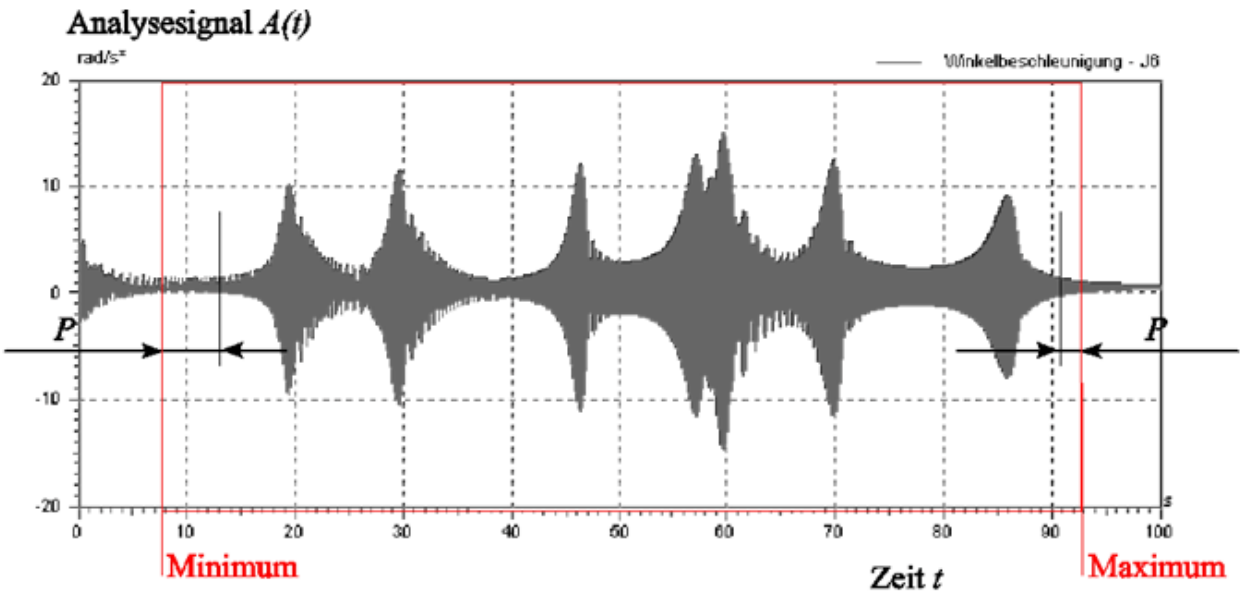



**Bild 6.72:** Festlegung des Analyseintervalls

Eine Änderung des Intervalls empfiehlt sich insbesondere dann, wenn ihre Mess- oder Simulationsdaten (z.B. durch Einschwingvorgänge) gegen Restriktionen verstoßen, die zur Berechnung (oder Darstellung) der Ergebnisse erforderlich sind. Dies betrifft insbesondere die Monotonie der Signale (siehe auch **Abschnitt 6.7.1**).

Bei ausgewählter Darstellung Basissignal über Ordnung werden die zugehörigen Grenzen des Basissignals dargestellt (siehe **Bild 6.63**). Die direkte Vorgabe dieser Grenzen ist nicht erlaubt! Die in den Diagrammen verwendete Einheit des Basissignals ist jedoch über die anschließende Auswahlbox änderbar.

### 6.7.2.3 Einstellung der Anzeigeoptionen

Die am häufigsten benötigten Diagrammeinstellungen werden im unteren Bereich des Einstellungsfensters vorgenommen.

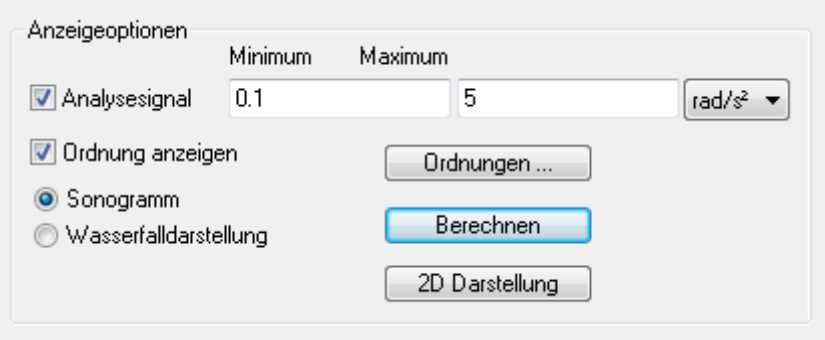



**Bild 6.73:** Anzeigeoptionen

Nach dem Markieren des Kontrollkästchens ***Analysesignal*** lassen sich Minimum und Maximum der darzustellenden FFT-Werte festlegen. Das ist insbesondere dann sinnvoll, wenn es große Unterschiede in den Dimensionen der FFT-Werte gibt.

Des Weiteren erfolgt hier die Umschaltung zwischen Sonogramm und Wasserfalldarstellung.

Die Perspektive der Wasserfalldarstellung lässt sich direkt mit der Maus durch Anklicken und gleichzeitiges Bewegen mit der Maus auf die Darstellung im Diagrammfenster ändern. Ein Zurücksetzen der Perspektive erfolgt über des Kontextmenü des Diagrammfensters – Menüpunkt ***Rücksetzen***. Ebenfalls über dieses Kontextmenü kann mittels ***Eigenschaften*** der Dialog zur Einstellung der Darstellungsoptionen geöffnet werden. Hier lassen sich eine Vielzahl von Diagrammeigenschaften einstellen.

- **Darstellung Ordnungslinien**

Über die Schaltfläche ***Ordnungen*** gelangt man in den Dialog ***Auswahl Ordnungslinien*** (**Bild 6.74**). Jede Ordnungslinie kann einzeln ein- bzw. ausgeblendet und mit einem eigenen Kommentar beschriftet werden. Halbe Ordnungen sind gleichfalls darstellbar.

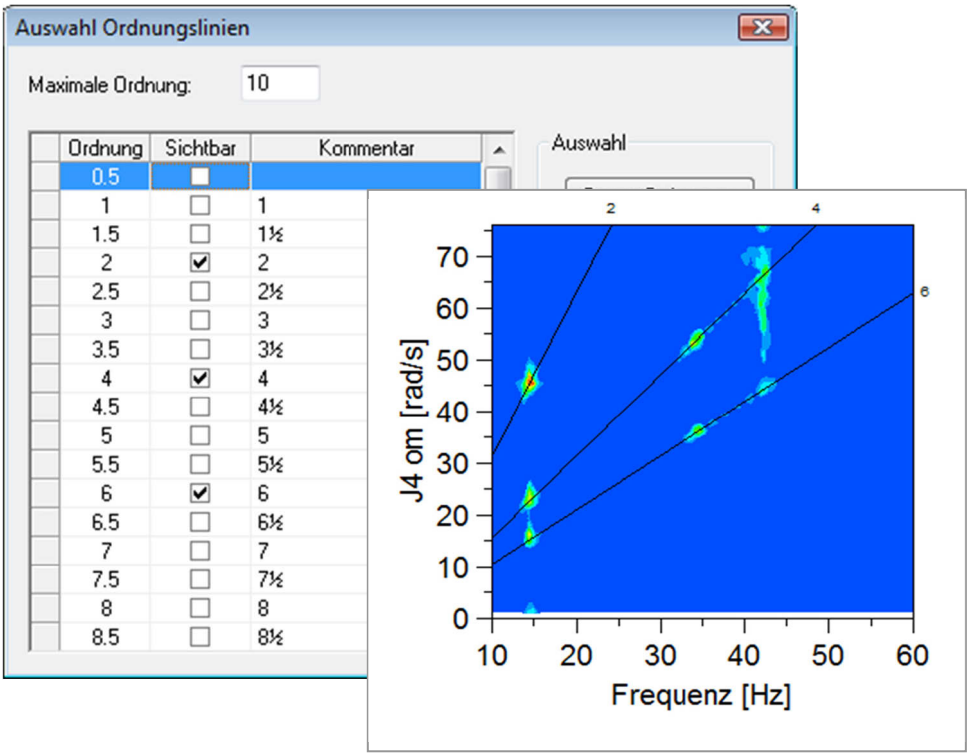



**Bild 6.74:** Auswahl der Ordnungslinien

- **2D Darstellung**

Der Aufruf des Diagramms erfolgt durch Betätigung der Schaltfläche ***2D Darstellung***.

Die 2D-Ansicht stellt die Amplituden der gemäß **Bild 6.74** ausgewählten Ordnungen je nach gewählter Darstellungsart entweder über dem Basissignal

(**Bild 6.75 -** *Basissignal über Frequenz)* oder der Ordnung (*Basissignal über Ordnung*) dar.

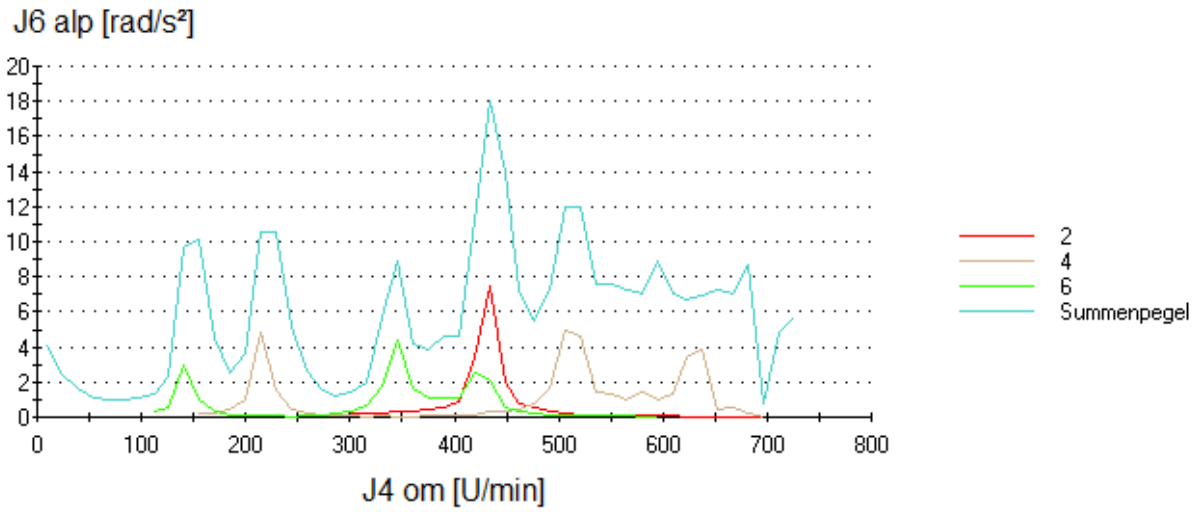



**Bild 6.75:** 2D Darstellung (inkl. Summenpegel)

Der Summenpegel ist das Maximum des Analysesignals im Zeitbereich bei der betrachteten Drehzahl. Die Darstellung des Summenpegels kann bei der Auswahl der Ordnungslinien ein- bzw. ausgeschaltet werden.

### 6.7.2.4 Speichern, Drucken, Export der Ergebnisse

Über den Menüpunkt ***Speichern*** lassen sich alle Daten zu jedem Zeitpunkt in einer Datei speichern. Auch die Analysedaten werden, sofern sie bereits importiert sind, mit gespeichert. Somit ist eine Benutzung des Programms auch unabhängig von der Simulationssoftware möglich. Die Daten werden im XML-Format gespeichert. Es ist daher möglich sie von Hand zu ändern oder in andere Programme einzubinden.

Für den Ausdruck stehen die Standardbefehle von Windows zur Verfügung.

Die Ergebnisgrafik lässt sich über das Kontextmenü des Diagrammfensters in die Zwischenablage kopieren. Der Export kann als Pixelgrafik (Bitmap) oder Vektorgrafik (Enhanced Metafile) erfolgen.

- **CSV-Export der 2D-Darstellung**

Im Fenster der 2D Darstellung lassen sich die Verläufe über das Menü ***Datei/Speichern…***im ASCII-Format exportieren. Die Werte sind darin spaltenweise abgelegt und jeweils durch Kommata getrennt. Dies erlaubt einen einfachen Import in Ergebnisfenster von SimulationX. Dort kann z.B. der Vergleich der Ergebnisse unterschiedlicher Quelldaten erfolgen.

# 6.8 Leistungsbilanz

## 6.8.1 Allgemeine Funktionsweise

Die Leistungsbilanz kann zur detaillierten Untersuchung der Leistungs- und Energieflüsse in einem SimulationX-Modell verwendet werden. Oft ist es nicht ausreichend den in ein System oder aus einem System heraus fließenden Leistungsfluss zu messen, weil dabei keine Aussagen über den Verbleib der zu- oder abgeführten Energie getroffen werden können. Es ist dabei unbekannt, inwieweit z.B. zugeführte Energie im System gespeichert oder als Verlustleistung umgesetzt wird. Diese Fragestellungen können mit Hilfe der Leistungsbilanz in einem SimulationX-Modell gezielt untersucht werden.

In SimulationX steht die Leistungsbilanz derzeitig für die Bibliotheken *rotatorische Mechanik, translatorische Mechanik, Elektrik* und *Hydraulik* zur Verfügung. Alle Elemente dieser Bibliotheken berechnen die in ihnen umgesetzten charakteristischen Leistungen. Diese ergeben sich jeweils als Produkt aus Flussgröße (z.B. mechanische Kraft) und Potentialgröße (z.B. mechanische Geschwindigkeit).

Dabei kann Leistung durch Elemente dem System zugeführt werden (z.B. mechanische Momentenquelle), im System gespeichert werden (z.B. mechanische Trägheit oder Feder) oder als Verlustleistung umgesetzt werden (z.B. mechanischer Dämpfer).

Es werden deshalb vier Leistungsarten im System definiert:

- zugeführte Leistung
- gespeicherte Leistung Typ A
- gespeicherte Leistung Typ B
- Verlustleistung

Für die beiden Arten der gespeicherten Leistung existieren keine einheitlichen Bezeichnungen in verschiedenen physikalischen Domänen (vgl. Mechanik vs. Elektrik). Als Oberbegriffe werden deshalb die allgemeinen Bezeichnungen gespeicherte Leistung Typ A und Typ B verwendet.
Einige Elemente, wie z.B. ein mechanisches Feder-Dämpfer-Element, berechnen aufgrund ihrer Funktionsweise mehrere Leistungsanteile.

Die Leistungsbilanz in SimulationX sammelt alle Leistungsanteile der Elemente des jeweiligen Modells ein und ordnet sie den 4 Leistungsarten zu. Das Modell erhält dabei 4 neue Ergebnisvariablen, wobei die Leistungsanteile der jeweiligen Art summiert werden.

Die folgende Tabelle stellt eine Übersicht dieser 4 Leistungsarten sowie Beispielelemente, wo diese auftreten, für die verschiedenen Bibliotheken dar.

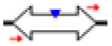
| Bibliothek | **zugeführte Leistung** | **Gespeicherte Leistung Typ A** | **Gespeicherte Leistung Typ B** | **Verlust-leistung** |
|---|---|---|---|---|
| translatorische Mechanik | Äußere Leistung<br>z.B. Äußere Kraft  | Änderung der kinetischen Energie<br>z.B. Masse | Änderung der potentiellen Energie<br>z.B. Feder | Verlustleistung<br>z.B. Dämpfer |
| rotatorische Mechanik | Äußere Leistung<br>z.B. Äußeres Moment | Änderung der kinetischen Energie<br>z.B. Trägheit | Änderung der potentiellen Energie<br>z.B. Feder | Verlustleistung<br>z.B. Dämpfer |
| Elektrik | Äußere Leistung<br>z.B. Spannungs-quelle | Änderung der elektrischen Feldenergie<br>z.B. Kondensator | Änderung der magnetischen Feldenergie<br>z.B. Spule | Verlustleistung<br>z.B. Widerstand |
| Hydraulik | Äußere Leistung<br>z.B. Druck-quelle | Änderung der kinetischen Energie<br>z.B. Leitung (ohne Verluste) | Änderung der potentiellen Energie<br>z.B. Volumen | Verlustleistung<br>z.B. Drossel |

**Tabelle 6.5:** Leistungsarten in verschiedenen Bibliotheken und Beispielelemente

Die **Tabelle 6.6** zeigt diese 4 Leistungsvariablen und definiert, welche Leistungsanteile den einzelnen Bibliotheken zugeordnet werden.

Wahlweise können zusätzlich aus diesen 4 Leistungsbilanzvariablen durch Integration die entsprechenden Energien gebildet werden.

Außerdem kann die Summe aller Leistungen PSum gebildet werden.
PSum berechnet sich aus den vier Leistungsbilanzvariablen:

$$PSum = PSrc - (PStoreA + PStoreB + PLoss) \qquad \textbf{(6.34)}$$

Diese Variable dient als Kontrollgröße ob alle Leistungen in der Bilanz berücksichtigt wurden. Sind alle Leistungen des Modells in den Bilanzgrößen korrekt summiert, gilt: PSum = 0.

| | Zugeführte Leistung<br><br>**PSrc** | Gespeicherte Leistung (Typ A)<br><br>**PStoreA** | Gespeicherte Leistung (Typ B)<br><br>**PStoreB** | Verlust-leistung<br><br>**PLoss** |
|---|---|---|---|---|
| translatorische Mechanik | Pe | Pk | Pp | Pl |
| rotatorische Mechanik | Pe | Pk | Pp | Pl |
| Elektrik | Pe | Pelek | Pmag | Pl |
| Hydraulik | Pe | Pk | Pp | Pdiss, Pfr |

**Tabelle 6.6:** Bilanzvariablen der Leistungsarten und zugeordnete Leistungen

## 6.8.2 Erstellung einer Leistungsbilanz in SimulationX

### 6.8.2.1 Allgemeine Vorgehensweise

Zum Einfügen einer Leistungsbilanz ist zuerst die Strukturansicht des zu analysierenden Modells anzuwählen.
Jetzt erfolgt die Auswahl der Option ***Leistungsbilanz*** unter Menü ***Einfügen/ Leistungsbilanz***.

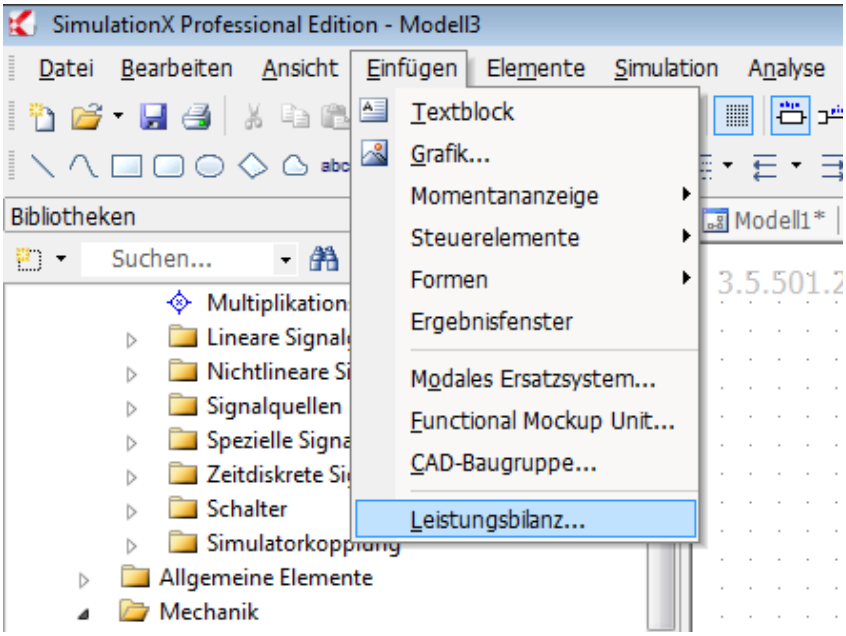



**Bild 6.76:** Einfügen der Leistungsbilanz

Im folgenden Auswahlfenster können die einzufügenden Leistungs- und Energiebilanzvariablen ausgewählt werden. Die Auswahl ist durch Klick auf den Button ***OK*** zu bestätigen.
Es werden jetzt die entsprechenden Ergebnisvariablen dem Modell hinzugefügt. Diese sind nun im Modellexplorer sichtbar und ihre Protokollierung kann hier aktiviert und deaktiviert werden.

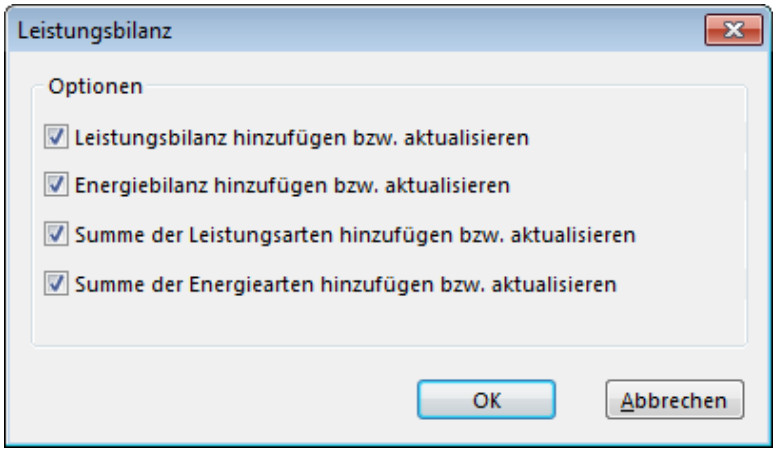



**Bild 6.77:** Auswahl der einzufügenden Leistungs- und Energievariablen

Öffnet man das Modell imTypdesigner (Klick im Modellexplorer auf *Typen* – Doppelklick auf *„Modelname“*), finden sie dort die Variablen auf der Seite *Komponenten*. In der Zeile *Deklarationsgleichung* ist die jeweilige Bilanzgleichung eingetragen.

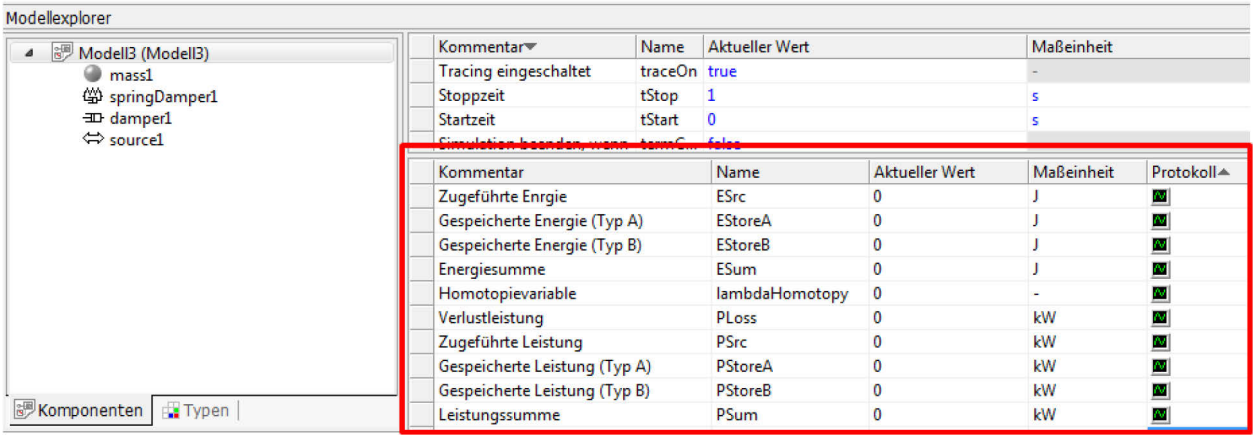



**Bild 6.78:** Modellexplorer – Leistungs- und Energiebilanzvariablen

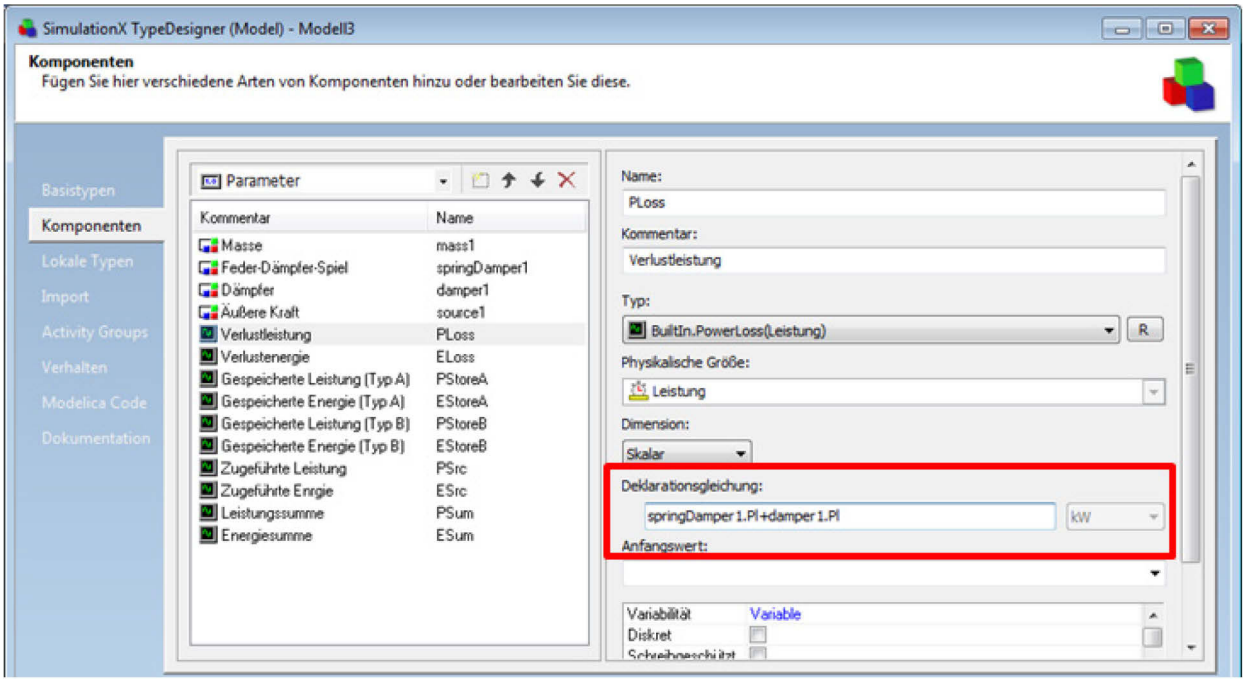



**Bild 6.79:** Typdesigner – Bilanzgleichungen

Bei Änderungen an der Modellstruktur oder Elementnamen im Modell muss die Leistungsbilanz aktualisiert werden. Klicken sie dafür erneut auf ***Leistungsbilanz*** (**Bild 6.79**)!

### 6.8.2.2 Vorgehensweise bei Modellen mit Compounds

Eine Leistungsbilanz kann ebenfalls für Modelle mit Substrukturen (Compounds) eingefügt werden. Dabei müssen zuerst in den Compounds selbst Leistungsbilanzen der enthaltenen Elemente eingefügt werden, bevor auf der obersten Modellebene die Bilanz des Gesamtmodelles erfolgen kann.

Zuerst muss dabei eine Leistungsbilanz auf der tiefsten Substrukturebene (z.B. Compound im Compound) ausgeführt werden. Anschließend erfolgen nacheinander Bilanzen auf der nächst höheren bis hin zur obersten Modellebene. Dieses Vorgehen sichert, dass alle Leistungsanteile auf allen Modellebene in die Gesamtbilanz eingebracht werden. Zum Einfügen der Leistungsbilanz eines Compounds ist die Strukturansicht des jeweiligen externen oder internen Elementtyps zu öffnen (*Rechtsklick auf den Elementtyp – Öffnen - Strukturansicht*). Anschließend muss, wie in **Kapitel 6.8.2.1** beschrieben, die Leistungsbilanz eingefügt werden.

Die Vollständigkeit der Leistungsbilanz kann mit Hilfe der Summe aller Leistungen PSum (vgl. **Kapitel 6.8.1**) kontrolliert werden. Sind alle Leistungsanteile auf allen Modellebenen korrekt erfasst, gilt im Gesamtmodell PSum = 0.

### 6.8.3 Leistungssensoren

In den Elementbibliotheken *translatorische Mechanik, rotatorische Mechanik, Elektrik* und *Hydraulik* stehen entsprechende Leistungssensoren zur Verfügung.

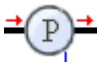
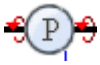
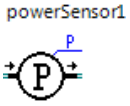
| | **Sensor** | **Ergebnisgröße** |
|---|---|---|
| translatorische Mechanik | Leistungssensor  | Mechanische Leistung $P_{mech}$ |
| rotatorische Mechanik | Leistungssensor  | Mechanische Leistung $P_{mech}$ |
| Elektrik | Leistungssensor | Elektrische Leistung $P_{el}$ |
| Hydraulik | Leistungssensor   | Hydraulische Leistung $P_{hydr}$ |

**Tabelle 6.7:** Leistungssensoren

Die Leistungssensoren können in Reihe zwischen zwei Elemente der jeweiligen Bibliothek geschaltet werden. Sie messen den Leistungsfluss in der Verbindung zwischen diesen beiden Elementen.

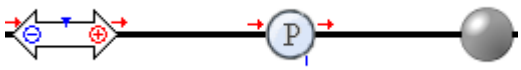


**Bild 6.80:** Einfügen eines Leistungssensors in der translatorischen Mechanik

## 6.9 Troubleshooting: Was tue ich, wenn...?

In diesem Kapitel und in **Kapitel 11** finden Sie Antworten zu den häufigsten Fragen und Hinweise, wie Sie auf Warnungen und Fehlermeldungen reagieren. Außerdem erhalten Sie Tipps, wie Sie die Lösereinstellungen hinsichtlich Genauigkeit und Rechengeschwindigkeit optimieren. Für das Verständnis der Begriffe zum Löser schlagen sie bitte in **Kapitel 12** nach.
Während der Simulation können verschiedene Probleme auftreten. Problemlösungen finden Sie in den **folgenden Kapiteln** und in **Kapitel 11.2**.

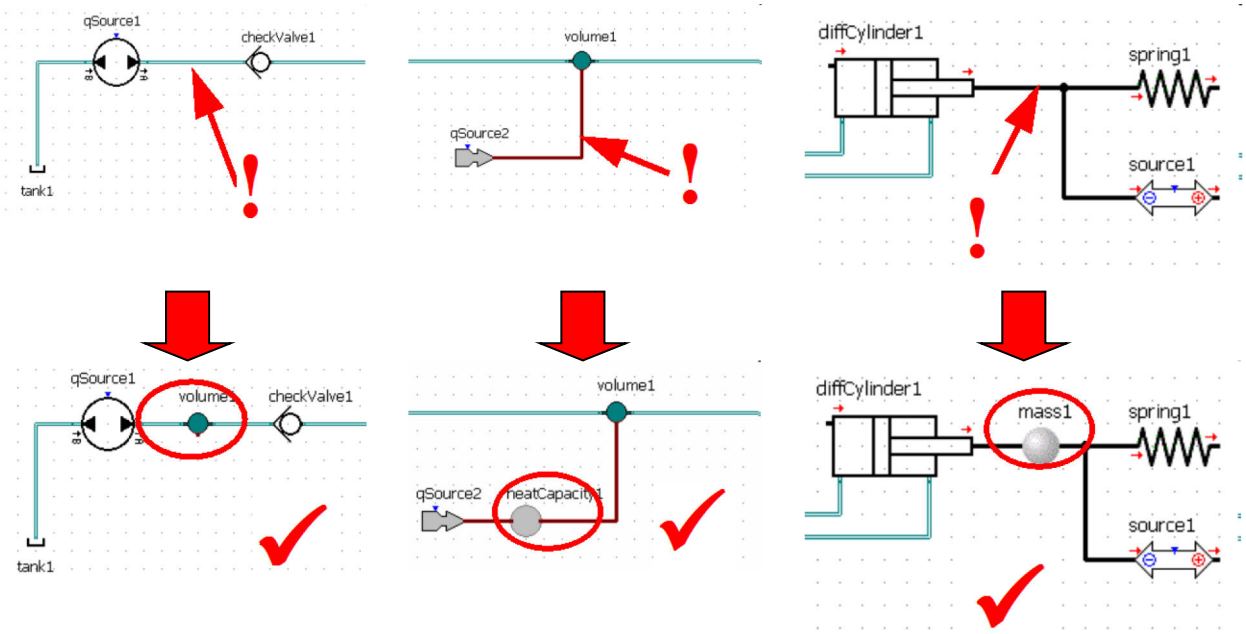
Im **Kapitel 11.2** sind folgende Problemlösungen aufgeführt:

- *Es gibt Parameter, die keinen Wert haben!*
- *Das System ist singulär!*
- *Es konnten keine konsistenten Anfangswerte berechnet werden!*
- *Kann nach einer Unstetigkeit keinen konsistenten Zustand herstellen.*
- *Anfangswertberechnung abgebrochen*
- *Der berechnete Anfangswert entspricht nicht meinen Erwartungen.*
- *Zyklus in der Event-Iteration gefunden.*
- *Die erforderliche Genauigkeit kann nicht erreicht werden.*
- *Das Verfahren konvergiert nicht.*
- *Lineares Gleichungssystem nicht lösbar und Verfahrensordnung und Schrittweite nicht verkleinerbar*
- *Überschreitung von Minimum/ Maximum der Zustandsgrößen*
- *Ausnahme während der Berechnung…*
- *Fehler bei der Berechnung des Dämpfungsfaktors für die Newton-Korrektur*
- *Der stationäre Zustand (statisches Gleichgewicht) konnte nicht gefunden werden!*
- *Berechnung des statischen Gleichgewichts abgebrochen*
- *Die absolute Toleranz wurde kleiner Null gewählt!*
- *Die relative Toleranz wurde kleiner Null gewählt!*
- *Die Anzahl der Zustandsgrößen und die Anzahl der Gleichungen sind verschieden voneinander!*
- *Die Berechnung über die Zeit dauert sehr lange.*
- *Wie kann ich die Richtigkeit / Genauigkeit der Ergebnisse prüfen?*

### 6.9.1 Typische Modellierungsprobleme

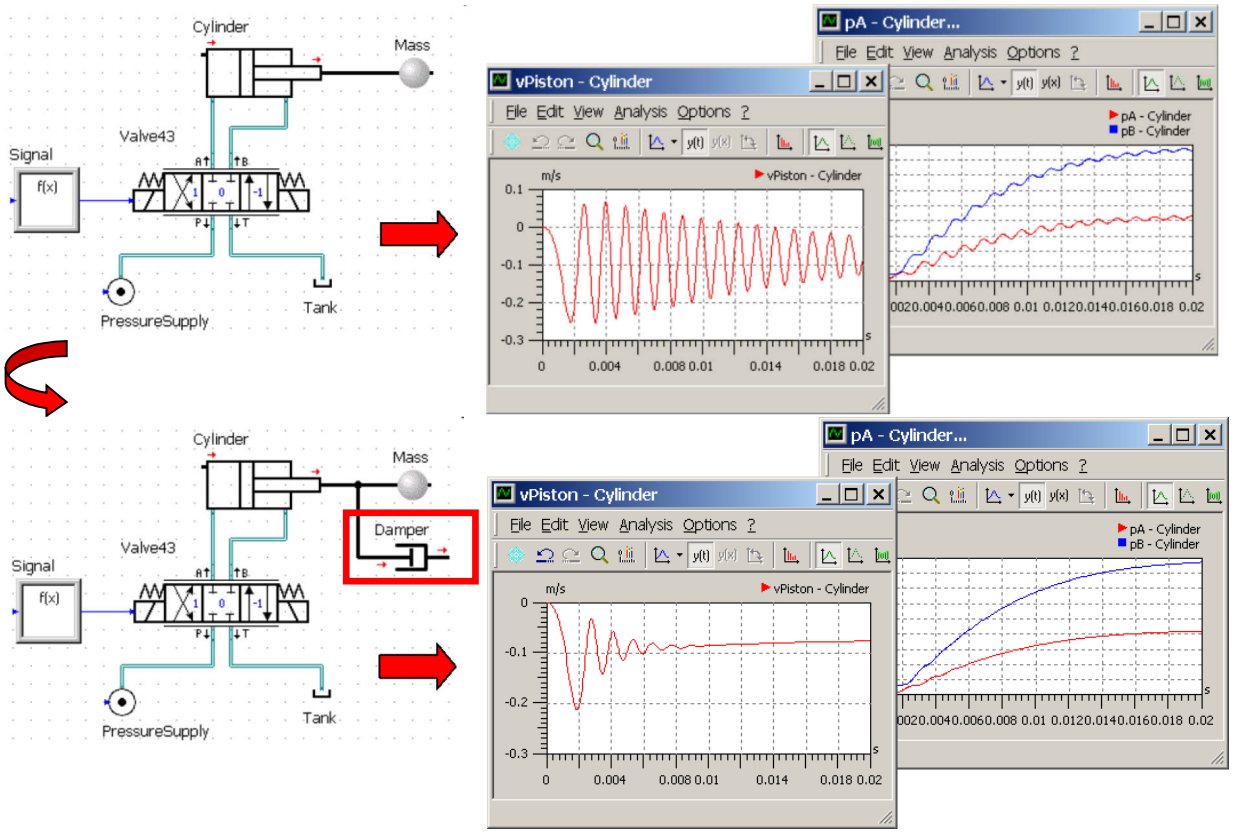
Lassen Sie uns jetzt einige häufig auftretende Modellierungsprobleme betrachten. Für eine tiefer greifende Fehlersuche sollte **Kapitel 11.2** hinzugezogen werden.

**Problem 1:** Fehlende Kapazitäten (Massenträgheiten, Volumen etc.) zwischen den Elementen




Es sollte unbedingt überprüft werden, ob in ähnlichen Modellen wirklich auf die Kapazitäten verzichtet werden kann. Unerwartetes Modellverhalten oder numerische Probleme bei der Lösung sind häufige Folgen solcher Modelle.

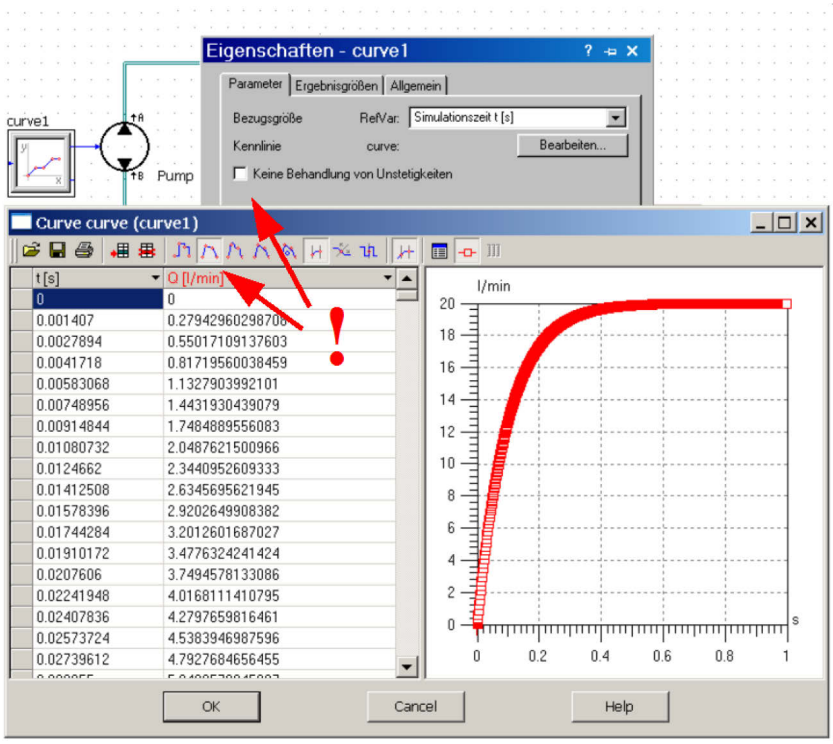
**Problem 2:** Fehlende Dämpfung



Fehlende oder zu geringe Dämpfungen lassen Schwingungen oftmals zu langsam abklingen oder gar erst entstehen. Unerwartetes Modellverhalten bzw. verlangsamte Simulationsrechnungen sind die Folge. Zu geringe Dämpfungen werden auch häufig bei elastischen Anschlägen verwendet.

**Problem 3:** Kennlinien mit vielen Punkten bei linearer Interpolation

Niedrige Rechengeschwindigkeit durch Unstetigkeitsbehandlung!




Spline-, Kreis-, hyperbolische Interpolation oder quadratische Approximation werden empfohlen, da diese Interpolationsarten **keine Unstetigkeiten** hervorrufen!

## 6.9.2 Es können keine konsistenten Anfangswerte berechnet werden

Für die Problemlösung sollten Sie das Tracing (siehe auch **6.1.1.4**) nutzen. Im folgenden Abschnitt werden einige wichtige Lösungsansätze vorgestellt.

Schritt (1): Verstehen des Problems:

Das System von Gleichungen kann bei t=0 mit den vorgegebenen Anfangswerten ("fixiert" oder "frei") nicht erfüllt werden.

Schritt (2): Suchen Sie die Ursache des Problems:

Keine Zustandsgröße hat Einfluss auf die Residuen!

↓

Modellierungsfehler!

Prüfen Sie die Fehlermeldung im Ausgabebereich. Suchen Sie die Residuen, welche oberhalb des Toleranzbereiches (normalerweise **`absTol`**, wenn nicht intern skaliert) liegen. Finden Sie die Zustandsgröße(n), welche die Residuen beeinflussen.

Schritt (3): Versuchen Sie das Problem durch...

... Ändern der entsprechenden Anfangswerte ( / ) oder

... Änderungen am Modell zu lösen.

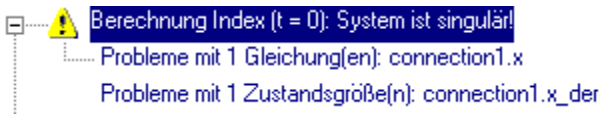
## 6.9.3 Das System ist singulär

Schritt (1): Verstehen des Problems:

Es gibt (mindestens) eine Zustandvariable, die keine der Gleichungen beeinflusst, bzw. es gibt (mindestens) eine Gleichung, die nicht durch irgendeine Zustandvariable (oder Ableitung) beeinflusst wird. Die Ursachen für die Singularität sind meist Modellierungsfehler (z. B. fehlende Kapazität, Masse, Volumen usw.). Es kann aber auch sein, dass ein Eintrag in der Jacobimatrix zu klein ist, und daher als "Null" interpretiert wird.

Schritt (2): Finden Sie die Ursache für das Problem:

Überprüfen Sie die Fehlermeldung im Ausgabebereich. Finden Sie die Zustandsgröße bzw. Gleichung, die singulär ist.




Schritt (3): Versuchen Sie das Modell entsprechend zu verändern!

## 6.9.4 Langsame Simulation

Schritt (1): Verstehen des Problems

Es gibt hauptsächlich 3 Gründe für eine langsame Simulation:

a) eine Schwingung mit hoher Frequenz und geringer Dämpfung

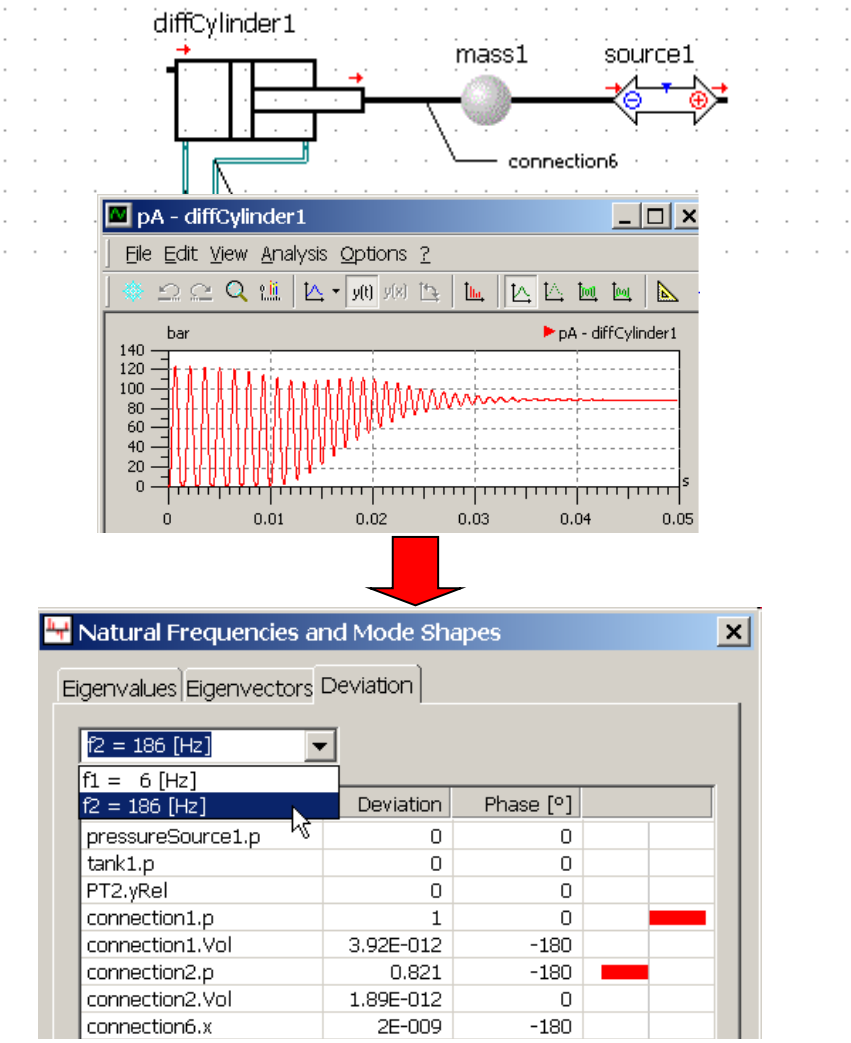
b) sehr viele Unstetigkeiten

c) sehr viele ungültige Schritte

Schritt (2): Finden der Ursache

a) Überprüfen der Eigenfrequenzen um die Modellkomponenten, die für die unerwartet hochfrequenten Schwingungen verantwortlich sind zu lokalisieren. Überprüfen Sie, ob diese Komponenten korrekt modelliert worden sind und ob beispielsweise ausreichende Dämpfung vorhanden ist.

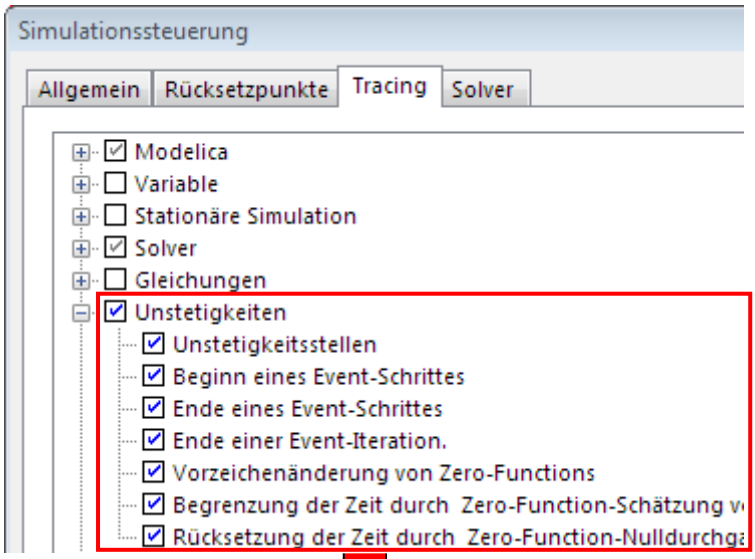
b)

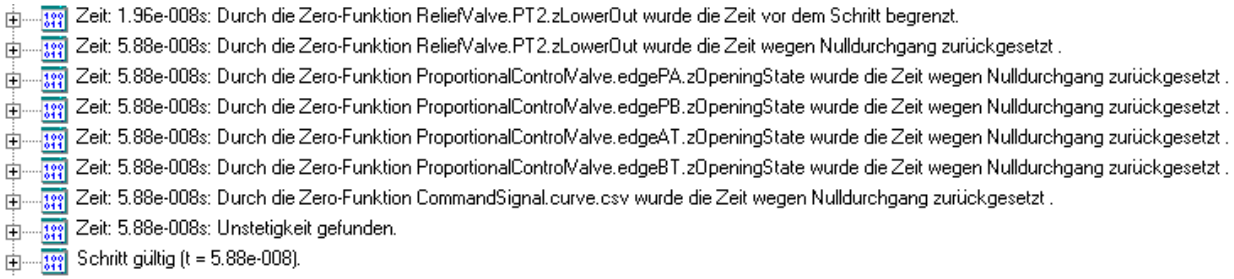


c) Prüfen der Unstetigkeiten

Mögliche Gründe für zu viele Unstetigkeiten sind

- Reibung (stick/slip)
- Kennlinien mit linearer Interpolation
- Instabiles Ventilverhalten (Öffnen/Schließen)
- Hin- und Herspringen an der Begrenzung
- ...

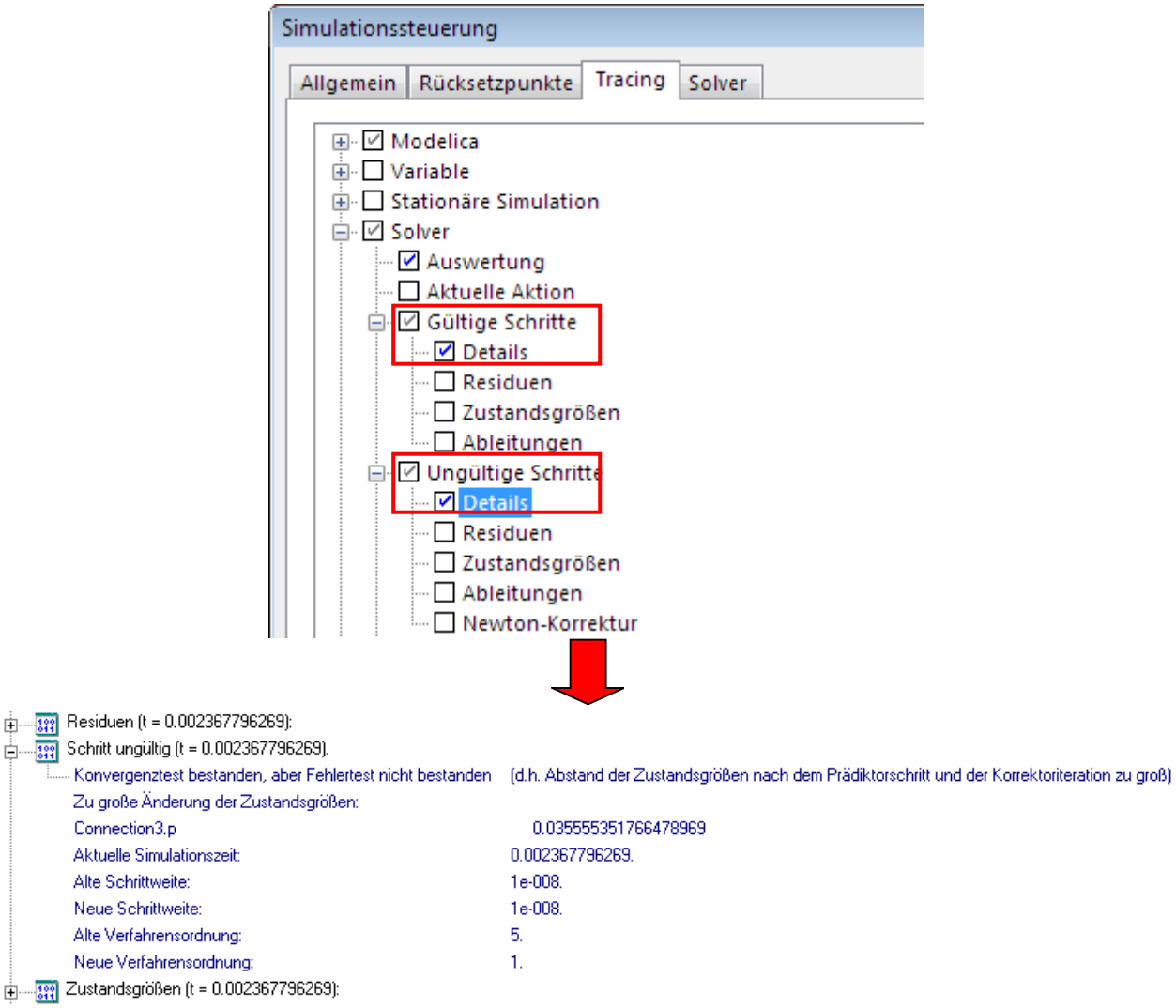



d) Prüfen des Auftretens von ungültigen Schritten!

Mögliche Gründe für zu viele ungültige Schritte sind

- in hohem Grade nichtlineares Verhalten
- schlechte Jacobi-Matrizen
- versteckte Unstetigkeiten ('noEvent(...)')




Schritt (3): Verringern Sie die Toleranzen und die Schrittweite!

### 6.9.5 Allgemeine Hinweise

Die absolute und die relative Toleranz (**`absTol`** und **`relTol`**) sollten nicht erhöht, sondern nur (wenn möglich) verringert werden.
Die minimale Schrittweite (**`dtMin`**) sollte für einige Hydraulik-Modelle mit kleiner Zeitkonstante (z. B. verursacht durch kleine Volumina) verringert werden.
Auswahl eines entsprechend großen Wertes für die minimale Schrittweite (**`dtProtMin`**) verhindert das Entstehen großer Datenmengen. Die Standard-einstellung (0.001 s) ist für eine Simulationszeit von 1 s angemessen. Für viel längere Simulationszeiten sollte der Wert von **`dtProtMin`** ebenfalls erhöht werden. Wenn Sie hochfrequente Schwingungen beobachten möchten, sollten Sie vor Simulationsstart den Wert von **`dtProtMin`** verringern.
Der Wert von **`dtDetect`** sollte nicht geändert werden. Er wird nur für das Erkennen von Unstetigkeiten genutzt.

SIMULATION X®

Powered by ITI

Results and Documentation

# Results and Documentation

# 7 Ergebnisauswertung, Dokumentation und allgemeine Einstellungen

## 7.1 Übersicht

Jedes Element eines Modells errechnet in Abhängigkeit von seiner Funktionalität und Komplexität eine Anzahl von Ergebnisgrößen. Zu jedem Ausgabezeitschritt werden alle Ergebnisgrößen mit aktivem Protokollattribut aller Elemente des Modells bereitgestellt. D. h., während der Simulations-Rechnung entsteht eine *große Menge Daten*.

Der folgende Abschnitt soll Sie über die Möglichkeiten des Umgangs mit den Simulationsergebnissen informieren. Abhängig von den jeweiligen Auswertungserfordernissen können Sie

- Momentananzeigen und
- Ergebnisfenster

zum Einsatz bringen.

Das Ausdrucken der Ergebnisse ist ebenfalls möglich (siehe **7.4**).

Um erstellte Modelle, Parameter und Ergebnisse mehrfach zu nutzen, besteht die Möglichkeit, diese zu speichern.

Dabei werden folgende Optionen vom Programm unterstützt:

- Ein gerechnetes Modell speichert seine momentanen/aktuellen und protokollierten Ergebnisse mit ab. Das geschieht unabhängig davon, ob zum jeweiligen Zeitpunkt des Speicherns ein Ergebnisfenster geöffnet ist oder nicht.
- Geöffnete Ergebnisfenster können mit dem Modell abgespeichert und wieder geladen werden.
- Einzelne Ergebnisse können als Datei auf einem Datenträger abgespeichert werden. Dateiformat, Dateiname und Verzeichnis sind dabei vom Anwender wählbar (vgl. **7.3.7**). Dies geschieht über das Menü ***Datei/Speichern***.

## 7.2 Momentananzeige

Zur Beurteilung der Güte eines Modells, z. B. in der Phase der Modellerstellung bzw. Fehlersuche, ist es oftmals ausreichend, ausgewählte Ergebnisgrößen zum jeweiligen Zeitschritt zu bewerten. Beispiele solcher Bewertungen sind:

- Überschreitet eine Ergebnisgröße einen bestimmten Grenzwert?
- Liegt eine Ergebnisgröße in einem bestimmten Bereich?
- Ändert sich eine Ergebnisgröße überhaupt?

Möglichst viele Ergebnisgrößen ständig im Blick zu behalten, ist insbesondere bei großen Modellen nicht ganz einfach. Wertvolle Dienste für die genannten Aufgaben leisten die in SimulationX bereitgestellten Momentananzeigen.

### 7.2.1 Momentananzeige öffnen

Momentananzeigen werden über das Menü ***Einfügen/Momentananzeige*** **(Bild 7.1)** erzeugt. Sie können jede Ergebnisgröße mit einer beliebigen Momentananzeige verknüpfen. Momentananzeigen können, wie Elemente auf dem Arbeitsblatt frei platziert werden. Beachten Sie bitte die Einstellungen in **Kapitel 7.6.1.**

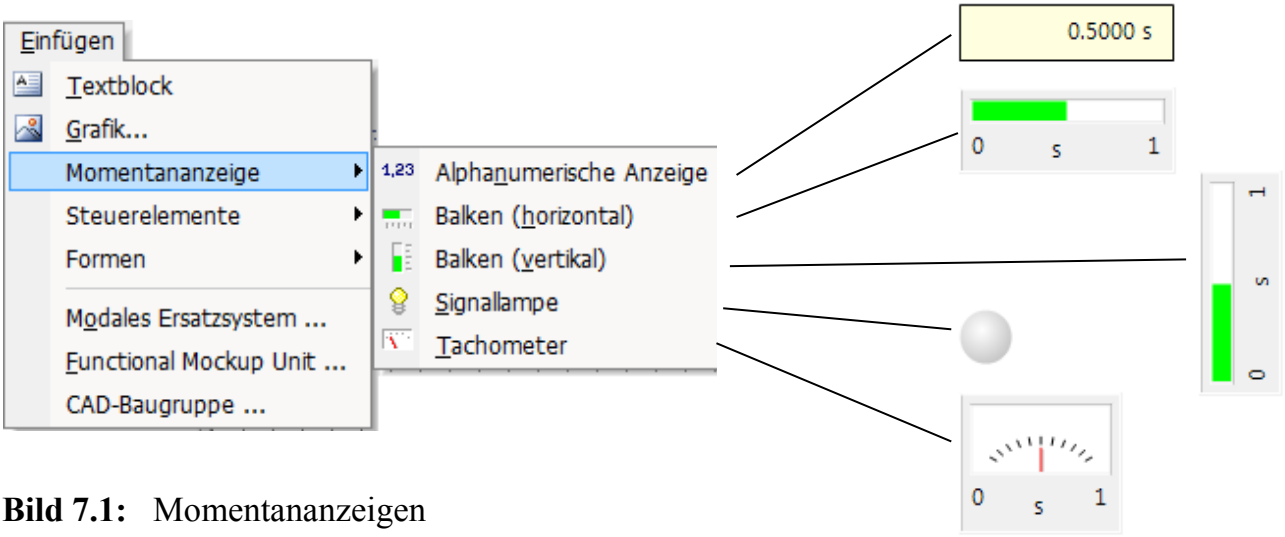



**Bild 7.1:** Momentananzeigen

### 7.2.2 Verknüpfung mit Ergebnisgrößen

Dazu öffnen Sie mit Doppelklick auf eine Momentananzeige deren Eigenschaftsfenster. Auf der Seite ***Momentananzeigen*** werden in einer Auswahlbox alle Elemente und Ergebnisgrößen dargestellt und können durch Anklicken ausgewählt werden (siehe **Bild 7.2**).

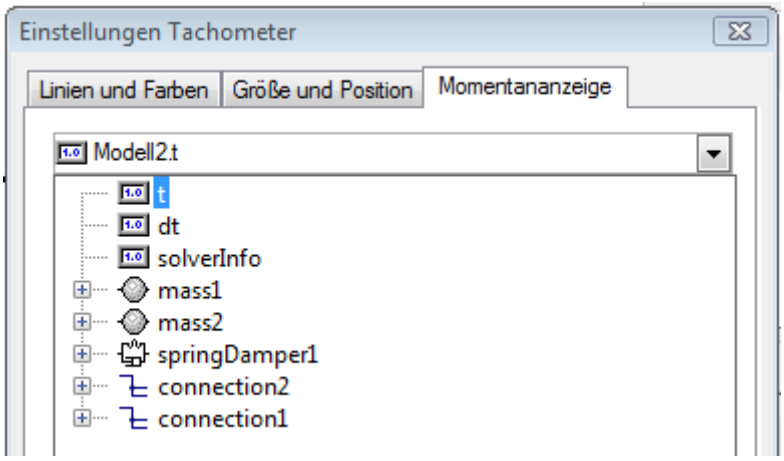



**Bild 7.2:** Auswahl im Register *Momentananzeige*

Die Maßeinheit ist entsprechend dem Eigenschaftsdialog des Elementes oder Modells voreingestellt und kann nicht geändert werden.

Weiterhin besteht die Möglichkeit, einen Min- und einen Max-Wert vorzugeben (siehe **Bild 7.3**).

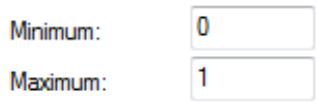



**Bild 7.3:** Vorgabe darstellbarer Bereich

Bei den Balkenanzeigen und dem Tacho entspricht das dem darstellbaren Bereich.

## 7.2.3 Weitere Einstellungen

### 7.2.3.1 Linien und Farben

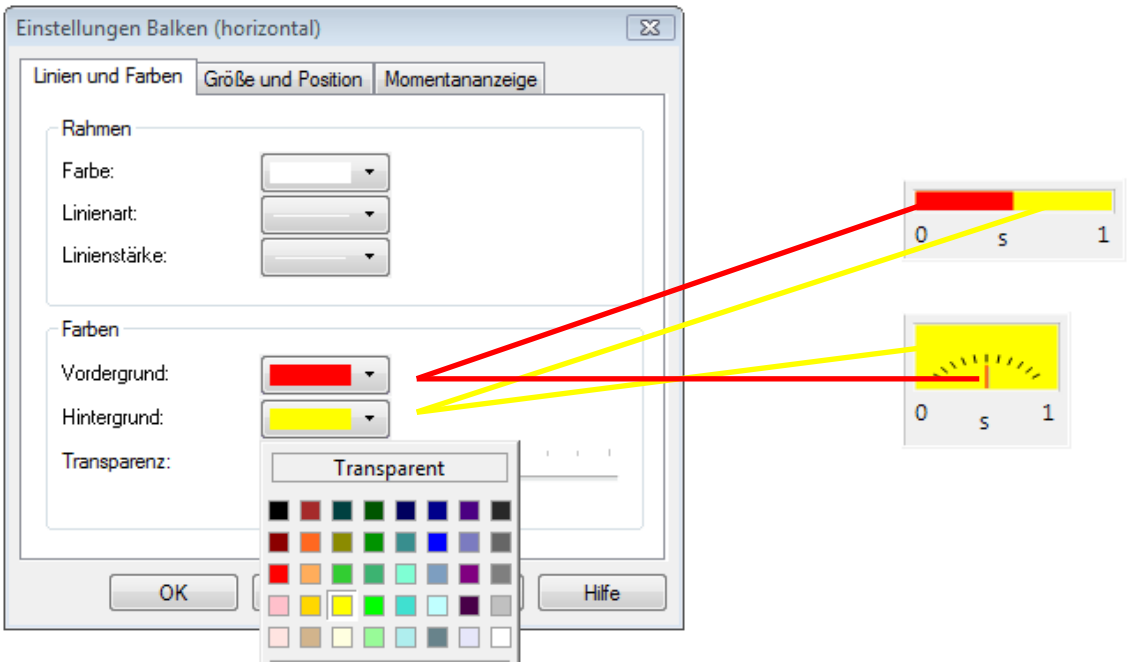



**Bild 7.4:** Zuordnung der einstellbaren Farben

Mittels des Registers ***Linien und Farben*** kann die farbliche Darstellung angepasst werden (s. **Bild 7.4**).

### 7.2.3.2 Größe und Position

Über dieses Register können Größe und Position geändert werden. Dies ist auch über das Verschieben bzw. Aufziehen des Fensters der Momentananzeige möglich.

## 7.2.4 Anmerkungen

Momentananzeigen sind überall im Modell zur Ergebnisdarstellung einsetzbar, wo der Wert und nicht der komplette Verlauf einer Größe über der Zeit von Bedeutung ist. Sie sollten bedenken, dass zur Aktualisierung bei jedem Ausgabezeitschritt Rechenzeit benötigt wird. D.h. mit steigender Anzahl von Anzeigeobjekten sinkt die Abarbeitungsgeschwindigkeit.
Momentananzeigen werden mit dem Modell gespeichert.

## 7.3 Ergebnisfenster

Ergebnisfenster dienen der grafischen Darstellung der Simulationsergebnisse sowie deren Auswertung. Sie werden bereits während der Simulationsrechnung aktualisiert und ermöglichen so die ständige Kontrolle über den Berechnungsverlauf.

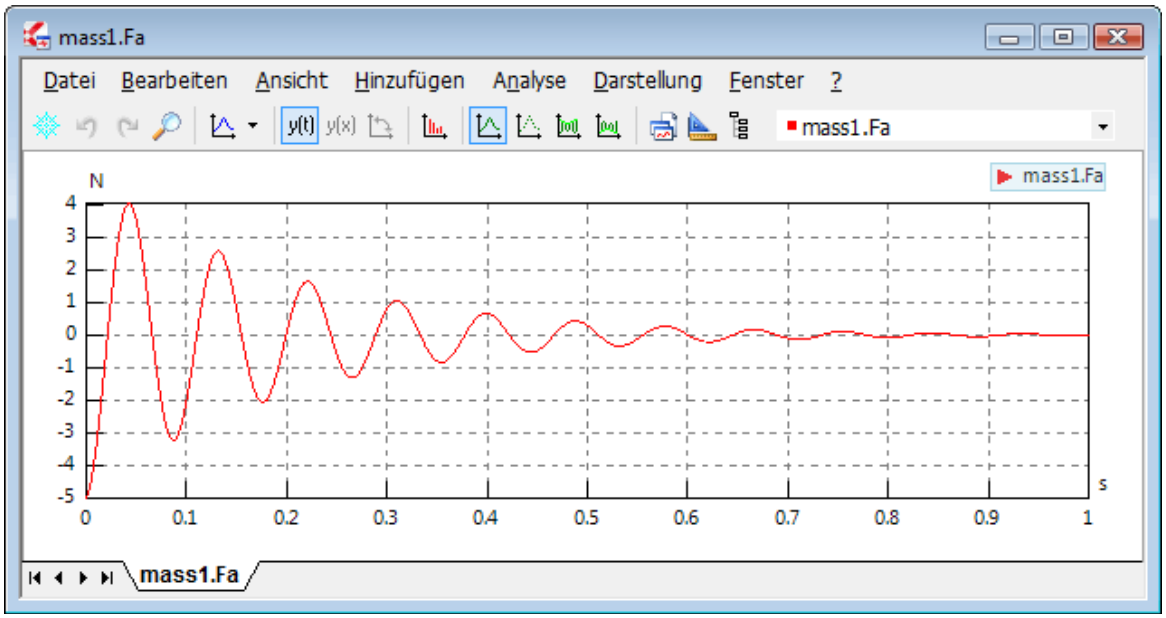



**Bild 7.5:** Ergebnisfenster mit Ergebnisverlauf

### 7.3.1 Grundlagen

Um für eine Modellvariable einen Ergebnisverlauf darstellen zu können müssen einige Voraussetzungen gegeben sein.

1. Die Variable muss als Ergebnisgröße im jeweiligen Modell vereinbart worden sein. Ergebnisgrößen sind alle Variablen, die nicht explizit als Parameter deklariert wurden.
2. Das Protokoll-Attribut der Variablen muss aktiviert sein. Diese Aktivierung können Sie im Eigenschaftsdialog der jeweiligen Komponente oder im Modellexplorer vornehmen. Eine weitere Möglichkeit zur Aktivierung der Protokoll-Attribute bietet die Ergebnisleiste im Ergebnisfenster selbst.

Die während der Rechnung aufgezeichneten Werte werden standardmäßig in temporäre Dateien gespeichert. Die Performance lässt sich erhöhen, indem die Speicherung im Arbeitsspeicher (RAM) vorgenommen wird oder zumindest die Puffergröße, die jeder Ergebnisgröße im Arbeitsspeicher reserviert wird, vergrößert wird. Diese Einstellung führt jedoch zu einer deutlichen Erhöhung des Speicherbedarfs und sollte nur bei entsprechend ausgestatteten Rechnern zur Anwendung kommen.
Der Speicherbedarf für die Aufzeichnung von Ergebnissen bei der transienten Simulation lässt sich wie folgt abschätzen:

$$s = n_{prot} \cdot (1 + n_{timeSteps} + n_{eventSteps}) \cdot 8$$

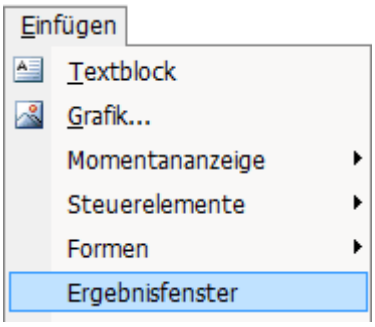
| | |
|---|---|
| s | Speicherbedarf in Byte |
| $n_{prot}$ | Anzahl der protokollierten Variablen |
| $n_{timeSteps}$ | Anzahl der gültigen Zeitschritte* |
| $n_{eventSteps}$ | Anzahl der Event-Schritte* |

* Diese Werte sind abhängig von den Einstellungen für die Protokollierung von Ergebnissen im Eigenschaftsdialog der transienten Simulation sowie dem Wert von `dtProtMin`.

## 7.3.2 Ergebnisfenster öffnen

Ergebnisfenster lassen sich auf unterschiedlichen Wegen öffnen. Einmal geöffnete Ergebnisfenster werden mit dem Model gespeichert und stehen nach dem erneuten Laden des gespeicherten Modells sofort wieder zur Verfügung.

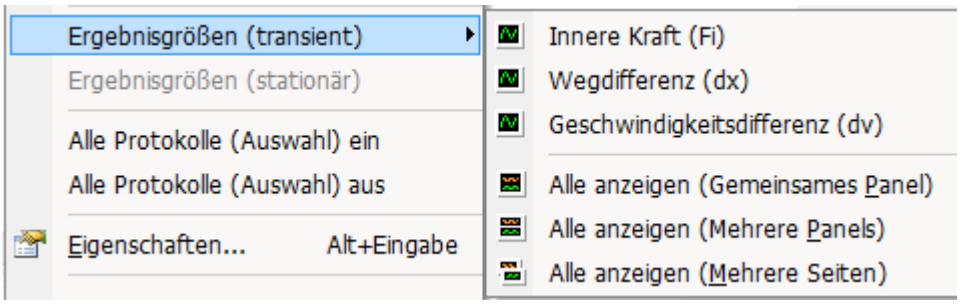
### 7.3.2.1 Leere Ergebnisfenster




Ein leeres Ergebnisfenster, d.h. eine Ergebnisfenster ohne Ergebnisverlauf lässt sich über den Menüpunkt ***Einfügen*** oder über die entsprechende Schaltfläche im Ergebnisfenstermanager anlegen. Leere Ergebnisfenster können per Drag & Drop (siehe **7.3.2.3**) oder mittels Einfügen zuvor kopierter Verläufe gefüllt werden. Beim Erzeugen eines leeren Ergebnisfensters wird automatisch die Ergebnisleiste aktiviert. Dies ermöglicht die schnelle Auswahl der Größen, die in dem neuangelegten Ergebnisfenster dargestellt werden sollen.

### 7.3.2.2 Kontextmenü (Strukturansicht)

Die Kontextmenüs von Elementen und Verbindungen in der Strukturansicht gestatten das direkte Anlegen von Ergebnisfenstern für einzelne Variablen oder Variablengruppen.



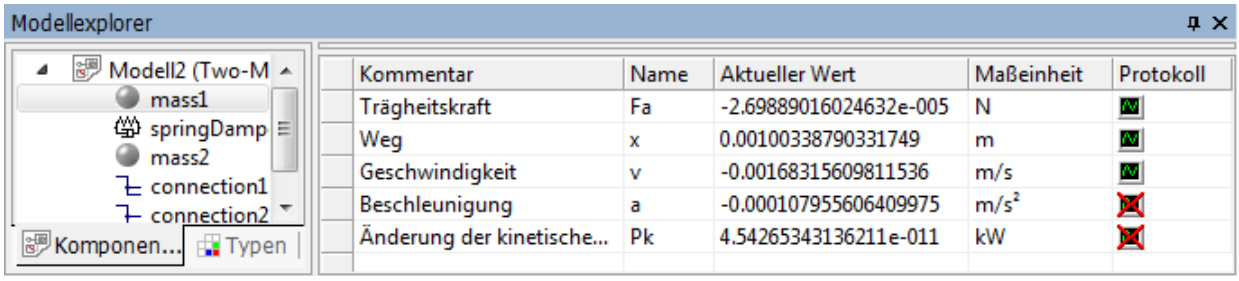

Werden mehrere Verläufe in das Ergebnisfenster übernommen, stehen folgende Optionen zu Verfügung

- Anordnung der Verläufe innerhalb eines Panels

- Anordnung der Verläufe in verschiedenen Panels innerhalb einer Seite
- Anordnung der Verläufe auf verschiedenen Seiten (jeweils ein Verlauf pro Seite)

Bei einer Mehrfachselektion werden die jeweils zusammengehörigen Variablen gleichen Namens immer gemeinsam in einem Panel dargestellt.

### 7.3.2.3 Modell-Explorer




Aus dem Modellexplorer heraus lassen sich Ergebnisfenster per ***Drag & Drop*** erstellen. Befolgen Sie dazu die folgenden Anweisungen:

1. Bewegen Sie den Mauszeiger über ein aktiviertes Ergebnisvariablen-Symbol.
2. Klicken Sie mit der linken Maustaste auf das Symbol und halten Sie die Maustaste gedrückt.
3. Bewegen Sie den Mauszeiger in eine Modellansicht des betreffenden Modells. Dabei kann sich um die Strukturansicht, eine 3D, oder auch eine Textansicht handeln.
4. Lassen Sie die gedrückte Maustaste los. SimulationX erstellt daraufhin ein neues Ergebnisfenster, das den gewünschten Verlauf enthält.

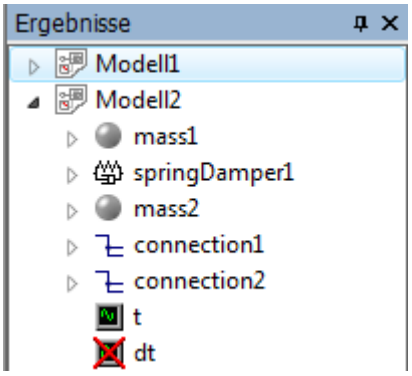
Sie können den gewünschten Ergebnisverlauf auch in ein bereits geöffnetes Ergebnisfenster ziehen. In diesem Fall wird der Verlauf zu den bereits im Fenster vorhandenen Verläufen hinzugefügt. Je nachdem, wo Sie die Drag & Drop-Operation beenden, wird der hinzugefügte Verlauf in ein bestehendes Panel integriert, oder als erster Verlauf eines neuen Panel oder einer neuen Seite dem Ergebnisfenster hinzugefügt. Einzelheiten hierzu erfahren Sie im Abschnitt **7.3.4**.

### 7.3.2.4 Eigenschaftsdialog (Modellelemente und Verbindungen)

Der Eigenschaftsdialog für Elemente und Verbindungen fasst die Ergebnisgrößen der jeweiligen Komponente auf einer oder mehreren Dialogseiten zusammen.

Über das entsprechende Symbol können Sie die Protokollierung der betreffenden Variable ein bzw. ausschalten. Sie können das Symbol per Drag & Drop jedoch auch in eine Ansicht des Modells oder in ein bestehendes Ergebnisfenster ziehen, um ein neues Ergebnisfenster zu erzeugen oder einen neuen Verlauf einem bestehenden Ergebnisfenster hinzuzufügen.

### 7.3.2.5 Ergebnisleiste (Ergebnisfenster)



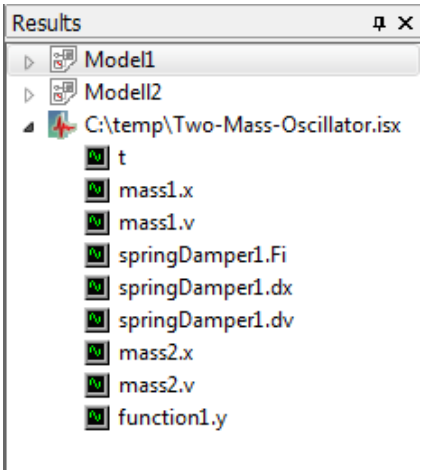

Die Ergebnisleiste stellt die Ergebnisvariablen aller aktuell geöffneten Modelle als Baum dar. Um den Verlauf einer Variablen darzustellen ziehen Sie das Variablensymbol in den Darstellungsbereich des Ergebnisfensters.

Um das Protokollattribut einer Variablen ein- bzw. auszuschalten, öffnen Sie das Kontextmenü für die jeweilige Variable und rufen den betreffenden Befehl auf.

Für Vergleiche mit vorhergehenden Simulationen können die Ergebnisse in ein vorhandenes Ergebnisfenster geladen werden. Dafür stehen die Datenformate

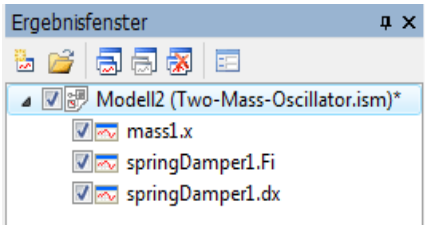
- SimulationX Ergebniskonfiguration (*.rsx)
- ITI SimX-Format (*.rfs)
- Textformat (*.txt, *.csv)
- SimulationX-Projekt (*.isx)

Die Ergebnisvariablen sind dann in der Ergebnisleiste verfügbar und können wie modelleigene Variable genutzt werden.




Beachten Sie dabei, dass eine SimulationX-Ergebniskonfiguration, anders als die anderen Dateiformate, nur die Variablenreferenzen und nicht die darzustellenden Werte selbst enthält. Die vorhandenen Ergebnisvariablen werden über die Ergebnisleiste verfügbar.

#### 7.3.2.6 Ergebnisfenster-Manager




Über den Ergebnisfenstermanager können neue, leere Ergebnisfenster erzeugt oder zuvor als Datei gespeicherte Ergebnisfenster geladen werden. Markieren Sie dazu das Modell, dem Sie das neue Ergebnisfenster zuordnen wollen, und betätigen Sie dann die entsprechende Schaltfläche aus der zugehörigen Symbolleiste.

Folgende Dateiformate werden für das Erstellen von Ergebnisfenstern unterstützt:

- SimulationX Ergebniskonfiguration (*.rsx)
- ITI SimX-Format (*.rfs)
- Textformat (*.txt, *.csv)
- IEEE-Binärformat (*.bin)
- SimulationX-Projekt (*.isx)

Für SimulationX-Ergebniskonfigurationen gilt das gleiche, wie unter **7.3.2.5**.

#### 7.3.2.7 COM-Interface

Ergebnisfenster können auch über das COM-Interface angelegt werden. Dazu verfügen die Klassen *Application* und *Document* über die Eigenschaft *ResultWindows*.

Die folgenden Codefragmente demonstrieren die Verwendung dieser Klassen und Methoden zur Script-gesteuerten Erstellung von Ergebnisfenstern.

```
ResultWindows.AddYt "model1.mass1.x"

ActiveDocument.ResultWindows.AddYt "mass1.x"
```

### 7.3.3 Aufbau von Ergebnisfenstern

#### 7.3.3.1 Begriffe

| | |
|---|---|
| Seite | grundlegender Bestandteile eines Ergebnisfensters. Enthält Panels. |
| Panel | dient der Darstellung der Ergebnisverläufe. Sind mehrfach innerhalb einer Seite platzierbar. |
| Serie | |
| Transformation | Nachträgliche Modifizierung von Ergebnisverläufen |
| Linie | dienen der Bewertung und Einschätzung von Ergebnis-verläufen (z.B. Mittelwert) |

### 7.3.3.2 Fensterlayout

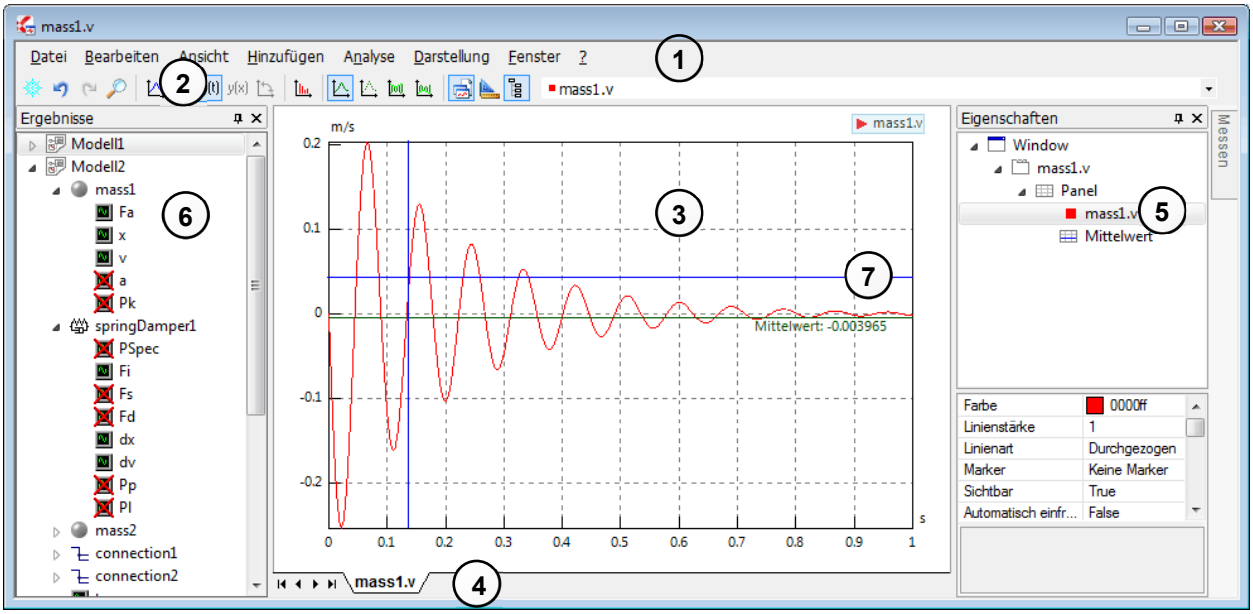



**Bild 7.6:** Ergebnisfenster

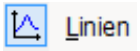
Ein Ergebnisfenster enthält folgende Bestandteile:

- Menüleiste (1)
- Symbolleiste (2)
- Darstellungsbereich (3)
- Seitenregister (4)
- Eigenschaftsleiste (5)
- Ergebnisleiste (6)
- Messleiste (7)
- Legendenleiste für Stationäre Simulation

Über den Menüpunkt ***Ansicht*** können sie das Layout des Ergebnisfensters ändern. Einzelne Bestandteile, wie z.B. Ergebnis- oder Eigenschaftenleiste können fixiert oder gelöst werden.

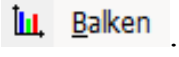
Über das Menü ***Darstellung/Darstellungsart*** sind die folgenden Darstellungsarten für Ergebnisfenster verfügbar:

- **Liniendiagramm**

Es handelt sich um die Standardeinstellung beim Öffnen eines Ergebnisfensters. Dieses wird dann entsprechend **Bild 7.5** dargestellt und ist auch über die Auswahl 

einstellbar.

- **Balkendiagramm (horizontal)**

Ergebnisgrößen können auch als Balken (**Bild 7.7**) dargestellt werden. Dazu klickt man auf die Auswahl 

.

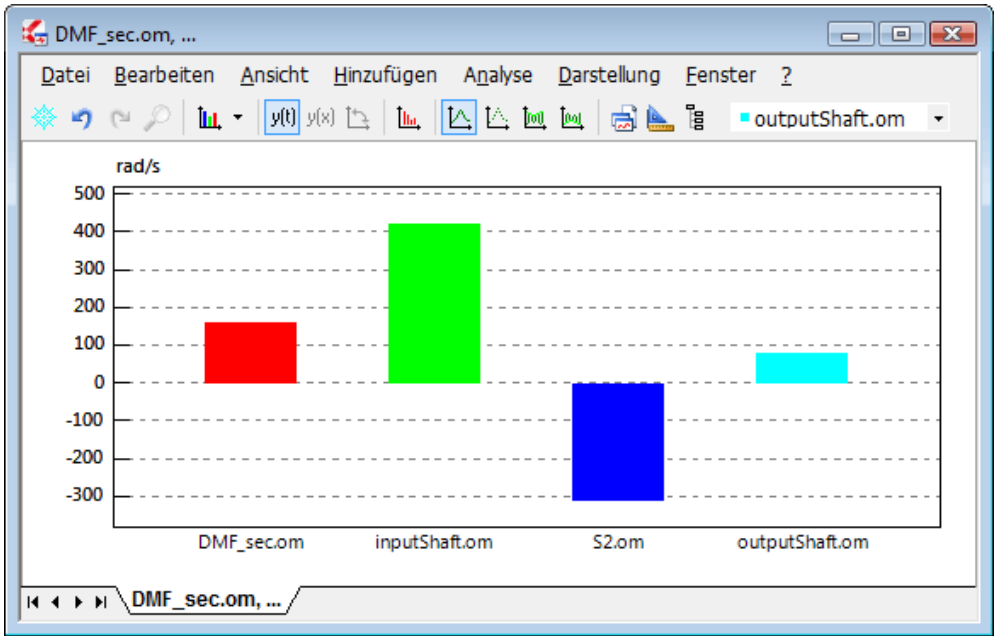



**Bild 7.7:** Ergebnisfenster mit Balkendiagramm

Da jeder Balken zu jedem Ausgabezeitschritt (i. A.) eine andere Höhe hat und das Ergebnisfenster ständig aktualisiert wird, entsteht während der Simulationsrechnung eine *einfache Prozess-Animation*.

- **Polarkoordinaten**

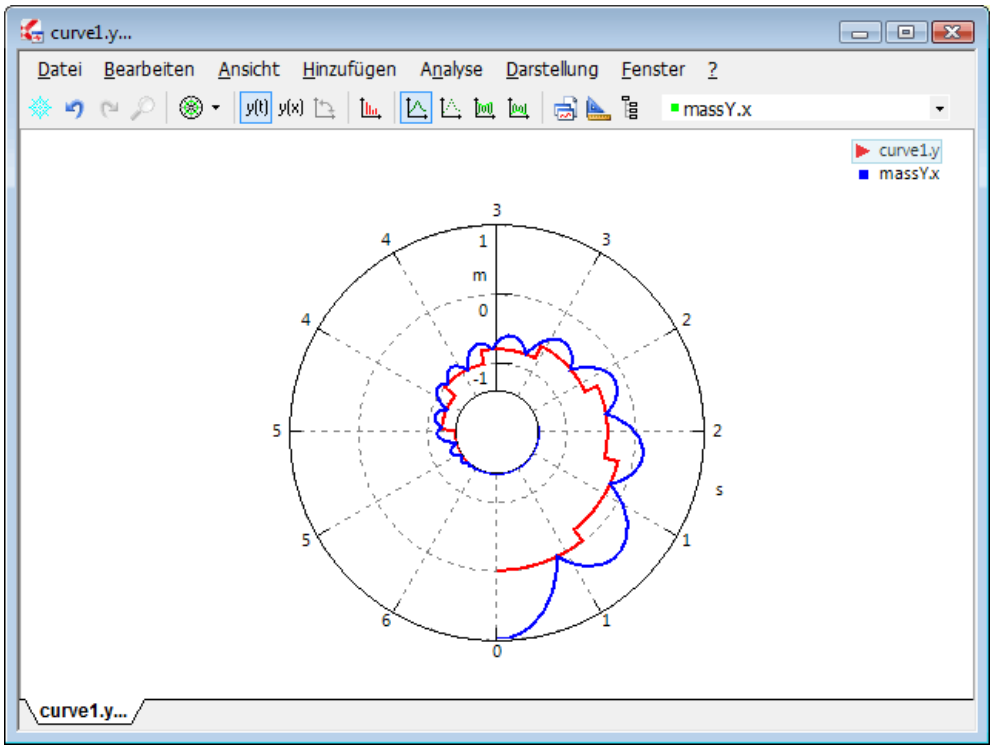



**Bild 7.8:** Polardiagramm

Bestimmte Ergebnisgrößen lassen sich anschaulicher in einem Polardiagramm darstellen, dazu auf die Auswahl Polar klicken. Dabei wird x auf dem Winkel phi und y auf r abgebildet.

- **Balkendiagramm (vertikal)**

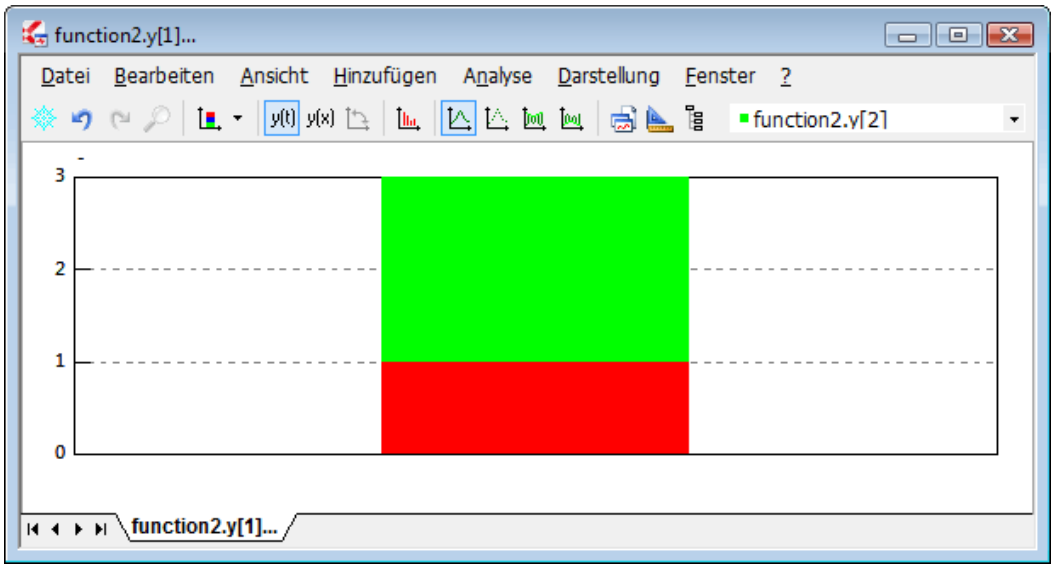



**Bild 7.9:** Balkendiagram mit vertikaler Anordnung

Bei dieser Darstellungsart werden die Ergebnisse in vertikaler Anordnung dargestellt. Somit kann der Anteil einer Ergebnisgröße am Gesamt veranschaulicht werden.

- **Tortendiagramm**

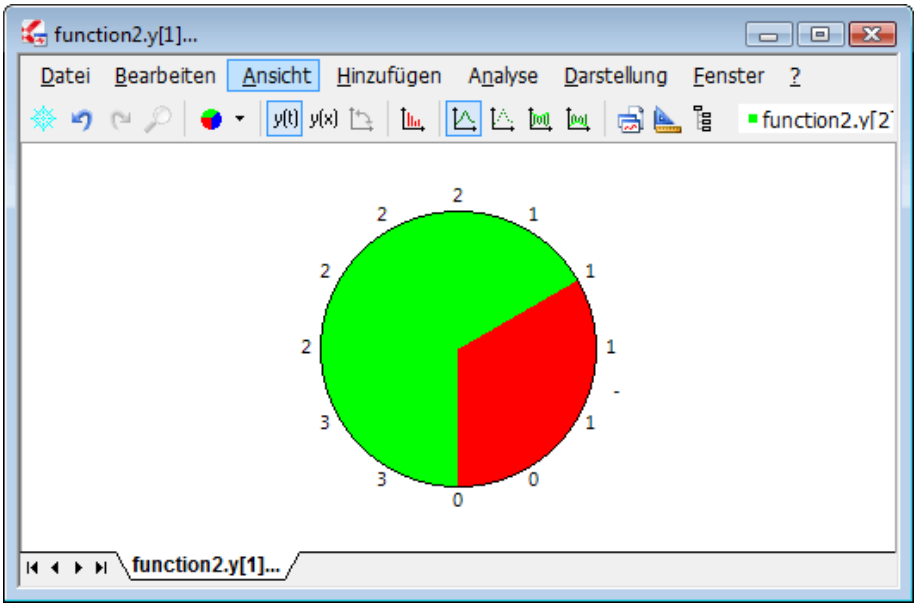



**Bild 7.10:** Tortendiagramm

Für bestimmte Fälle ist die Darstellung in Form des Tortendiagramms anschaulicher.

- **Sankey-Diagramm**

Für die Darstellung von Mengenflüssen (Energiebilanzen) eignet sich das Sankey-Diagramm besonders gut. Dabei werden die Mengen durch mengenproportional dicke Pfeile dargestellt.

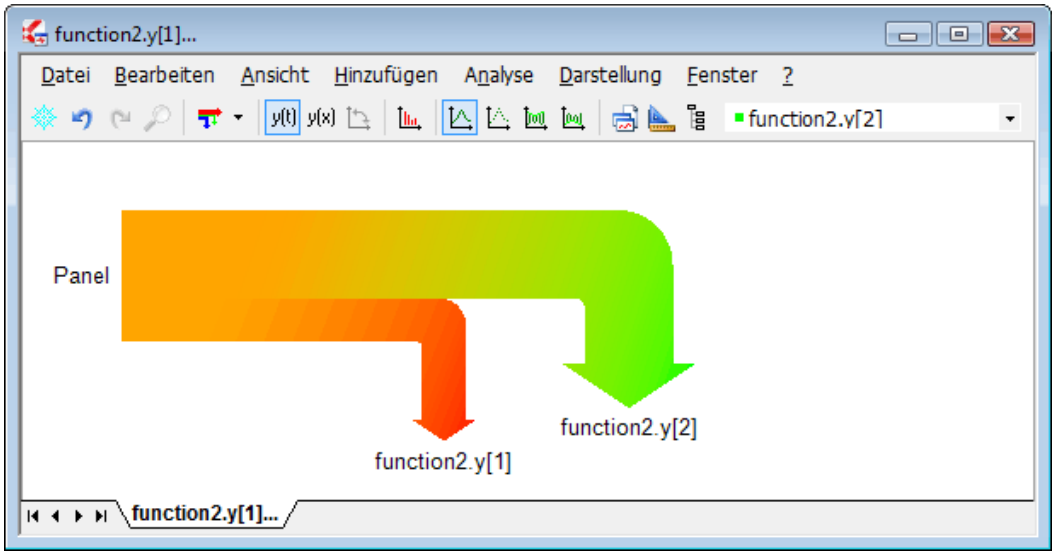



**Bild 7.11:** Sankey-Diagramm

- **y(t) - Darstellung**

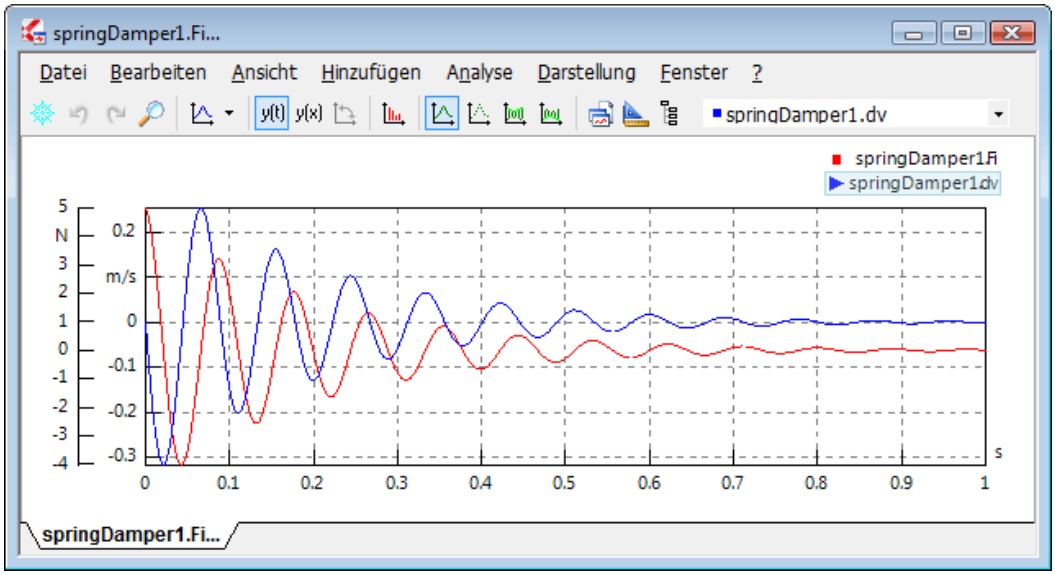



**Bild 7.12:** Darstellung zweier Ergebnisgrößen

**Bild 7.12** erhält man über die Schaltfläche y(t), wobei die Ergebnisgröße über der Simulationszeit dargestellt wird.

Wird mehr als eine Ergebnisgröße mit unterschiedlichen physikalischem Inhalt oder unterschiedlichen Einheiten in einem Fenster dargestellt, wird automatisch für jede Ergebnisgröße eine neue y-Achse erzeugt. Dies ist über den Eigenschafts-dialog änderbar. Entsprechendes gilt auch für die x-Achse.

- **y(x) – Darstellung**

Wird eine gerade Anzahl von Kurven in einem Ergebnisfenster dargestellt, so ist eine Darstellung y(x) möglich.

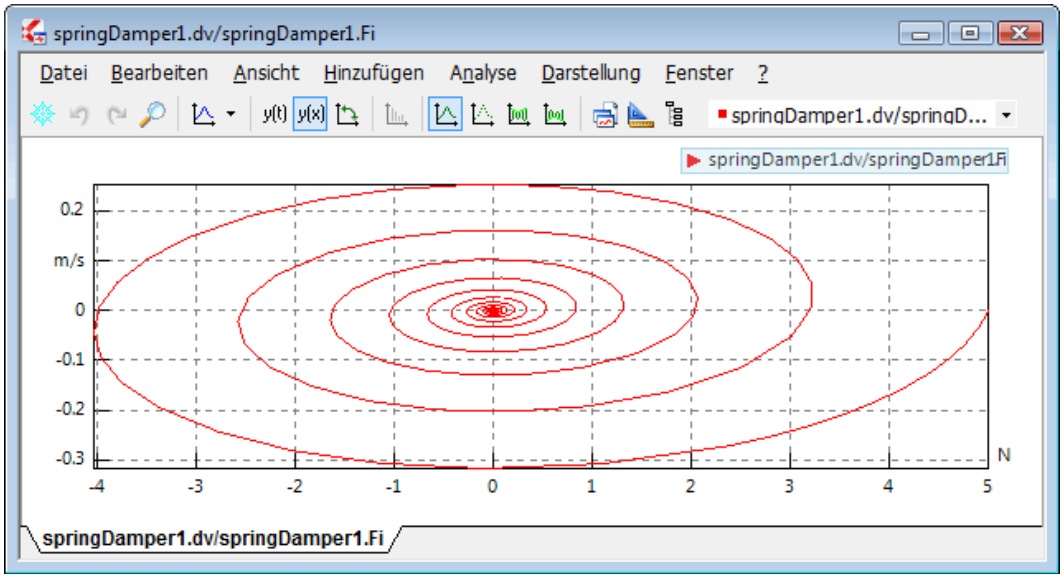



**Bild 7.13:** y(x)-Darstellung

- **Anzeige Ergebnisverlauf als Linie**

Es handelt sich um die Standardeinstellung einer Ergebniskurve. Diese kann mit der Schaltfläche ein- bzw. ausgeschaltet werden. Da sich die Aktion auf die jeweils aktive Kurve bezieht, kann dies zum temporären Ausblenden von Kurven in einem Ergebnisfenster mit mehreren Ergebnisverläufen (z. B. wenn mehrere Kurven aufeinander liegen) genutzt werden.

- **Darstellung der berechneten Punkte**

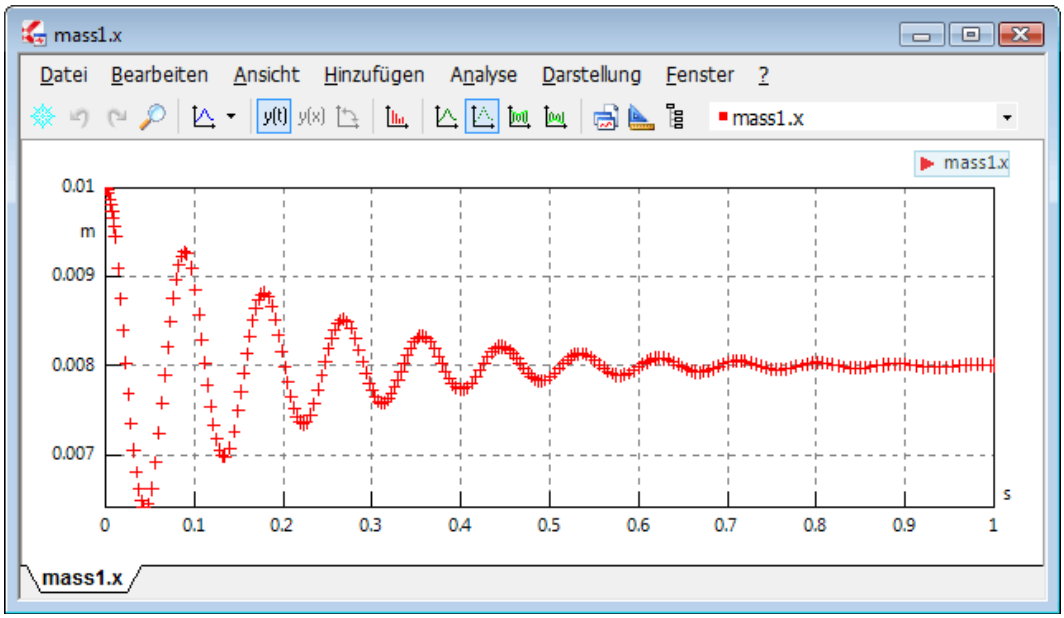



**Bild 7.14:** Ergebnisverlauf in Punktdarstellung

Anstelle der durchgehenden Linie ist eine Anzeige der berechneten Punkte mit der Schaltfläche zuschaltbar. Sie erhalten eine Anzeige, wie in **Bild 7.14**.

Beachten Sie, dass für eine reine Darstellung der Punkte die ***Anzeige Linie*** auszuschalten ist!

## 7.3.4 Verschieben und Kopieren

### 7.3.4.1 Verschieben und Kopieren von Verläufen

Die Ergebnisgrößen verfügen über einen Drag & Drop Mechanismus. Sichtbare Ergebnisgrößen kann man *in die Hand nehmen* und bewegen. Dazu fasst man die gewünschte Ergebnisgröße mit der Maus an und verschiebt diese z. B. in ein neues Panel. Legen Sie einen Verlauf außerhalb eines Ergebnisfensters ab, so wird ein neues Ergebnisfenster erstellt.
Drücken Sie während des *Haltens* zusätzlich die *Strg*-Taste (Cursor bekommt ein *Plus* ) und lassen Sie die Ergebnisgröße über dem Zielergebnisfenster *fallen, so* wird diese *kopiert*. Die kopierte Ergebnisgröße wird im Zielfenster mit den dort bereits vorhandenen Ergebnissen gemeinsam dargestellt. Diese Funktion ist auch über das Menü ***Bearbeiten/Kopieren*** und ***Bearbeiten/Einfügen als Kopie*** möglich.

### 7.3.4.2 Verschieben und Kopieren von Panels

Es bestehen 3 Möglichkeiten den Ergebnisverlauf auf der aktuellen Seite des Zielergebnisfensters abzulegen:

- vor dem vorhandenen Panel
- im vorhandenen Panel

nach dem vorhandenen Panel

- **<u>Einfügen vor dem vorhandenen Panel</u>**

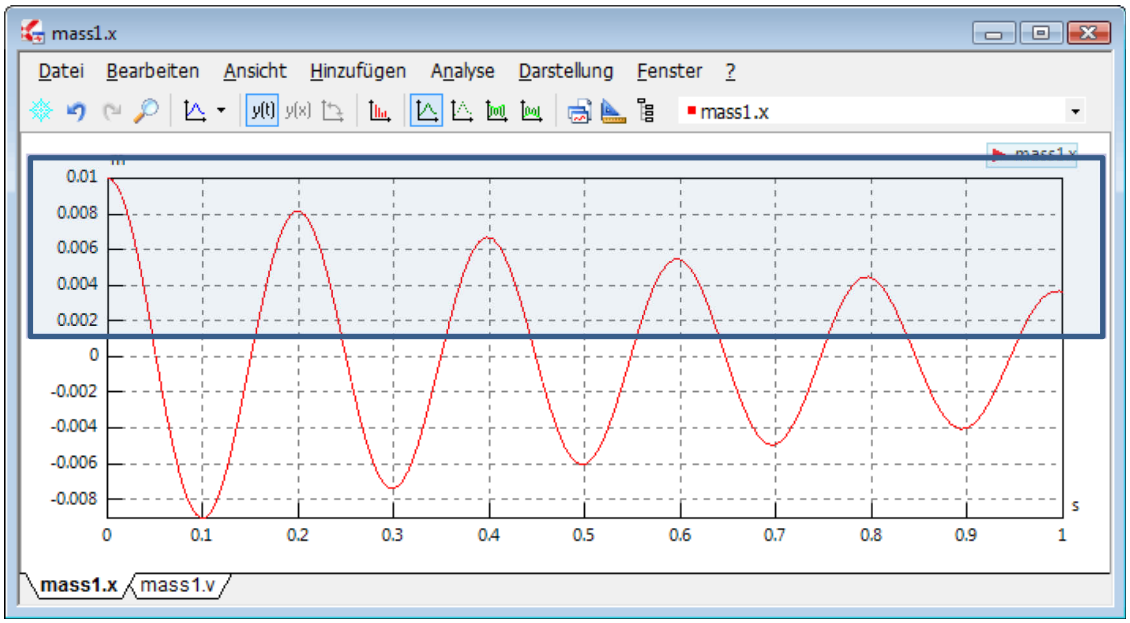



**Bild 7.15:** Ergebnisfenster mit Panel – oben

Ziehen Sie die Ergebnisgröße in den oberen Bereich (siehe **Bild 7.15**) eines Ergebnisdiagramms, so wird ein Panel oberhalb des vorhandenen eingefügt. Dann erhalten Sie eine Darstellung, wie in **Bild 7.25**.

- **Einfügen in das vorhandene Panel**

Wählen Sie den in **Bild 7.16** gezeigten Bereich, so wird die Kurve in das vorhandene Panel eingefügt, und es werden 2 Kurven in einem Diagramm angezeigt **(Bild 7.12)**.

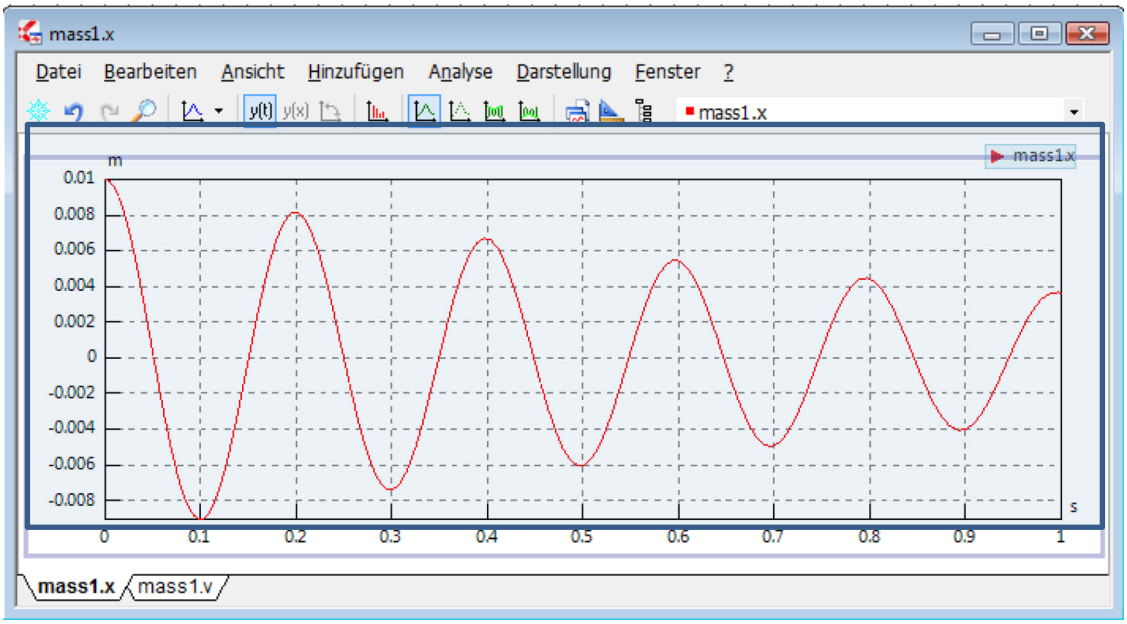



**Bild 7.16:** Ergebnisfenster mit Panel – mitte

- **Einfügen nach dem vorhandenen Panel**

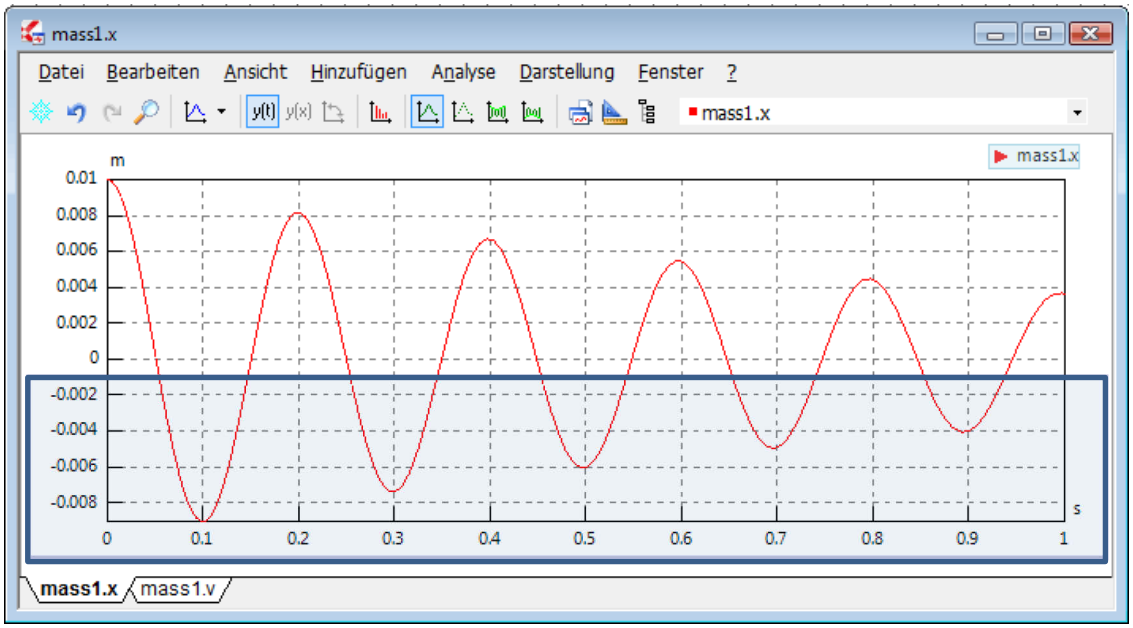



**Bild 7.17:** Ergebnisfenster mit Panel – unten

Wenn Sie die Kurve in den unteren Bereich (**Bild 7.17**) verschieben, wird ein neues Panel unterhalb des vorhandenen  angelegt und dort die Kurve eingefügt.

Sie können auch Ergebnisverläufe innerhalb eines Ergebnisfensters verschieben.

Mittels Drag & Drop können auch Panels und Serien in horizontaler Richtung verschoben werden. Diese Funktion ist jedoch nur möglich, wenn noch keine vertikale Anordnung vorhanden ist. Danach ist eine weitere Aufteilung nur über den Eigenschaftsdialog der Seite (siehe **7.3.8.1**) oder durch das Popup-Menü eines Panels möglich.

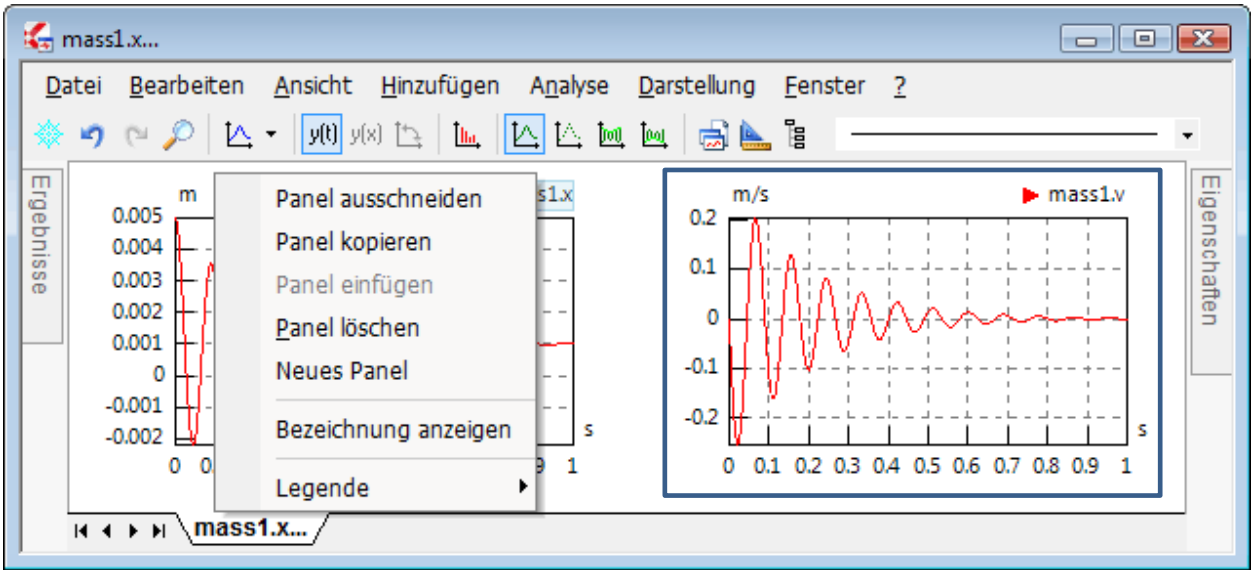



**Bild 7.18:** Drag & Drop eines Panels in horizontaler Richtung

### 7.3.4.3 Verschieben und Kopieren von Seiten

Mittels Drag & Drop können Seiten innerhalb eines Ergebnisfensters oder in ein anderes Ergebnisfenster verschoben werden.

Bitte beachten Sie, dass diese Drag & Drop Funktion durch eine Zeitbedingung geschützt ist. Sie müssen nach Drücken der linken Maustaste daher noch etwa eine halbe Sekunde warten, bevor Sie die Seite in Richtung des gewünschten Ziels weiterziehen.

Wird eine Seite in den Darstellungsbereich verschoben, öffnet sich ein neues Ergebnisfenster.

Drücken Sie die ***Ctrl***-Taste währen des Verschiebens wird die Seite kopiert.

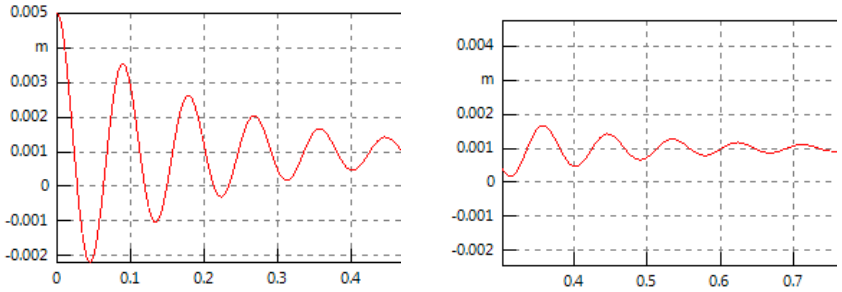
## 7.3.5 Anpassen der Ansicht

Das Aussehen und die Skalierung eines Ergebnisdiagramms sind an die Anforderungen des Nutzers anpassbar. Es können Abschnitte vergrößert betrachtet und Bezugslinien eingefügt werden. Eine individuelle Beschriftung der Achsen ist möglich.

### 7.3.5.1 Achsenanpassung und Skalierung

Skalierung mit der mittleren Maustaste :

*Maustaste* — Verschiebt den Ergebnisverlauf innerhalb des Panels (aktuelles Panel)

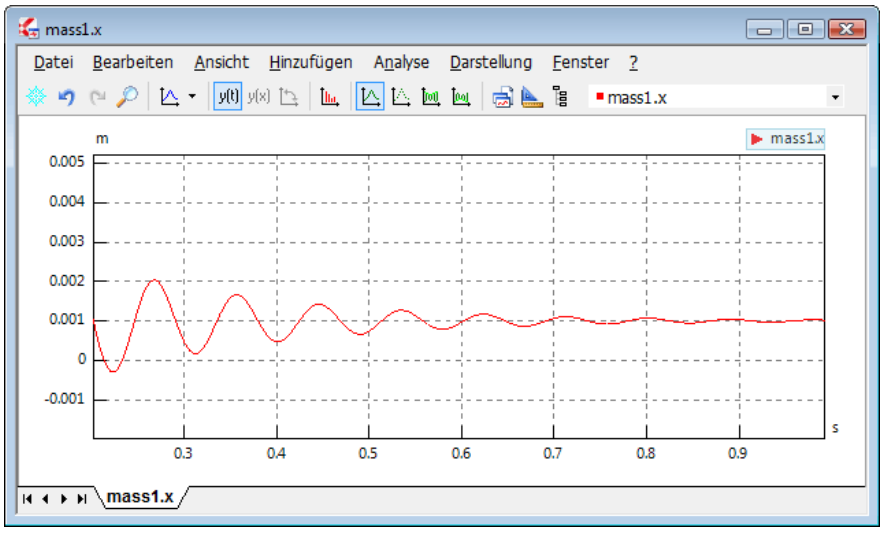



vor dem Verschieben — nach dem Verschieben

***Strg***+*Maustaste* — Skaliert die X- bzw. Y-Achse

Dazu wird an einer Achse (X oder Y) der Mauszeiger positioniert und anschließend *Strg*+*Maustaste* gedrückt.

Jetzt kann die Achse durch eine Bewegung des Mauszeigers entlang der Achse skaliert werden.




***Umsch***+*Maustaste* — Synchronisierung der X-Achse aller Panels (aktuelle Seite) beim Verschieben

***Strg+Umsch***+ *Maustaste* — Synchronisierung der X-Achse aller Panels (aktuelle Seite) beim Skalieren

- **Darstellung zoomen**

Alternativ zur Ausschnittbildung über Minimum- und Maximumeingabe im Eigenschaftsdialog kann ein Ausschnitt interaktiv mittels Lupenfunktion erzeugt werden.

Mit dem Menübefehl *Zoom* oder der Schaltfläche wird in die Lupenfunktion gewechselt. Anschließend wird über dem interessierenden Bereich mit der Maus ein Rechteck aufgezogen (siehe **Bild 7.19**).

Wenn Sie die Zoom-Funktion eines Ergebnisfensters nutzen, besteht die Möglichkeit mit den Kursortasten den Zoom-Bereich zu scrollen. In Ergebnisfenstern mit mehreren Diagrammen kann in jedem Diagramm separat gezoomt werden.

| | |
|---|---|
| ***Strg*** | Zoom-Modus bleibt eingeschaltet (mehrfacher Zoom möglich) |
| ***X**-Taste* | nur X-Achse wird verändert |
| ***Y**-Taste* | Nur Y-Achse wird verändert |

Ein Rücksetzen der Lupenfunktion erfolgt mit dem Befehl „Rückgängig machen“ . Die automatische Anpassung der Achsen wird dabei wieder aktiviert.

**Tipp:**
Für Dokumentationszwecke eines wertgenauen Ausschnittes ist die Eingabe von Minimum und Maximum im Eigenschaftsdialog zu bevorzugen (automatische Anpassung ausschalten!).

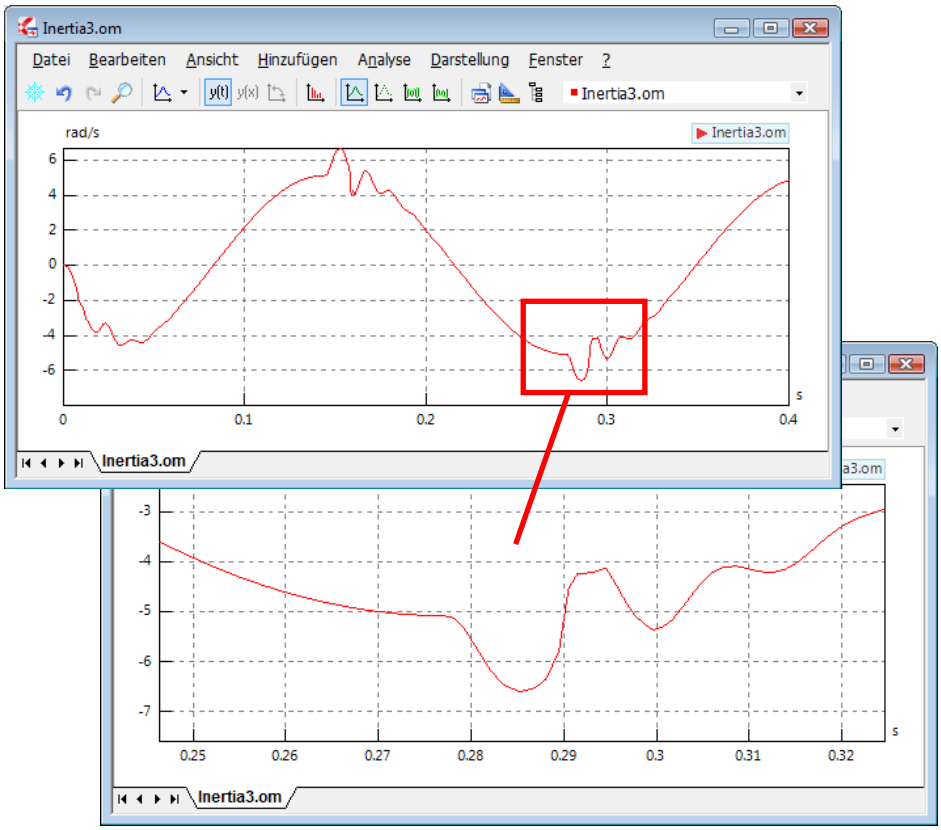
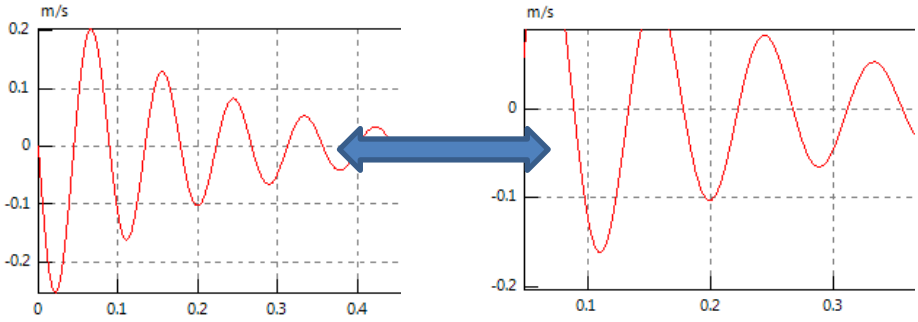



**Bild 7.19:** Ausschnittbildung mittels Lupenfunktion

Für das Zoomen mit dem *Mausrad* in mehreren Panels gibt es folgende zusätzliche Option:

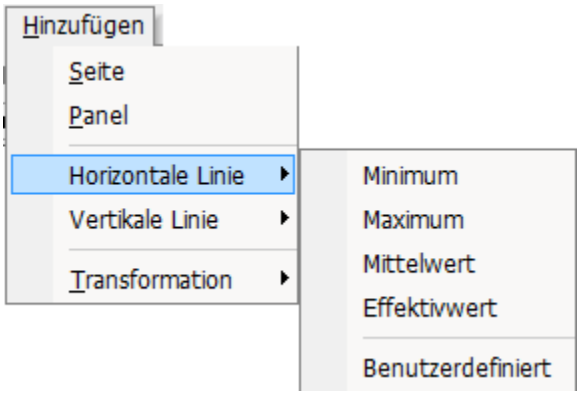
***Strg*+*Mausrad*** Stufenloser Zoom innerhalb eines Panels.




***Strg*+*Umsch*+*Mausrad*** Synchronisierung der X-Achse aller Panels beim Zoomen

### 7.3.5.2 Hilfslinien

Für die Bewertung der Ergebnisse ist es häufig nützlich, Bezüge zu einem bestimmten Wert (z. B. Minimum, Maximum, Mittelwert...) herzustellen.




Über ***Hinzufügen/Horizontale Linie*** oder ***/Vertikale Linie*** können entsprechende Werte im Ergebnisfenster eingeblendet werden. SimulationX bietet eine Reihe von Werten an.
Darüber hinaus ist über *Benutzerdefiniert* auch die Vorgabe eigener Werte vorgesehen.

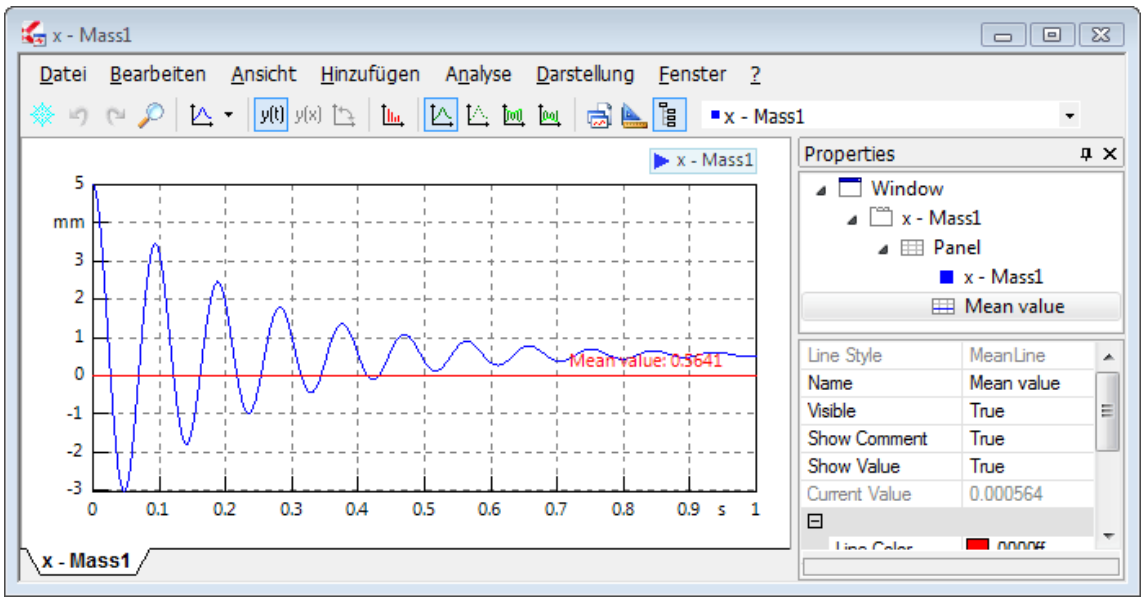
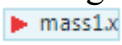



**Bild 7.20:** Ergebnisfenster mit Mittelwert (Hilfslinie)

Farbe, Linienart und Linienstärke einer Hilfslinie können durch Anklicken mit der Maus in der Eigenschaftenleiste angepasst werden.

Mittels Klick mit der rechten Maustaste auf eine Hilfslinie im Eigenschaftendialog der Ergebnisgröße kann diese gelöscht werden.

#### 7.3.5.3 Ändern von Beschriftungen

SimulationX erzeugt die Beschriftungen automatisch. Die Legendenbezeichnung (Bezeichner 

) einer Kurve wird von SimulationX aus der Bezeichnung der Ergebnisgröße und dem Namen des Elementes erzeugt, ist aber änderbar.

Das Attribut Titelformat dient der individuellen Gestaltung der Legendenbezeichnung einer ausgewählten Ergebniskurve. Die Beschriftung ändert den Namen der Variablen nicht und ist für eine verständlichere Darstellung der Ergebnisse vorgesehen.

| | |
|---|---|
| <nameX>, <nameY> | Name der Ergebnisgröße |
| <commentX>, <commentY> | Kommentar der Ergebnisgröße |
| <numX>, <numY> | Subscription für Verläufe aus mehrdimensionalen Ergebnisgrößen |
| <identX>, <identY> | Vollständiger Identifikator der Ergebnisgröße relativ zum Modell |
| <nameEnvX>, <nameEnvY> | Name des Modellobjekts, in der die Ergebnisgröße enthalten ist |
| <commentEnvX>, <commentEnvY> | Kommentar des Modellobjekts, in der die Ergebnisgröße enthalten ist |
| <nameRootX>, <nameRootX> | Name des zugehörigen Modells |
| <commentRootX>, <commentRootX> | Kommentar des zugehörigen Modells |

Das Dialogfeld ***Zahlenformat*** wird zur Zahlenausgabe (z.B. zur Achsenbeschriftung, Dateiausgabe, usw.) verwendet.

Die Formatierung hat die Form:

**%[*flags*][*width*][*.precision*]l*type***

Optionale Parameter sind in eckige Klammern gesetzt.

Die Parameter haben folgende Bedeutung:

| flag | Bedeutung | Standardeinstellung |
|---|---|---|
| - | veranlasst eine linksbündige Ausrichtung der Zahl | Ausrichtung rechtsbündig |
| + | die Zahl wird immer mit Vorzeichen ausgegeben | Vorzeichenausgabe nur bei negativen Zahlen |
| Leerzeichen | positiven Zahlen wird eine Leerzeichen vorangestellt | kein Leerzeichen bei positiven Zahlen |
| 0 | bis zur definierten Anzahl Zeichen (width) wird mit Nullen aufgefüllt; links oder rechts, entsprechend der Ausrichtung | |
| # | bei type = e, E oder f enthält die Ausgabe immer einen Dezimalpunkt | Dezimalpunkt erscheint nur, wenn Ziffern folgen |
| # | bei type = g oder G werden Nullen am Schluss nicht unterdrückt | Nullen am Schluss werden abgeschnitten |

**width** definiert die minimale Anzahl Zeichen, welche ausgegeben werden soll. Der Punkt trennt die Feldbreite (width) von dem Zahlenwert für die Genauigkeit (precision).

| Auswahl für type | Bedeutung des Wertes für precision | Standardeinstellung |
|---|---|---|
| f | Die Genauigkeit definiert die Anzahl Ziffern nach dem Dezimalpunkt. Wenn es einen Dezimalpunkt gibt, steht mindestens eine Ziffer links davon. Die Zahl wird auf die gewünschte Anzahl Ziffern gerundet. | 6 |
| e, E | Die Genauigkeit definiert die Anzahl Ziffern nach dem Dezimalpunkt. Die letzte Ziffer wird gerundet. | 6 |
| g, G | Die maximale Anzahl signifikanter Ziffern wird festgelegt. | 6 |

Das Zeichen **l** (Prefix für den type-Spezifikator) ist notwendig, da es sich hier um Zahlen vom Typ 'double' handelt.

| type | Bedeutung | Standardeinstellung |
|---|---|---|
| f | float; Gleitkommazahl, die in der Form [-]mmm.ddd dargestellt wird, wobei *precision* die Anzahl der d festlegt. Ist *precision* = 0, entfällt der Dezimalpunkt. | |
| e, E | float; Gleitkommazahl, die in der Form [-]m.dddddde±xxx oder [-]m.ddddddE±xxx dargestellt wird, wobei *precision* die Anzahl der d festlegt. Ist *precision* = 0, entfällt der Dezimalpunkt. | |
| g, G | float; %e oder %E wird benutzt,wenn der Exponent kleiner als -4 oder nicht kleiner als die Genauigkeit (*precision*) ist, sonst wird %f verwendet. Null und Dezimalpunkt am Schluß werden nicht dargestellt. | |

Die **Standardeinstellung** für das Zahlenformat ist **%6lg.**

## 7.3.6 Post-Processing

### 7.3.6.1 Transformationen

Eine Transformation ist die nachträgliche, von der Simulationsrechnung unabhängige Modifizierung von Berechnungsergebnissen innerhalb der Ergebnisfenster. Um auf den aktiven Verlauf innerhalb eines Ergebnisfensters eine Transformation anzuwenden, gehen sie wie folgt vor:

1. Markieren Sie den entsprechenden Verlauf in der Legende, der Auswahlbox in der Symbolleiste oder in der Eigenschaftsleiste.
2. Wählen Sie über den Menüpunkt ***Einfügen → Transformation*** die gewünschte Operation.
3. Bearbeiten Sie bei Bedarf die Parameter der hinzugefügten Transformation in der Eigenschaftsleiste.

### 7.3.6.2 Verknüpfungen

Operationen sind spezielle Transformationen, bei denen der Ausgangsverlauf einer Variablen mit anderen Variablen verknüpft wird.
Um zwei Verläufe miteinander zu verknüpfen, führen Sie folgende Aktionen aus:

1. Ziehen Sie den Verlauf, den Sie mit einem anderen verknüpfen wollen, per Drag & Drop auf das Zielpanel.
2. Legen Sie den Verlauf auf einem der dargestellten Transformationsfelder ab. Die hinzuzufügende Transformation bezieht sich auf den aktuellen Verlauf im Panel.

3. Bearbeiten Sie bei Bedarf die Parameter der Transformation in der Eigenschaftsleiste.

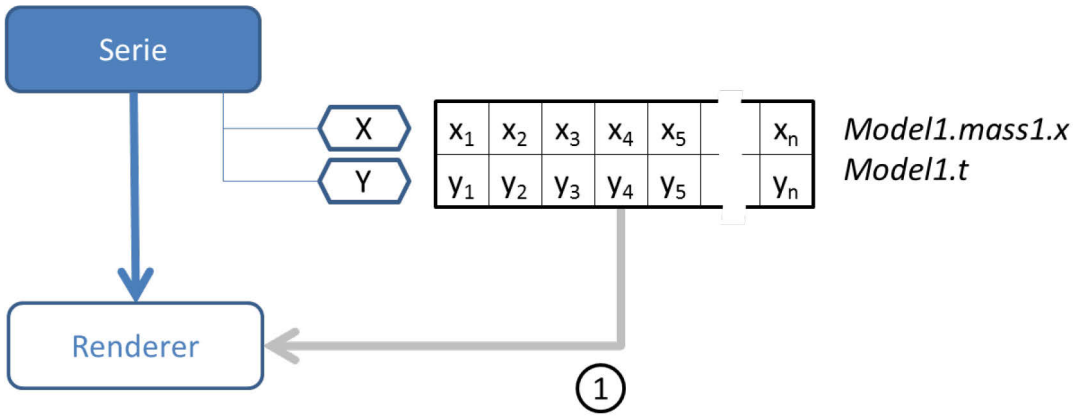
Nach der Übernahme der entsprechenden Variablenreferenzen in den Zielverlauf wird die Ursprungskurve gelöscht. Um dies zu verhindern kann während der Drag & Drop-Operation die ***Strg-Taste*** gedrückt gehalten werden.
Folgende Verknüpfungen können auf diese Art erstellt (Punkt 1-4) bzw. erweitert (Punkt 5) werden:

1. Summe zweier oder mehrerer Verläufe
2. Differenz zweier oder mehrere Verläufe
3. Produkt zweier oder mehrere Verläufe
4. Durchschnitt / Mittelwert zweier oder mehrerer Verläufe
5. Hinzufügen eines weiteren Arguments zu einer bereits bestehenden Verknüpfung

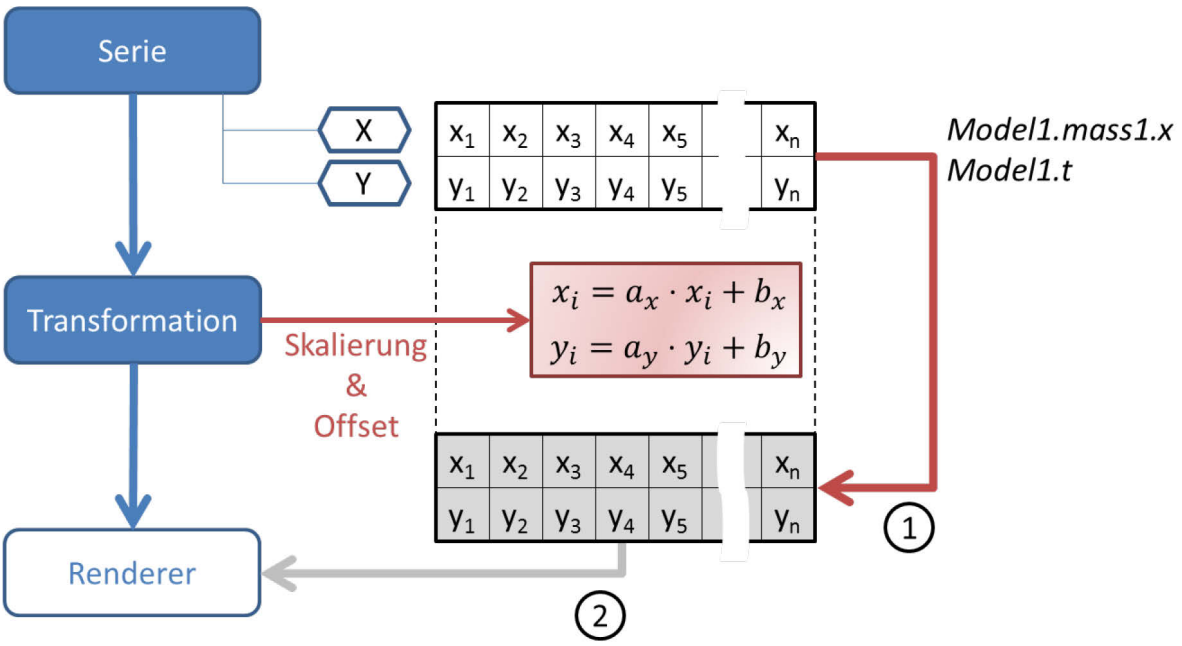
Die Art der Verknüpfung sowie die Argumente, auf die die Verknüpfung angewandt werden soll, können im Nachhinein in der Eigenschaftsleiste angepasst werden.

### 7.3.6.3 Datenmodell und interne Struktur

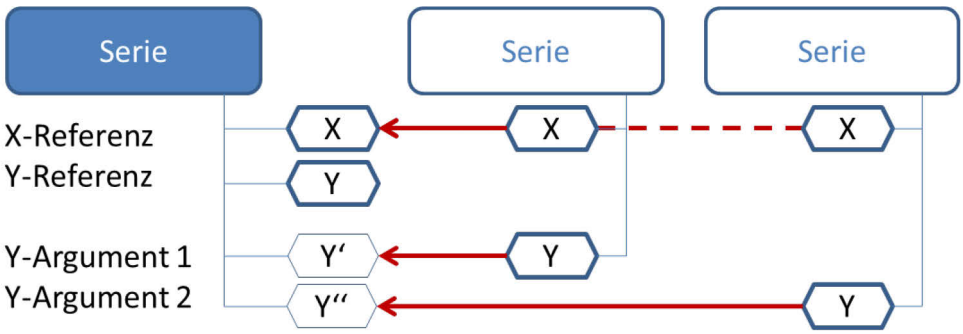
Ein im Ergebnisfenster dargestellter Verlauf bezieht seine Daten aus den in den Ergebnisvariablen über der Simulationszeit protokollierten Werten. Dazu werden diese punktweise, d.h. als Wertepaare von X und Y in einen Darstellungspuffer kopiert und von dort zur grafischen Anzeige gebracht. Die Verwaltung dieser Zwischenspeicherung obliegt den Serie-Objekten, die neben den Referenzen auf die entsprechenden Variablen auch die Darstellungseigenschaften speichern.



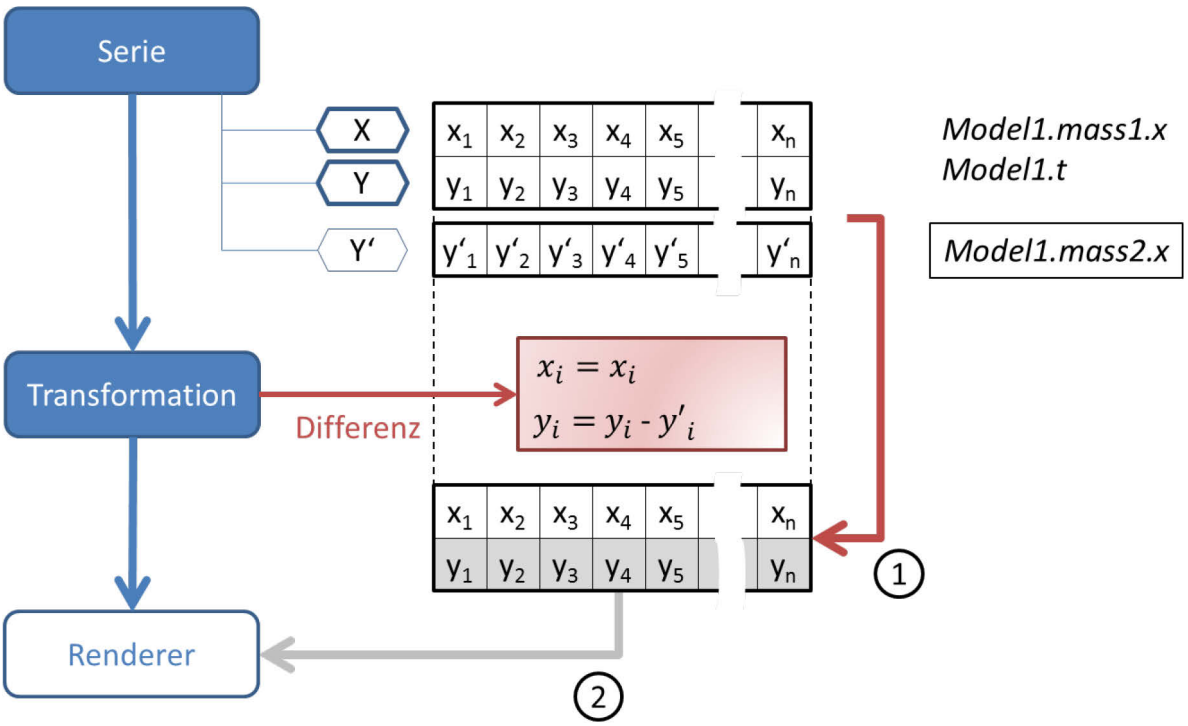

Auf ihrem Weg in den Darstellungspuffer können die Daten durch die Anwendung einer oder mehrerer Transformationen gezielt verändert oder auch mit den aufgezeichneten Werten weiterer Variablen verknüpfen werden.



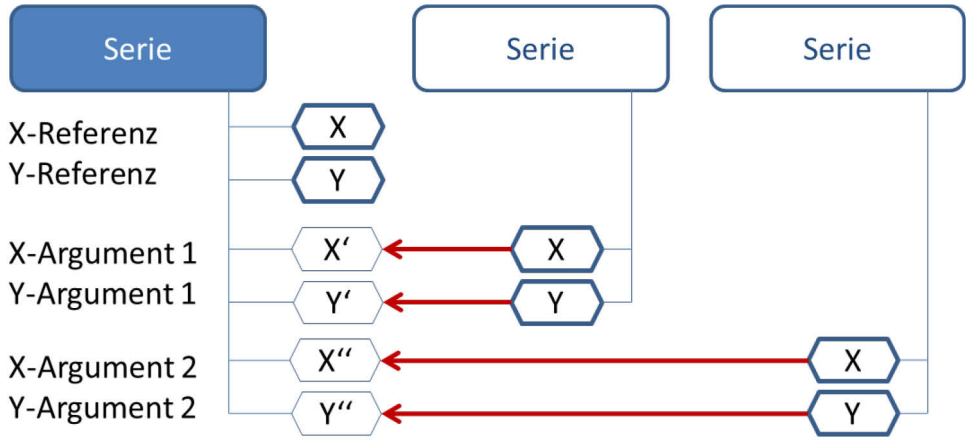

Für Verknüpfungen können im Serie-Objekt weitere Variablenreferenzen als potentielle Argumente für Transformationen und Verknüpfungen gespeichert werden. Beziehen sich die zu verknüpfenden Variablen auf eine gemeinsame Basis (z.B. die Simulationszeit t ein und desselben Modells) werden nur die Y-Referenzen der zusätzlichen Variablen gespeichert und die Berechnung der Verknüpfung wird recht einfach.



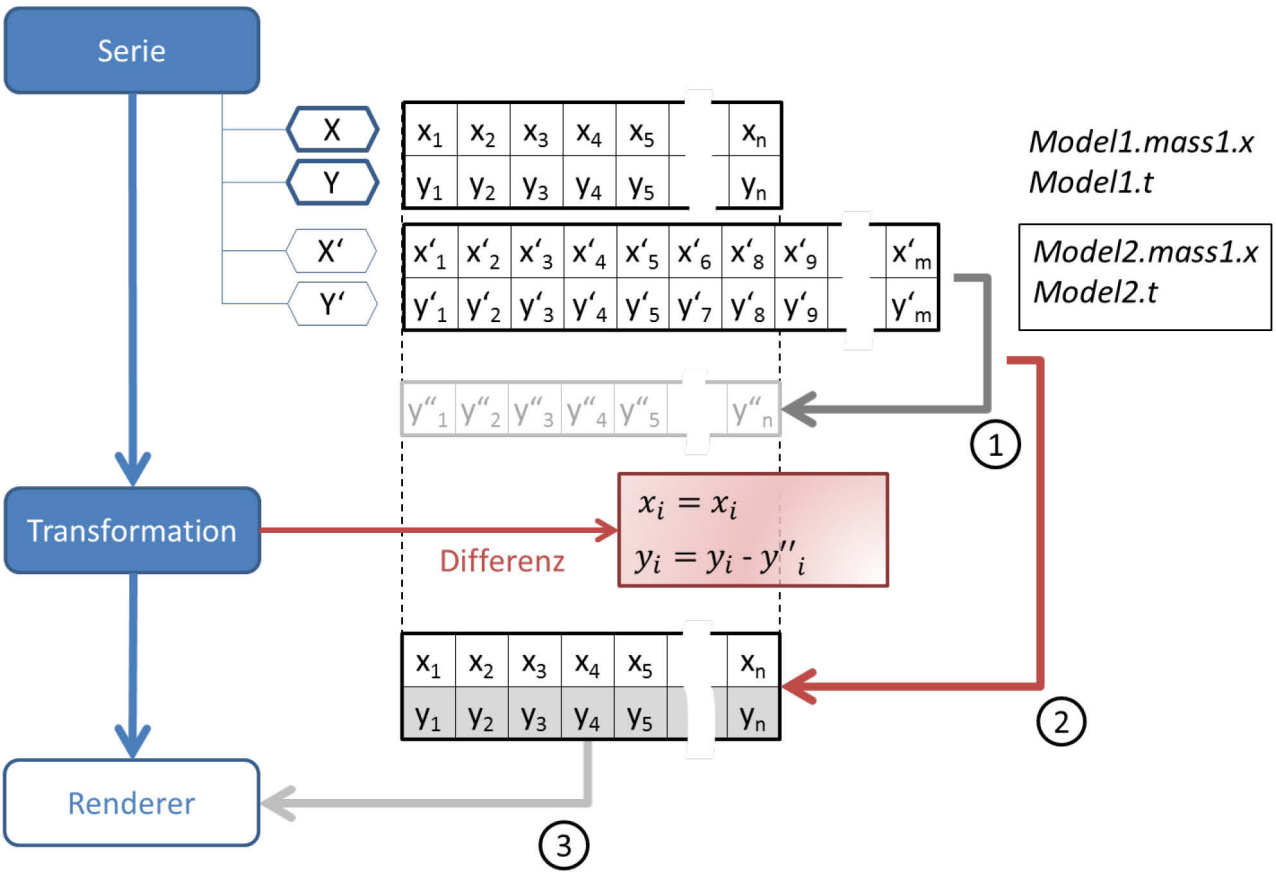

So werden zur Bildung der Differenz zweier Variablen für jedes Wertepaar die Y-Werte beider Verläufe voneinander abgezogen.




Kommen die zu verknüpfenden Variablen aus verschiedenen Modellen oder Simulationsrechnungen (z.B. durch Import oder Einfrieren von Berechnungsergebnissen), müssen sowohl die X- als auch die Y-Referenzen gespeichert werden.




Vor der eigentlichen Transformation muss zunächst eine interne Anpassung der Wertepaare vorgenommen werden. Dazu werden durch lineare Interpolation für den hinzugefügten Verlauf die Funktionswerte zu den X-Koordinaten des Ursprungsverlaufs bestimmt. Die Anzahl und Lage der Stützstellen der Zielfunktion werden damit immer durch den Ausgangsverlauf bestimmt.




Da ein Serie-Objekt mehrere zusätzliche Variablenreferenzen verwalten kann, ist es bei Verknüpfungen notwendig anzugeben, mit welcher der Argumentvariablen die gewünschte Operation durchgeführt werden soll. Als erster Operand wird immer die Variable des Ursprungsverlaufs verwendet.

### 7.3.6.4 Verkettungen von Transformationen und Operationen

Um mehrere Operationen hintereinander auszuführen, können Transformationen beliebig verkettet werden (siehe **Bild 7.21**).

Bei der Verkettung von Verknüpfungen (Operationen mit mehreren Operanden) ist zu beachten, dass nur der erste Operand Ergebnis einer vorherigen Transformation oder Verknüpfung sein kann. Alle übrigen Operanden verweisen direkt auf die jeweiligen Variablen.

Bemerkung:
In einer späteren Version von SimulationX wird es möglich sein, Transformationen und Verknüpfungen über Modelica-Ausdrücke zu beschreiben. Damit wird die genannte Einschränkung entfallen.

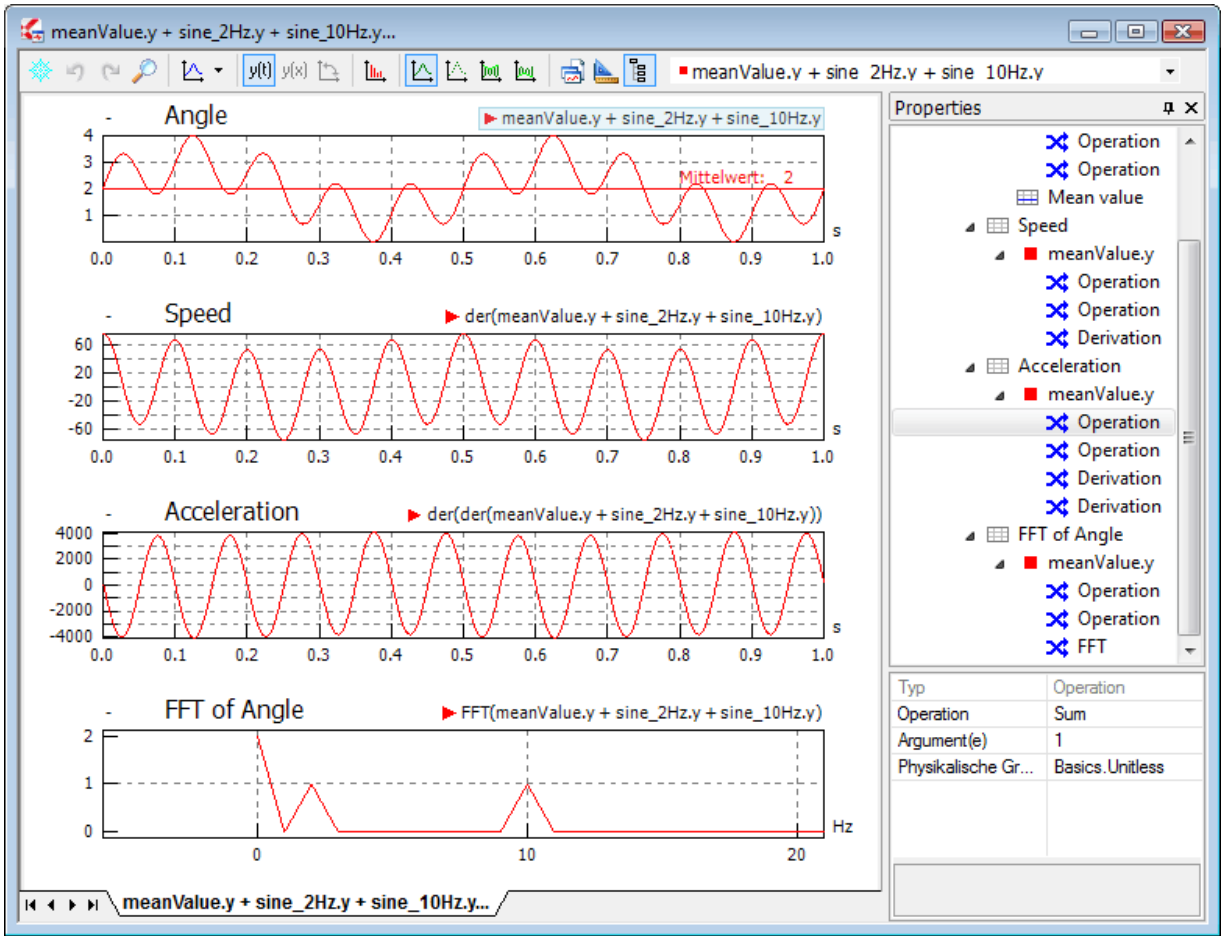



**Bild 7.21:** Verkette Transformationen

### 7.3.6.5 Ausmessen von Verläufen

- **Wertanzeige (Kurve ausmessen)**

Für die Darstellungsformen y(t) und y(x) können die Werte der Funktion an einem beliebigen Ort der aktiven Kurve des aktuellen Panels abgegriffen und angezeigt werden. Die Werteanzeige gilt nur für die aktive Kurve.

So messen Sie eine Kurve aus:

1. Bewegen Sie den Mauszeiger in den Bereich der grafischen Darstellung und drücken die  Maustaste. Der Cursor wandelt sich in ein Fadenkreuz und markiert den aktuellen Punkt. In der Statusleiste werden die Werte für X und Y für die **aktive** Kurve angezeigt.
2. Bewegen Sie den Cursor mit gedrückter Maustaste entlang der Kurve werden die Werte aktualisiert. Es werden nur **errechnete Punkte** angezeigt. Bei einer geringen Werteanzahl **springt** das Fadenkreuz von Punkt zu Punkt. Beim Loslassen der Maustaste verschwindet das Fadenkreuz.

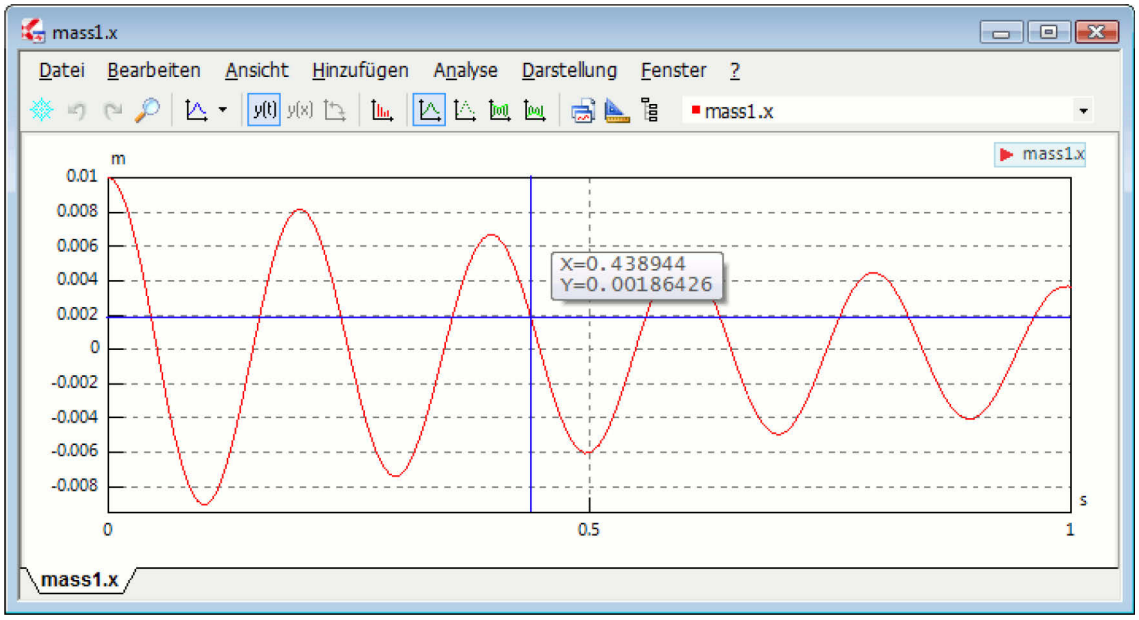



**Bild 7.22:** Ergebnisfenster mit Wertanzeige

Mittels der Option Messwertleiste können Sie für die Werteanzeige einen gesonderten Bereich zuschalten, wobei das Fadenkreuz und die Messwerte erhalten bleiben, wenn die Maustaste losgelassen wird. Durch Anfassen des Fadenkreuzes mit der Maus und Bewegen entlang der Kurve können weitere Messpunkte ermittelt werden. Im Anzeigebereich werden die Werte für X und Y ausgegeben.

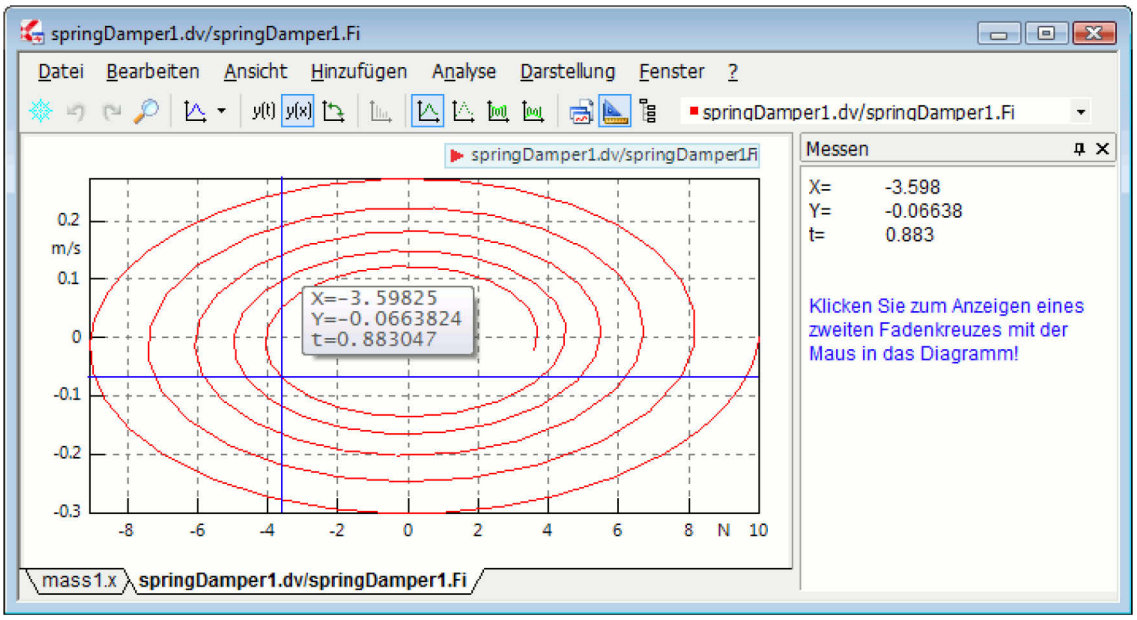



**Bild 7.23:** Ergebniskurve mit Messwert

Durch erneutes Klicken mit der Maus über dem Kurvenverlauf erhalten Sie ein zweites Fadenkreuz.

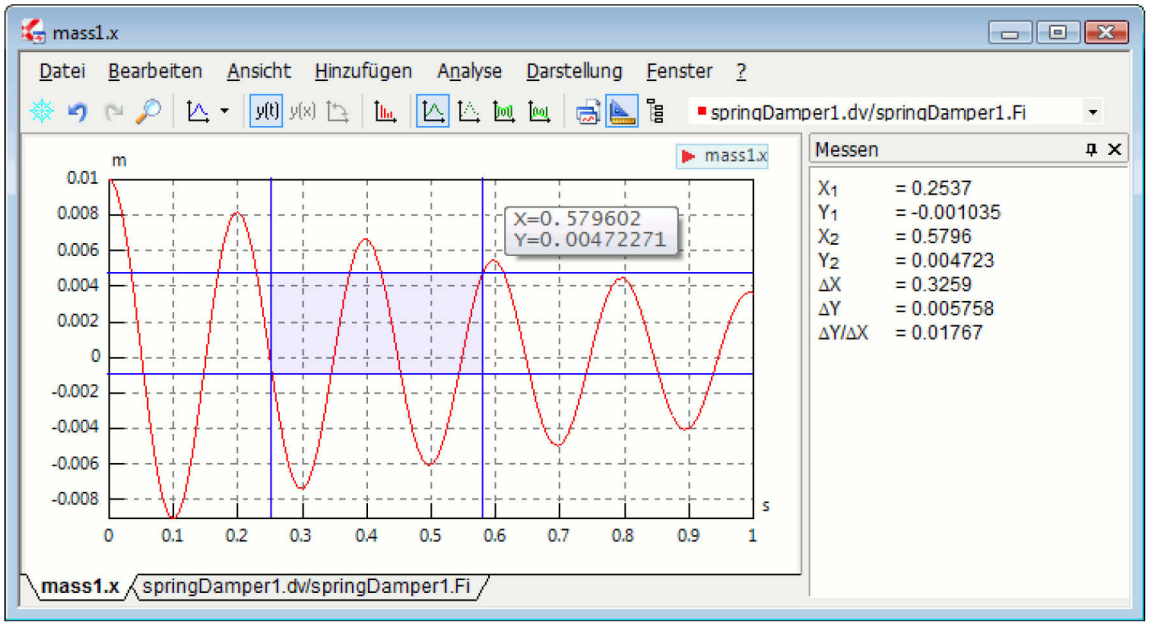



**Bild 7.24:** Ergebniskurve mit 2 Messwerten

Neben den Werten der beiden Messpunkte ($X_1$, $X_2$, $Y_1$, $Y_2$) erhalten Sie Angaben zu den Wegdifferenzen ($\Delta X$, $\Delta Y$) und dem Anstieg ($\Delta X/\Delta Y$) zwischen den Punkten (siehe **Bild 7.24**).

- **<u>Ausmessen mit mehreren Ergebnisverläufen</u>**

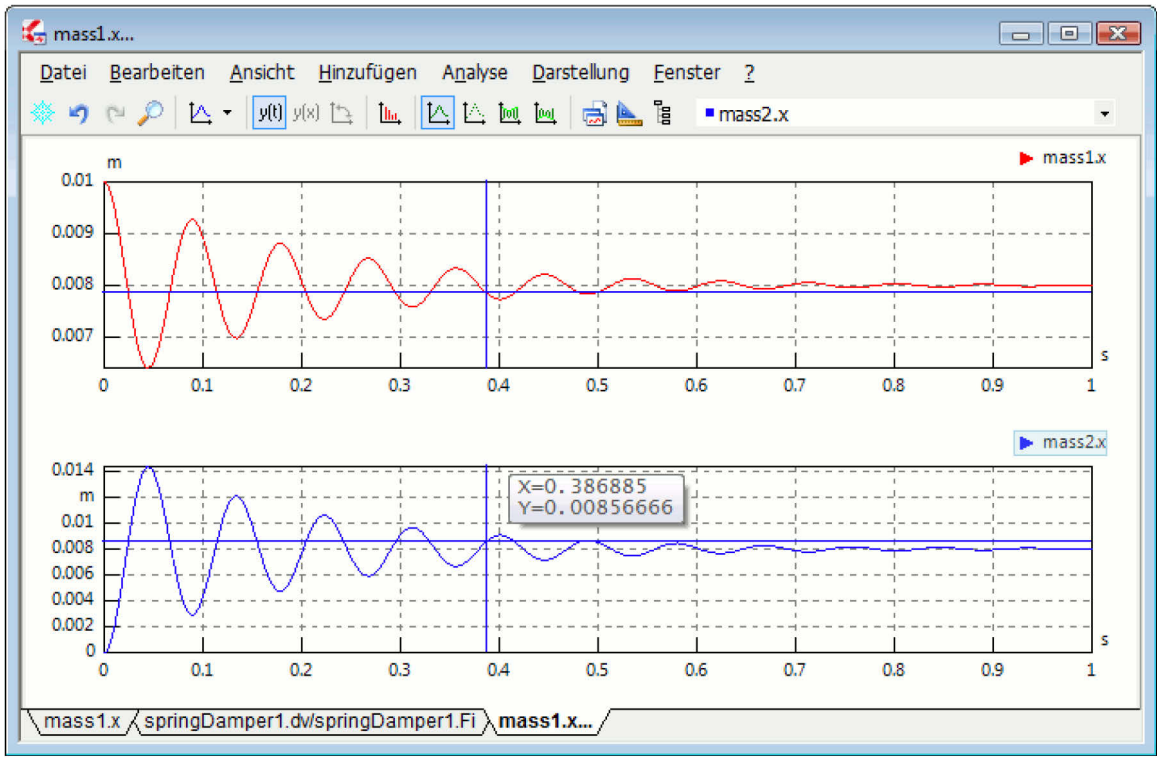



**Bild 7.25:** Messwertanzeige mit 2 Diagrammen

Bei der Darstellung von mehreren Ergebnisdiagrammen in einem Ergebnisfenster werden in allen Diagrammen die Kurven abgefahren und es erfolgt eine Werteanzeige für den aktiven Ergebnisverlauf.

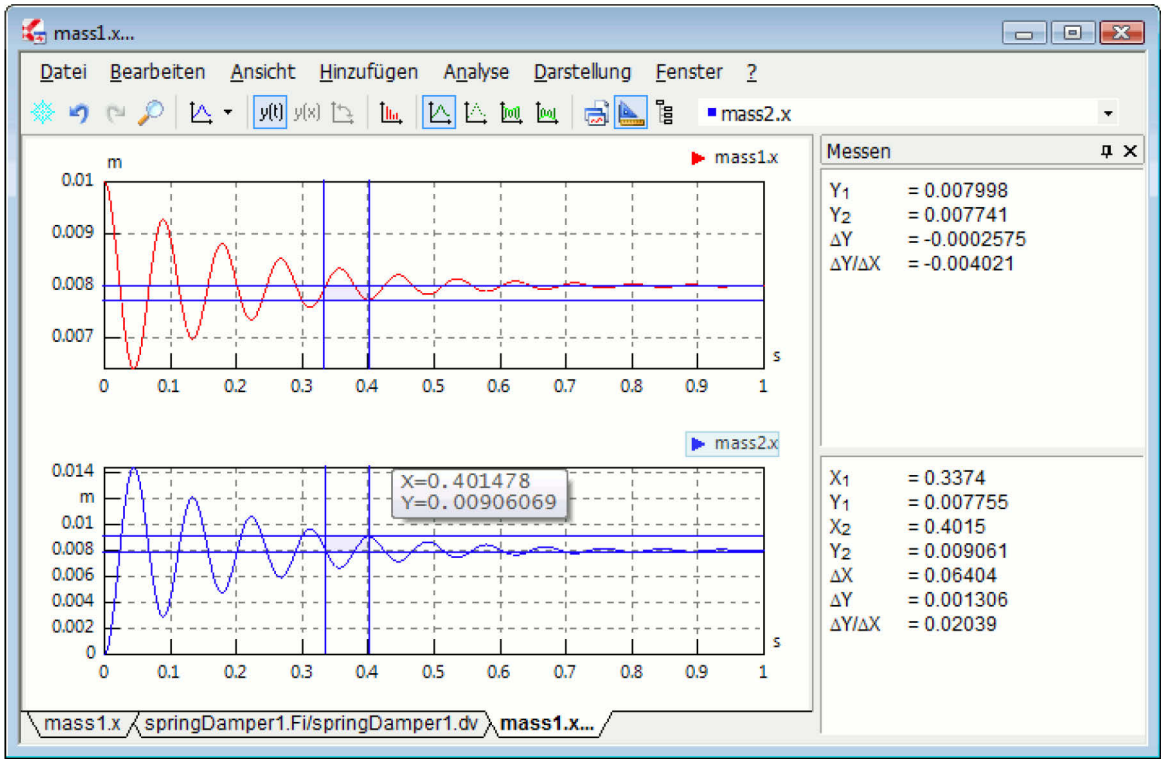



**Bild 7.26:** Messwertbereichs-Darstellung

Bei der Arbeit mit Messwertleisten werden für alle Kurven die Y-Werte ausgegeben. Für den aktiven Ergebnisverlauf werden auch die Werte der X-Achse aufgeführt (siehe Kurve Weg-mass2 in **Bild 7.26**)

## 7.3.7 Einfrieren und Import von Ergebnisverläufen

*Einfrieren* - dient dem Festhalten von einmal berechneten Ergebnisverläufen, um sie mit Ergebnissen nachfolgender Simulationen zu vergleichen. Dazu erzeugt SimulationX eine Kopie (schwach dargestellte Kurve) vom aktuellen Ergebnisverlauf, die beim Rücksetzen des zugehörigen Simulationsmodells erhalten bleibt. Eingefrorene Verläufe werden mit dem Modell gespeichert!

*Öffnen* - von gespeicherten (s.u.) Verläufen in ein Ergebnisfenster

*Speichern* - gestattet das Speichern aller im Ergebnisfenster dargestellter Verläufe auf einem Datenträger in einem der folgenden Formate (siehe auch **Kap. 8.1**):

- Text (*.txt)
- IEEE Binär (*.bin)
- ITI SimulationX (*.rfs)
- CSV(Trennzeichen getrennt) (*.csv)
- DIAdem-Header (*.dat)
- XML (*.xml)

- Grafik (*.bmp, *.jpeg, *.gif, *.tiff, *.png, *.emf)

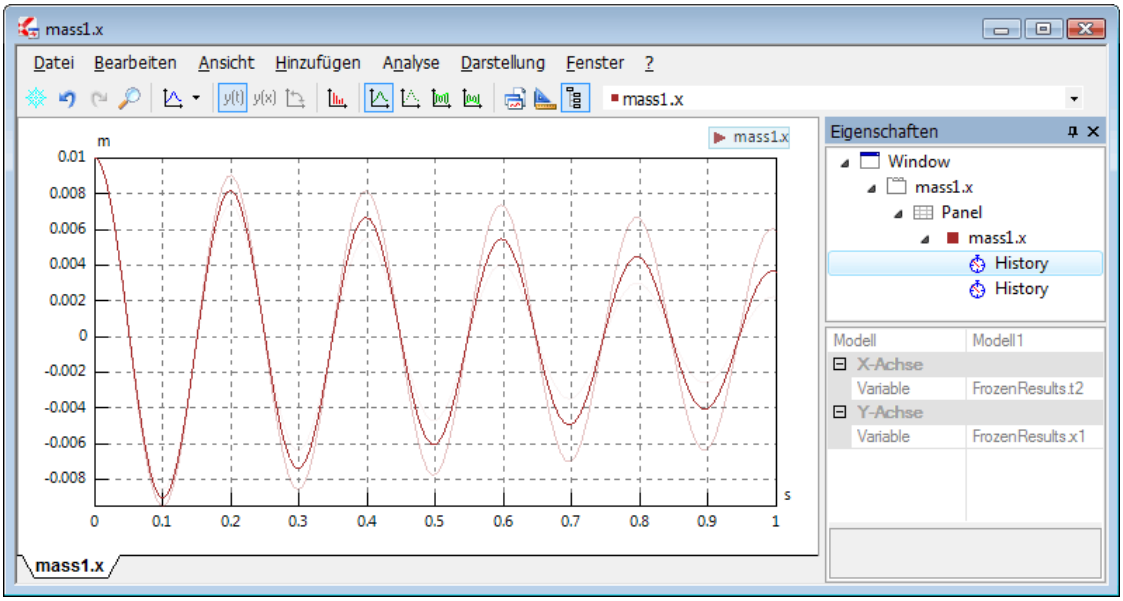



**Bild 7.27:** Ergebnisfenster mit eingefrorenem Verlauf x-Mass1.x (History)

## 7.3.8 Ergebnisse und Druck

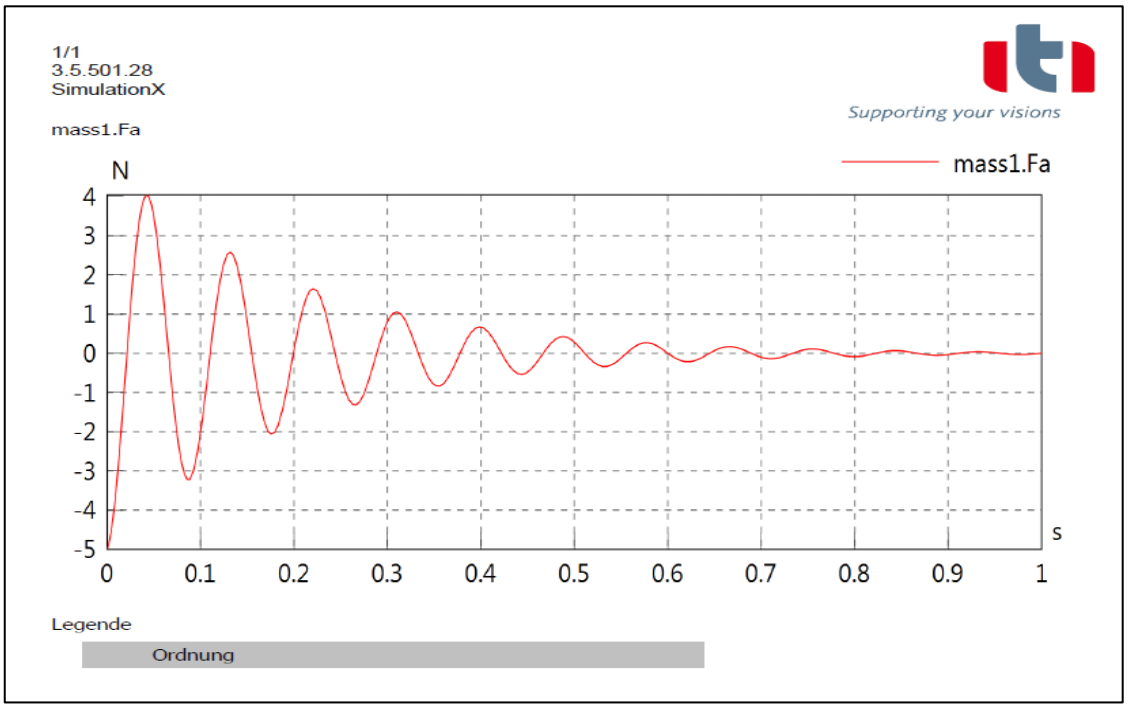



**Bild 7.28:** Druckausgabe eines Ergebnisverlaufes

Über das Menü Export können die Berechnungsergebnisse exportiert werden. Dabei werden alle Ergebnisse des aktuellen Modelles in folgenden Formaten gespeichert:

- Text (*.txt)
- CSV(Trennzeichen getrennt) (*.csv)

- XML (*.xml)

Ein Druck der Ergebnisverläufe ist möglich. Dabei wird die aktuell ausgewählte Seite des Ergebnisfensters mit allen Ergebnisverläufen ausgegeben.

#### 7.3.8.1 Einstellungen - Die Eigenschaftsleiste

Über das Menü ***Ansicht/Eigenschaftsleiste*** gelangen Sie zum Eigenschaftendialog für Ergebnisfenster.

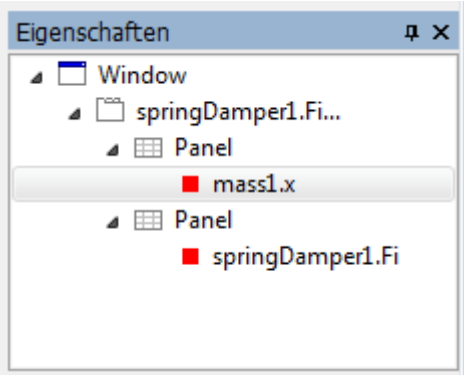



**Bild 7.29:** Eigenschaften – Auswahl der Kategorie

Über diesen Dialog sind alle Attribute der Darstellung im Ergebnisfenster einstellbar. Bei Auswahl einer entsprechenden Kategorie werden die möglichen Einstellungen im unteren Teil der Eigenschaftenleiste sichtbar. Im Weiteren werden die wichtigsten Einstellungen beschrieben.

Menüleiste, Symbolleisten und Statusleiste können mittels des Menüs ***Ansicht*** des Ergebnisfensters aus- bzw. eingeblendet werden. Mit diesen Optionen lässt sich Platz auf dem Desktop sparen (vorteilhaft, wenn viele Ergebnisfenster gleichzeitig gebraucht werden).

- **Eigenschaften des Fensters**

| **Fenster** | |
|---|---|
| Name / Erläuterung | Wert |
| **Achseinstellungen beim Einfrieren erhalten**<br>Die aktuellen Einstellungen für Minimum und Maximum bleiben beim Einfrieren erhalten. Die automatische Skalierung wird deaktiviert. | True/False |

| | |
|---|---|
| **Unabhängige Kurven für eingefrorene Ergebnisse**<br>Wählen Sie diese Einstellung, um beim Einfrieren von Verläufen eigenständige Kurven zu erhalten. Dies ermöglicht die Anwendung von Transformation oder das Verschieben in andere Fenster, Seiten oder Panels. Ist diese Option deaktiviert, werden eingefrorene Verläufe fest mit dem Ursprungsverlauf verknüpft. | True/False |
| **Hohe Darstellungsqualität**<br>Erhöht die Darstellungsqualität für die Bildschirmdarstellung der Ergebnisfenster. Die Darstellungsperformance wird dadurch vermindert. Beim Export von Verläufen als Grafik wird automatisch eine hohe Darstellungsqualität gewählt. | True/False |
| **Gepufferte Werte beim Deaktivieren erhalten**<br>Mit dieser Einstellung können Sie die Darstellungsgeschwindigkeit beim Umschalten zwischen den Seiten eines Ergebnisfensters auf Kosten des benötigten Speicherplatzes erhöhen. | True/False |
| **Anzeige von Info-Tipps**<br>Bei aktivierter Einstellung kann SimulationX für verschiedene Interaktionen (z.B. Zoom) zusätzliche Informationen über mögliche Optionen in Form eines Info-Tipps anzeigen. | True/False |
| **Ungültige Verläufe entfernen**<br>Deaktivieren Sie diese Option, um das automatische Löschen von Verläufen zu verhindern, deren zugehörigen Variablen aus dem entsprechenden Modell entfernt wurden. Die betreffenden Verläufe verbleiben im Ergebnisfenster und werden in der Legende entsprechend gekennzeichnet. Sobald im Modell wieder eine Variable mit dem gleichen Ident vorhanden ist, wird der betreffende Verlauf wiederhergestellt. | True/False |
| **Mit Modell sichtbar machen**<br>Mit dieser Einstellung legen Sie fest, ob das Ergebnisfenster beim Aktivieren des zugehörigen Modells automatisch angezeigt wird. Dazu muss im Ergebnisfenstermanager die entsprechende Option eingeschaltet sein. | True/False |

| **Schriftart** | |
|---|---|
| Name | Wert |
| Name | Auswahl Schriftart |
| Schriftgröße | Zahlenwert (integer) |
| Schriftfarbe | Auswahl Farbe |

| | | |
|---|---|---|
| | **Postition** | |
| | X | Zahlenwert (integer) |
| | Y | Zahlenwert (integer) |
| | **Größe** | |
| | Breite | Zahlenwert (integer) |
| | Höhe | Zahlenwert (integer) |

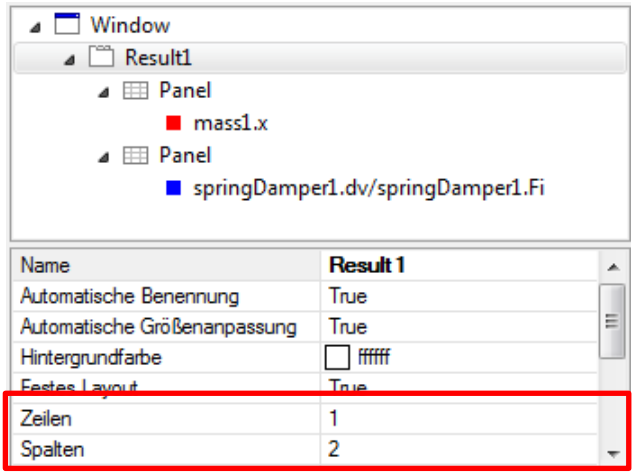
Die gewählte *Schriftart* bezieht sich auf alle Beschriftungen im Ergebnisfenster.

- **Eigenschaften der Seite**

| **Seite** | |
|---|---|
| **Name**<br>Zeigt den aktuellen Namen der Seite an. Mit dem Ändern des Namens wird die Option *Automatische Benennung* deaktiviert | Zeichenkette |
| **Automatische Benennung**<br>Schaltet die automatische Benennung der Seiten ein oder aus. Die automatische Benennung wird aus dem ersten Verlauf der Seite gebildet. | True/False |
| **Hintergrundfarbe**<br>Legt die Hintergrundfarbe für die Seite fest. | Auswahl Farbe |
| **Zeilen**<br>Zeigt die Anzahl der nebeneinander angeordneten Panels an, oder legt diese fest. | Zahlenwert |
| **Spalten**<br>Zeigt die Anzahl der untereinander angeordneten Panels an, oder legt diese fest. | Zahlenwert |
| **Automatische Größenanpassung**<br>Aktiviert die automatische Größenanpassung. Damit wird die Größe der auf der Seite vorhandenen Panels optimal an die jeweilige Fenstergröße angepasst.<br>Für den Druck oder Grafikexport ist es oftmals eine feste Größe der Darstellung zweckmäßig, die Sie mit dem Deaktivieren dieser Option erreichen. | True/False |

| | |
|---|---|
| **Seitenformat**<br>Legen Sie hier die gewünschte Seitengröße fest<br>(nur verfügbar, wenn *Automatische Größenanpassung* deaktiviert ist). | • ISO/DIN A4<br>• ISO/DIN B4<br>• ISO/DIN C4<br>• ISO/DIN D4<br>• Legal<br>• Letter<br>• User |
| **Orientierung**<br>Legen Sie hier die gewünschte Seitenorientierung fest (nur verfügbar, wenn *Automatische Größenanpassung* deaktiviert ist). | • Landscape<br>• Portrait |
| **Benutzerdefinierte Breite**<br>Legen Sie hier die nutzerdefinierte Breite der Seite in Millimetern fest (nur verfügbar bei *Automatische Größenanpassung* und Seitenformat *User*). | Zahlenwert (real) |
| **Benutzerdefinierte Höhe**<br>Legt die nutzerdefinierte Höhe der Seite in Millimetern fest. (Nur verfügbar bei *Automatische Größenanpassung* und Seitenformat *User*). | Zahlenwert (real) |
| **Seitenränder in mm** | |
| **Links**<br>Legt den linken Seitenrand in Millimetern fest (nur verfügbar bei *Automatische Größenanpassung*). | Zahlenwert (real) |
| **Rechts**<br>Legt den rechten Seitenrand in Millimetern fest (nur verfügbar bei *Automatische Größenanpassung*). | Zahlenwert (real) |
| **Oben**<br>Legt den oberen Seitenrand in Millimetern fest (nur verfügbar bei *Automatische Größenanpassung*). | Zahlenwert (real) |
| **Unten**<br>Legt den unteren Seitenrand in Millimetern fest (nur verfügbar bei *Automatische Größenanpassung*). | Zahlenwert (real) |

Über den Einstellungsdialog der Seite können Sie die Anzahl der Panels und deren Anordnung innerhalb einer Seite des Ergebnisfensters festlegen. Im Beispiel wurden 2 Panels in einer Zeile definiert.




Das Ergebnis ist im folgenden Bild sichtbar.

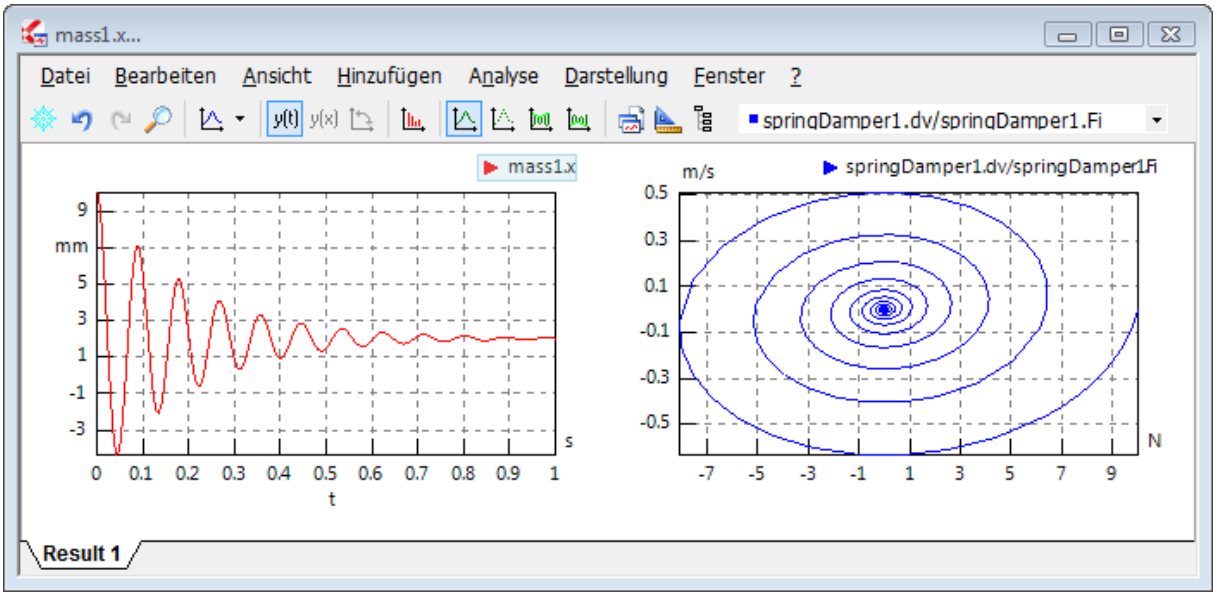



**Bild 7.30:** Ergebnisfenster mit einer Seite und 2 Panels in einer Zeile

- **Eigenschaften des Panels**

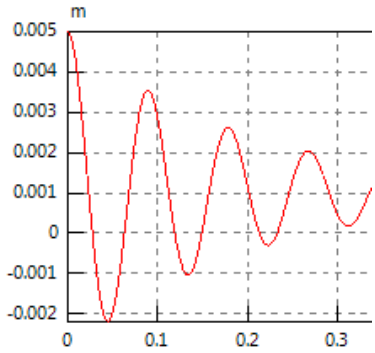
| **Panel** | |
|---|---|
| **Name**<br>Geben sie hier dem Panel eine (optionale) Bezeichnung. | Zeichenkette |
| **Namen anzeigen**<br>Wählen Sie diese Option, um den Namen des Panels links oberhalb des Koordinatensystems anzuzeigen. | True/False |

| | | |
|---|---|---|
| **Darstellungsart**<br>Wählen Sie hier die Darstellungsart des Panels. Gebräuchliche Einstellungen für Funktionsverläufe sind *Kartesische Koordinaten* und *Polarkoordinaten*. Die übrigen Einstellungsmöglichkeiten sind für die Visualisierung von Momentanwerten bestimmt und werden im Text weiter unter erläutert. | | • Kartesische Koordinaten<br>• Polarkoordinaten<br>• Balken |
| **Hintergrundfarbe**<br>Legt die Hintergrundfarbe für das Koordinatensystem fest. | | Farbauswahl |
| **Achsenfarbe**<br>Legt die Farbe der Achsen fest. | | Farbauswahl |
| **Legende**<br>Legt die Position der Legende fest. | | • Keine Legende<br>• Oben<br>• Rechts |
| | **Gitter** | |
| | **Linienfarbe**<br>Legt die Linienfarbe des Gitters fest. | Farbauswahl |
| | **Linienstärke**<br>Legt die Linienstärke des Gitters fest. | Zahlenwert (integer) |
| | **Linienart**<br>Legt die Linienart des Gitters fest. | • Durchgezogen<br>• Strich<br>• Punkt<br>• Strichpunkt<br>• Strichpunkt-Punkt |
| | **X-Achse / Y-Achse** | |
| | **Gitter**<br>Legt die Sichtbarkeit des Gitters fest. | True/False |
| | **Achsenbeschriftung**<br>Legt die Sichtbarkeit der Achsenbeschriftung fest | True/False |
| | **Gemeinsame Achse**<br>Legt fest, ob bei mehreren Verläufen im Panel eine gemeinsame Achse verwendet werden soll. | True/False |
| | **Automatische Anpassung**<br>Schaltet die automatische Skalierung ein bzw. aus. | True/False |

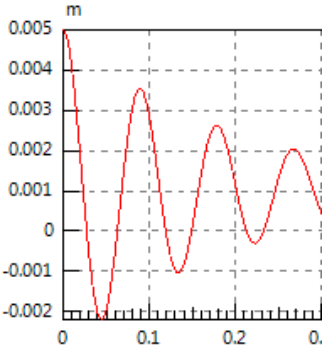
**Zusätzliche Unterteilung** — True/False

Sorgt für eine zusätzliche Unterteilung der Hauptintervalle der Achsenunterteilung.

False



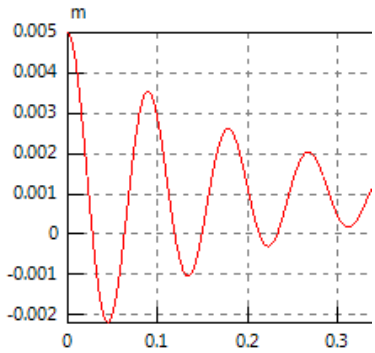

True



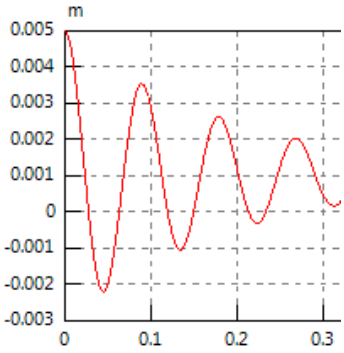

**Ganzzahlige Unterteilung** — True/False

Sorgt für eine ganzzahlige Unterteilung der Achsen. Minimum und Maximum fallen mit den Grenzen eines Hauptintervalls der Achsenunterteilung zusammen.

False



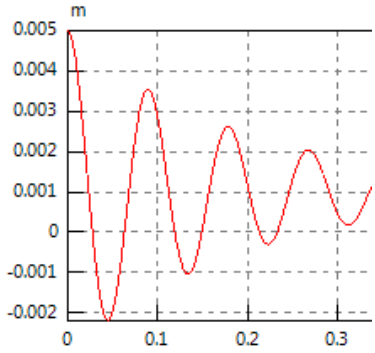

True



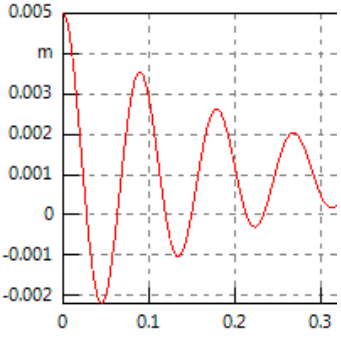

**Maßeinheit nach DIN 416** — True/False

Positioniert die Maßeinheit in der Mitte des letzten Hauptintervalls der Achsenunterteilung.

False




True



| | | |
|---|---|---|
| | **Zahlen anzeigen**<br>Zeigt die Zahlenwerte an den Hauptintervallgrenzen an. | True/False |

- **Eigenschaften der Achsen**

Um individuelle Einstellungen für jeden Ergebnisverlauf vornehmen zu können, ist das entsprechende Panel auszuwählen.

X- und Y-Achse haben die gleichen Eigenschaften. Alle Eigenschaften sind für beide Achsen getrennt einstellbar.
Enthält das Ergebnisfenster mehrere Kurven, kann für die jeweilige Koordinate (X, Y) eine *gemeinsame Achse* (true) dargestellt werden.

Ist *automatische Anpassung* gewählt (true), wird aus allen (gemeinsame Achse) oder aus der jeweiligen Kurve Minimum und Maximum berechnet und die Achse so dargestellt, dass alle Werte sichtbar sind und das Fenster optimal genutzt wird. Minimum, Maximum und Achsteilung sind dann **nicht mehr** wählbar.

Die Achsenbeschriftung ist wahlweise zu- oder abschaltbar (true). Sie bezieht sich auf die Skalenbeschriftung und auf die Maßeinheit.
Mit dem Parameter *Zusätzliche Unterteilung* lässt sich die Anzahl der Abschnitte (Raster) der Achse einstellen. Das Raster lässt sich ein- bzw. ausschalten.
Wahlweise kann jede Achse mit einer logarithmischen Teilung dargestellt werden.
Je nach allgemein gewählten Eigenschaften können im Dialogteil *Einstellungen* die Parameter *Minimum*, *Maximum* (z. B. zur Ausschnittbildung), *Maßeinheit* und *Achsenunterteilung* eingegeben werden.
Für die **X-Achse** kann die Achsrichtung über *Negative Orientierung* (true/false) gewählt werden.

- **Eigenschaften der Kurvendarstellung**

Jede einzelne Kurve kann mit unterschiedlichen Darstellungseigenschaften, wie Farbe, Linienart und -stärke sowie Markern, versehen werden. Diese Einstellungen sind durch Auswahl einer Ergebnisvaraiblen in der Eigenschaftenleiste (**Bild 7.28** ) vorzunehmen.
Zur Anpassung der Kurvendarstellung ist wie folgt zu verfahren:

1. Wählen Sie die anzupassende Kurve in der ***Eigenschaftenleiste***.
2. Wählen Sie die gewünschten Eigenschaften.

| **Serie** | | |
|---|---|---|
| **Bezeichnung**<br>Legt die Bezeichnung des Verlaufs fest. Für die automatische Generierung der Bezeichnungen können Sie verschieden Makros verwenden (siehe **7.3.5.3**). | | Zeichenkette |
| **Sichtbar**<br>Nutzen Sie diese Option zum (temporären) Ausblenden einer Kurve. | | True/False |
| **Automatisches Einfrieren beim Rücksetzen**<br>Ist diese Option eingeschaltet, wird beim Rücksetzen des zugehörigen Modells die betreffende Kurve automatisch eingefroren. | | True/False |
| **Transformation im Titel anzeigen**<br>Schließt die Transformation automatisch in den Title ein. So wird beispielsweise eine FFT von mass1.om als FFT(`mass1.om`) angezeigt. | | True/False |
| | Darstellung | |
| | **Linienfarbe**<br>Legt die Linienfarbe der Kurve fest. | Farbauswahl |
| | **Linienstärke**<br>Legt die Linienstärke der Kurve fest. Für die schnellste Darstellung auf dem Bildschirm sollte die Linienstärke 1 gewählt werden. Für den Druck bzw. den Grafikexport sorgt größere Linienstärke für eine bessere Lesbarkeit der Diagramme. | Zahlenwert (integer) |
| | **Linienart**<br>Legt die Linienart des Verlaufs fest. Für die höchste Darstellungsgeschwindigkeit wählen Sie eine durchgezogene Linie. | • Durchgezogen<br>• Strich<br>• Punkt<br>• Strichpunkt<br>• Strichpunkt-Punkt |
| | **Marker**<br>Zur besseren Unterscheidung von Verläufen können Sie die Kurven mit Markern kennzeichnen. | • Keine Marker<br>• Rechteck<br>• Rechteck (gefüllt)<br>• Kreis<br>• Kreis (gefüllt)<br>• Dreieck<br>• Dreieck (gefüllt) |

| | | |
|---|---|---|
| | | • Alternatives Dreieck<br>• Alternatives Dreieck (gefüllt)<br>• Kreuz<br>• Alternatives Kreuz |
| | **Verlauf anzeigen**<br>Zeigt den Verlauf als Line an. Dabei werden die vorliegenden Stützstellen miteinander verbunden. Zwischen den Stützstellen wird linear interpoliert. | True/False |
| | **Punkte anzeigen**<br>Zeigt die berechneten Stützstellen als kleine Kreuze an. | True/False |
| | **Obere Hüllkurve anzeigen**<br>Zeigt die obere Hüllkurve als Verbindung der lokalen Maxima an. | True/False |
| | Untere Hüllkurve anzeigen<br>Zeigt die untere Hüllkurve als Verbindung der lokalen Minima an. | True/False |
| | **X-Achse / Y-Achse** | |
| | **Maßeinheit**<br>Bietet die Möglichkeit den Verlauf in einer anderen Maßeinheit anzuzeigen.<br>Die Änderung hier hat keinen Einfluss auf das zugrundeliegende Modell. | Auswahlbox |
| | **Minimum**<br>Legt das Achsen-Minimum fest. | Zahlenwert (real) |
| | **Maximum**<br>Legt das Achsen-Maximum fest. | Zahlenwert (real) |
| | **Teilungsursprung**<br>Legt den Beginn der Achsenunterteilung fest. | Zahlenwert (real) |
| | **Teilungsintervall**<br>Legt die Größe der Teilungsintervalle fest. | Zahlenwert (real) |
| | **Negative Orientierung**<br>Kehrt bei Aktivierung die Richtung der Achse um. | True/False |
| | **Logarithmische Teilung**<br>Schaltet die logarithmische Unterteilung der Achse ein bzw. aus. | True/False |

| | | | | |
|---|---|---|---|---|
| | **Ursprung bei Polarkoordinaten**<br>(in Grad) | | | Zahlenwert (real) |
| | **Bezeichnung**<br>Legt den Text für die optionale Achsenbeschriftung fest. Wie für die Bezeichnung der Serie selbst, können Makros verwendet werden (siehe **7.3.5.3).** | | | Zeichenkette |
| | **Bezeichnung sichtbar**<br>Damit kann optional eine Achsenbeschriftung angezeigt werden. | | | True/False |
| | **Zahlenformat**<br>Legt das Zahlenformat für die Beschriftung der Achsenintervalle fest. | | | siehe **7.3.5.3** |
| | | Variable | | |
| | | **Ident**<br>Verweist auf die zum Verlauf gehörige X-Variable bzw. Y-Variable. Der Ident wird i.A. relativ zum Modell (siehe Eigenschaft *Modell* weiter oben) angegeben. | | Zeichenkette |
| | | **Index**<br>Verweist auf die korrekte Komponente einer mehrdimensionalen Variablen. Der Eintrag kann nur bearbeitet werden, wenn die Variable nicht skalar ist. | | Zahlenwert (integer) |
| | | Argument(e) | | |
| | | **Anzahl**<br>Zeigt die Anzahl zusätzlicher Argument für Verknüpfungen und andere Operationen | | |
| | | | **Variable** | |
| | | | **Ident**<br>Verweist auf eine Variable. Der Ident wird i.A. relativ zum Modell (siehe Eigenschaft *Modell* weiter oben) angegeben. | Zeichenkette |
| | | | **Index**<br>Verweist auf die korrekte Komponente eines mehrdimensionalen Arguments. Der | Zahlenwert (integer) |

| | | | | |
|---|---|---|---|---|
| | | | Eintrag kann nur bearbeitet werden, wenn die Variable nicht skalar ist. | |
| **Model**<br>Zeigt den Namen des zugehörigen Modells an | | | | Zeichenkette |
| **Anzahl Wertepaare**<br>Zeigt Anzahl der Wertepaare für diese Kurve an. Der Wert dient ausschließlich zur Information und kann daher nicht geändert werden. | | | | Zahlenwert (integer) |
| **Genauigkeit**<br>Gibt an, in welchem Gleitpunktformat die Werte zur Darstellung zwischengespeichert werden. Unabhängig davon werden die Simulationsergebnisse selbst immer mit doppelter Genauigkeit aufgezeichnet. | | | | Single bzw. Double |
| **Linie** | | | | |
| **Linientyp**<br>Zeigt den Linientyp an. Diese Einstellung kann nicht geändert werden. Zum Ändern des Linientyps müssen Sie die Linie löschen und durch den gewünschten Typ ersetzen. | | | | • Minimum<br>• Maximum<br>• Mittelwert<br>• Effektivwert<br>• Benutzerdefiniert jeweils Horizontal bzw. Vertikal |
| **Name**<br>Zeigt die Bezeichnung der Linie an oder legt diese fest. | | | | Zeichenkette |
| **Sichtbar**<br>Legt fest ob die Linie angezeigt werden soll oder nicht. | | | | True/False |
| **Name anzeigen**<br>Legt fest ob der Name in die Beschriftung der Linie aufgenommen werden soll. | | | | True/False |
| **Wert anzeigen**<br>Legt fest ob der aktuelle Wert in die Beschriftung der Linie aufgenommen werden soll. | | | | True/False |
| **Aktueller Wert**<br>Zeigt den aktuellen Wert, d.h. die Position der Linie im Koordinatensystem an. Der Wert kann nur bei benutzerdefinierten Linien geändert werden. | | | | Zahlenwert (real) |

| | | |
|---|---|---|
| | Darstellung | |
| | **Linienfarbe** | Farbauswahl |
| | **Linienstärke** | Zahlenwert (integer) |
| | **Linienart** | • Durchgezogen<br>• Strich<br>• Punkt<br>• Strichpunkt<br>• Strichpunkt-Punkt |
| **Transformation** | | |
| **Typ**<br>Gibt den Typ der Transformation bzw. Verknüpfung an. | | Zeichenkette |
| *Parameter 1…*<br>Die weiteren Parameter sind abhängig von der jeweiligen Transformation. | | |

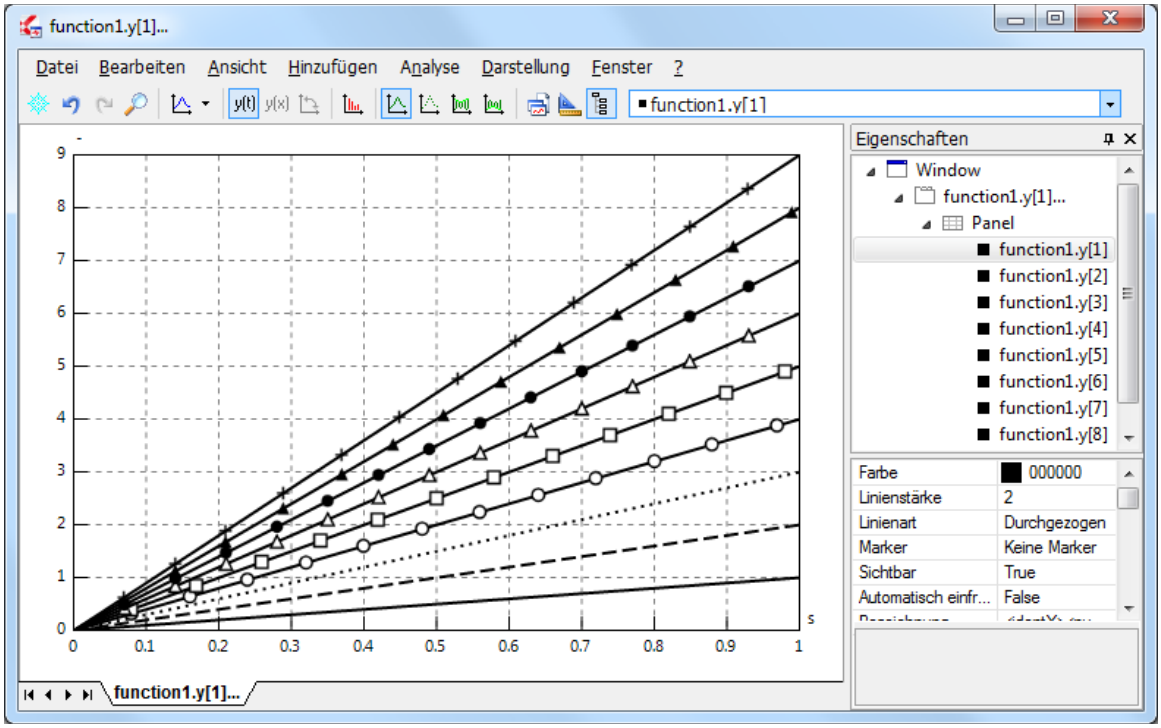
### 7.3.8.2 Laden und Speichern von Einstellungen

Die vorgenommenen Einstellungen für das Layout des Ergebnisfensters können mittels ***Darstellung/Einstellungen speichern*** gesichert werden (*.rsx-Format). Diese Einstellungen sind mit der Funktion ***Darstellung/Einstellungen laden*** wieder abrufbar. Dabei werden die geladenen Einstellungen sofort auf die jeweilige Darstellung angewandt. So wird es einfach möglich, zwischen verschiedenen Layouts, z.B. für den Ausdruck auf einem Farb- bzw. S/W-Laserdrucker zu wechseln. Um eine solche Vorlage zu erstellen, können Sie wie folgt vorgehen:

1. Erstellen Sie ein neues Simulationsmodell.
2. Positionieren Sie einen Funktionsblock f(x) in der Strukturansicht und tragen Sie als Wert für f(x) folgenden Ausdruck ein:

   ```
   time*{1, 2, 3, 4, 5, 6, 7, 8, 9}
   ```

   Die Anzahl der Vektorkomponenten sollte der maximal gewünschten Anzahl von Verläufen innerhalb eines Panels entsprechen.
3. Simulieren Sie das Modell mit den Standardeinstellungen.
4. Öffnen Sie ein Ergebnisfenster für die Ausgangsvariable y.
5. Nehmen Sie jetzt in dem Ergebnisfenster alle Einstellungen vor, die Sie als Vorlage speichern wollen, wie z.B. Linienfarbe, -stärke und -art.
6. Speichern Sie die Vorlage unter einem aussagekräftigen Namen (z.B. „Vorlage Druck S/W“).

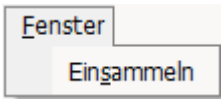



### 7.3.8.3 Festlegen der Standardeinstellung

Über das Menü ***Darstellung*** kann unter ***Als Standard speichern*** die vorgenommene individuelle Einstellung als Standard für alle neu anzulegenden Ergebnisfenster festgelegt werden. Die Standardeinstellungen werden als Datei *default.rsx* in dem unter den SimulationX-Verzeichniseinstellungen festgelegten Pfad für **Programmeinstellungen** gespeichert. Möchten Sie für ein bestehendes Ergebnisfenster die als Standard festgelegten Einstellungen übernehmen, können Sie über den Befehl ***Darstellung/Einstellungen laden*** diese Datei direkt laden.

## 7.3.9 Verwaltung von Ergebnisfenstern

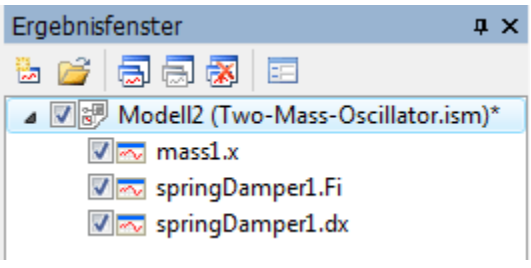
### 7.3.9.1 Zusammenführen von Ergebnisfenstern




Über das Menü ***Fenster/Einsammeln*** werden alle vorhandenen Ergebnisfenster des betreffenden Modells in einem Ergebnisfenster zusammengefasst, wobei die einzelnen Fenster als neue Seite in das Zielfenster eingefügt werden.

### 7.3.9.2 Der Ergebnisfenstermanager

Der Ergebnisfenstermanager dient der Verwaltung der Ergebnisfenster aller geöffneten Modelle.




Über die Checkboxen der einzelnen Einträge können Sie die Sichtbarkeit der Ergebnisfenster steuern. Dabei ist zu beachten, dass die Checkboxen für Modelle und Ergebnisfenster unterschiedliche Bedeutung haben.

**Aktivieren der Checkbox für ein Ergebnisfenster**
Das Ergebnisfenster wird zur Anzeige gebracht.

**Deaktivieren der Checkbox für ein Ergebnisfenster**
Das Ergebnisfenster wird minimiert und in Abhängigkeit der Einstellung zur Anzeige minimierter Ergebnisfenster als Symbol in der Taskleiste angezeigt. Gleichzeitig wird das betreffende Ergebnisfenster deaktiviert und damit von der weiteren Anzeige beim Aktivieren eines Modells ausgeschlossen. Diese Einstellung ist am grauen Symbol zu erkennen.

**Aktivieren der Checkbox für ein Modell**
Alle aktiven Ergebnisfenster des betreffenden Modells werden zur Anzeige gebracht. Deaktivierte Ergebnisfenster werden hingegen nicht angezeigt. Um alle zum betreffenden Modell gehörenden Ergebnisfenster anzuzeigen, verwenden Sie das Kontextmenü des Modells im Ergebnisfenstermanager.

**Deaktivieren der Checkbox für ein Modell**
Alle Ergebnisfenster des betreffenden Modells werden minimiert. Der Aktivierungszustand der Ergebnisfenster wird dabei nicht geändert.

Öffnet ein leeres Ergebnisfenster (wirksam, wenn Ebene Modell aktiv)

Öffnet ein Ergebnisfenster aus einer Datei (wirksam, wenn Ebene Modell aktiv)

Zeigt alle, d.h. auch die deaktivierten Ergebnisfenster an, unabhängig davon, zu welchem Modell sie gehören.

Deaktiviert und verbirgt alle Ergebnisfenster.

Schließt alle Ergebnisfenster

Öffnet den Eigenschaftsdialog für den Ergebnisfenstermanager. Folgende Einstellungen können vorgenommen werden:

- **Minimierte Ergebnisfenster in der Windows Taskleiste anzeigen**. Diese Funktion ermöglicht die Nutzung der Vorschaufunktion von Windows 7. Um die Taskleiste nicht mit minimierten Ergebnisfenstern zu überfrachten, sollten Sie diese Einstellung für Windows XP bzw. Vista deaktivieren.
- **Minimierte Ergebnisfenster in der Windows Taskleiste anzeigen**. Diese Funktion ermöglicht die Nutzung der Vorschaufunktion von Windows 7. Um die Taskleiste nicht mit minimierten Ergebnisfenstern zu überfrachten, sollten Sie diese Einstellung für Windows XP bzw. Vista deaktivieren.

- **Ergebnisfenster bei der Aktivierung eines Modells automatisch anzeigen.** Bei der Aktivierung eines Modells werden alle aktiven Ergebnisfenster, die zu diesem Modell gehören, zur Anzeige gebracht. Die übrigen, nicht zu dem aktivierten Modell gehörigen Fenster werden hingegen ausgeblendet.
- **Ergebnisfenster von Modellen aus SimulationX 3.4 (und davor) zusammenfassen.** Aktivieren sie diese Funktion, werden beim Laden eines Modells, das mit einer Vorgängerversion von SimulationX 3.5 erstellt wurde, die einzelnen Ergebnisfenster zu Seiten innerhalb eines einzigen Ergebnisfensters zusammengefasst.
- **Vorschaubild anzeigen**. Bewegen Sie die Maus über einen Eintrag im Ergebnisfenstermanager, wird eine Vorschau des betreffenden Fensters angezeigt.

### 7.3.9.3 Frequenzanalyse FFT

Die Frequenzanalyse gestattet die Ermittlung eines Amplitudengangs über der Frequenz für ein vorliegendes Zeitsignal (y(t)-Darstellung vorausgesetzt).

Wenn Sie die Schaltfläche betätigen oder den Menüpunkt ***Analyse/FFT*** aufrufen, wird für alle im Ergebnisfenster befindlichen Zeitverläufe das Amplitudenspektrum errechnet und in einem neuen Ergebnisfenster angezeigt. Eine weitere Möglichkeit zum Einfügen einer ***FFT*** besteht über das Menü ***Hinzufügen/Transformation***.
Die FFT wird stets für den im Fenster dargestellten Zeitbereich durchgeführt. Handelt es sich dabei um einen Ausschnitt, erhält man das Ergebnis einer so genannten Zoom-FFT. Über vorherige Ausschnittsbildung mit der Zoom-Funktion gemäß Abschnitt **Darstellung zoomen** lassen sich also Aussagen zum Frequenzinhalt an separaten Stellen eines Signals machen.

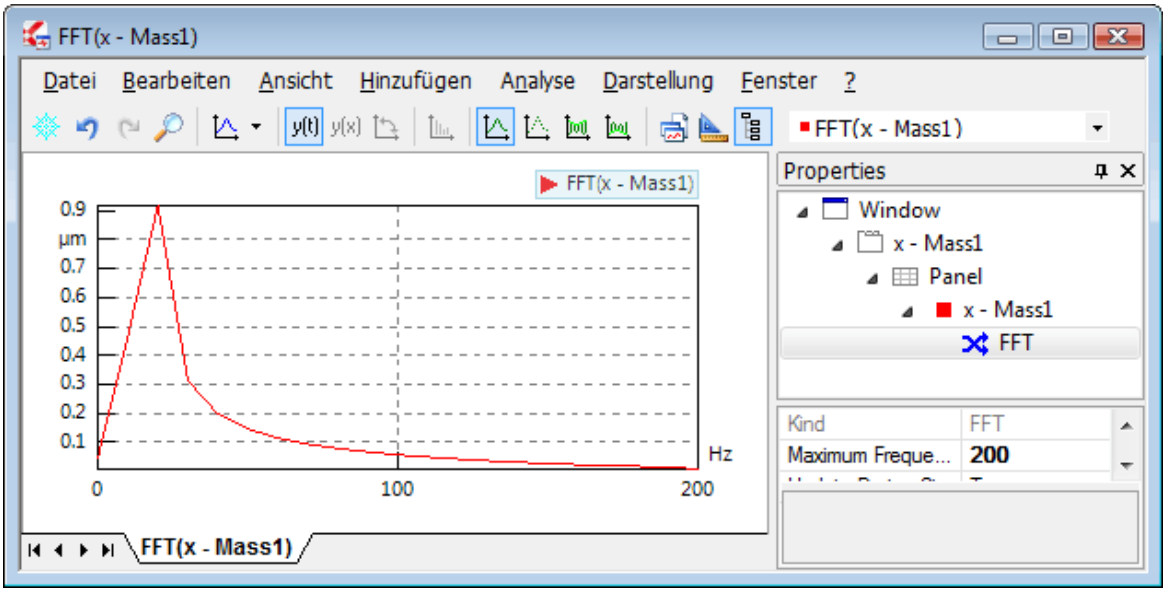



**Bild 7.31:** Frequenzanalyse einer einfachen Schwingung

### 7.3.9.4 Komplexer Übertragungsfrequenzgang (FRF)

Die Berechnung des komplexen Übertragungsfrequenzganges (FRF ... **F**requenzy **R**esponse **F**unction) bildet oft die Basis für eine Beurteilung der Struktur hinsichtlich des dynamischen Verhaltens im Frequenzbereich. Deshalb werden Frequenzgänge in der Signalanalyse auch für den Vergleich zwischen Rechen- und Messergebnissen herangezogen.

Der komplexe Übertragungsfrequenzgang eines Systems repräsentiert das komplexe Verhältnis zwischen zwei Signalen (Ergebnisverläufe aus dem Zeitbereich). Er ist definiert als die Fouriertransformierte des einen Signals dividiert durch die Fouriertransformierte des anderen Signals:

$$H(j\varpi) = \frac{Y(j\varpi)}{X(j\varpi)} \qquad \textbf{(7.1)}$$

X ... Signal2 (Eingang)

Y ... Signal1 (Ausgang)

H ... Übertragungsfunktion

Wird der komplexe Übertragungsfrequenzgang zwischen einem Ausgangssignal (Antwort) und einem Eingangssignal (Erregung) gebildet bedeutet er im physikalischen Sinn, dass ein Eingangssignal mit der Frequenz f eine Ausgangsbewegung mit der gleichen Frequenz bewirkt. Die Ausgangsamplitude ist die Eingangsamplitude, multipliziert mit $|H(f)|$, und die Phase des Ausgangs wird um $\varphi(f)$ gegenüber dem Eingang verschoben.

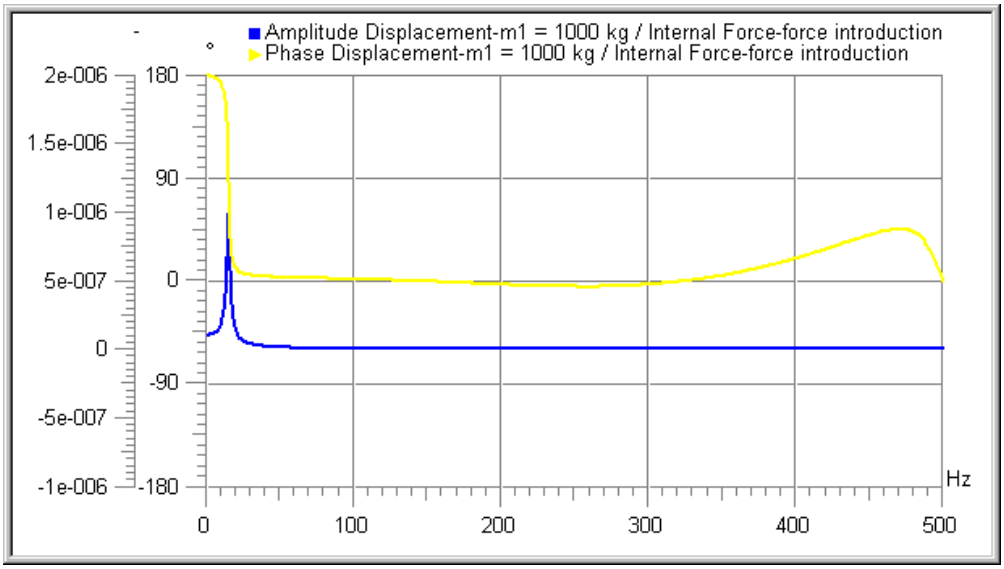



**Bild 7.32:** Komplexer Übertragungsfrequenzgang

Komplexe Übertragungsfrequenzgänge können aus dem Verhältnis verschiedener Ausgangs- und Eingangssignale gewonnen werden. Beispiele hierfür sind:

- Dynamische Nachgiebigkeit (Weg zu Kraft),
- Beweglichkeit (Geschwindigkeit zu Kraft),

- Beschleunigbarkeit (Beschleunigung zu Kraft)
- Dynamische Steifigkeit (Kraft zu Weg)
- Impedanz (Kraft zu Geschwindigkeit)
- Mitschwingende Masse (Kraft zu Beschleunigung).

## 7.4 Drucken (PrintEngine)

In diesem Abschnitt wird erläutert, wie Sie Modelle und Ergebnisse dokumentieren können. Alle zum Windows-Standard gehörenden Bedienhandlungen zum Drucken werden als bekannt vorausgesetzt.

Die ITI SimulationX PrintEngine ist über den Menübefehl ***Datei/Seitenansicht*** aufrufbar.

### 7.4.1 Seitenansicht

Über ***Datei/Seitenansicht*** erreichen sie die Einstellungen für den Druck. Es öffnet sich das Seitenansicht-Fenster (siehe **Bild 7.32**). Hier sind **alle Eingaben und Ergebnisse** zum Ausdrucken aufbereitet.
Das Drucklayout (ständig in der Druckvorschau verfügbar) kann den individuellen Erfordernissen angepasst werden.

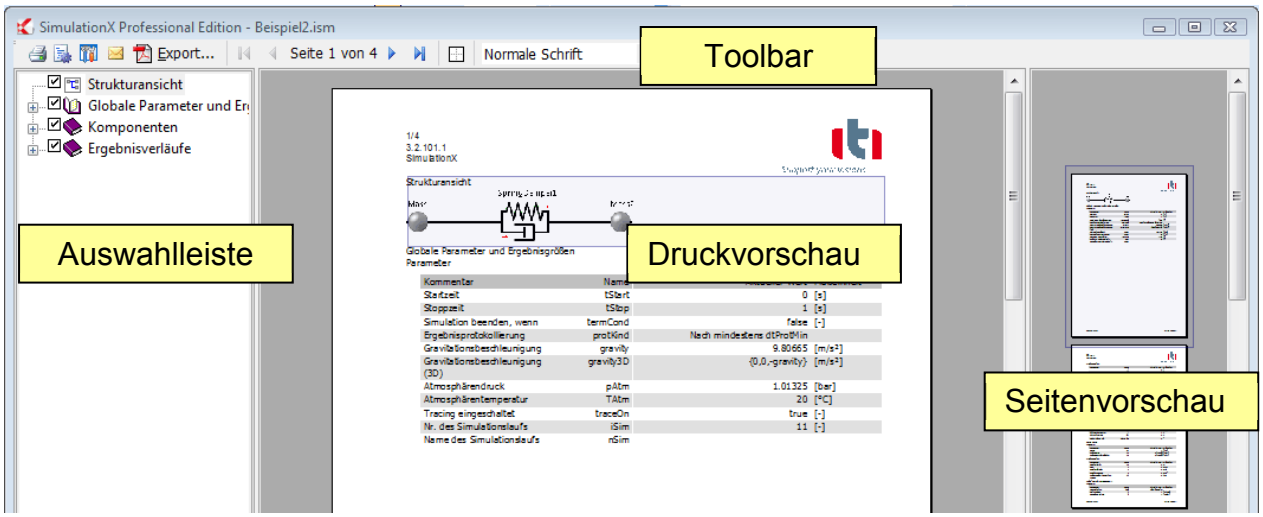



**Bild 7.33:** SimulationX – Seitenansicht

#### 7.4.1.1 Auswahlleiste

In einer hierarchischen Baumansicht werden alle druckbaren Bestandteile, wie

- Strukturansicht,
- globale Parameter und Ergebnisse,
- Komponenten und
- Ergebnisverläufe

des aktuellen Modells aufgelistet.

Durch Auswahl im Kontrollkästchen des Elements (Haken setzen bzw. entfernen) können Sie einzelne Bestandteile in den Druck aufnehmen bzw. ausschließen.

Jede Änderung wird in der Druckansicht sofort angezeigt.
Zur Auswahl können Sie zwischen der

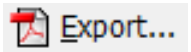
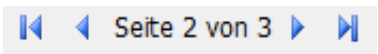
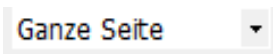
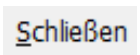
- Dokumentenstruktur und der
- Ansicht

umschalten.

In der Dokumentenstruktur werden die druckbaren Bestandteile nach strukturellen Kriterien sortiert. Jede Auswahl bezieht sich auf den ausgewählten Bestandteil und die darunter liegenden.
In der Ansicht werden die druckbaren Bestandteile nach inhaltlichen Kriterien sortiert.

### 7.4.1.2 Toolbar

Zur Drucklayoutgestaltung stehen eine Reihe von Bedienfunktionen zur Verfügung.

| | |
|---|---|
| | Öffnet den Windows-Dialog zum Drucken. |
| | Diese Schaltfläche öffnet den Dialog zur Seiteneinrichtung. Papiergröße und – quelle, Orientierung und Ränder können eingestellt werden. |
| | Öffnet den Windows-Dialog zur Druckereinrichtung. |
| | Generiert eine E-Mail, der das Druckdokument als Anlage „report.pdf“ angefügt wird. |
|   | Mit dieser Schaltfläche lässt sich das Druckdokument als **PDF**-, **RTF**- oder **HTML**-Datei exportieren. |
|   | Mit dieser Schaltfläche kann innerhalb des Dokuments zwischen den Seiten geblättert werden. |
| | Die Randmarkierungen werden ein- bzw. ausgeblendet. |
| | Auswahl der Schriftgröße (5 Stufen). |
|   | Mit dieser Auswahlliste kann die Darstellung in der Druckansicht geändert werden. |
|   | Schließen der Seitenansicht. |

### 7.4.1.3 Seite einrichten

Über die entsprechende Schaltfläche gelangen sie zum Dialog für die Einrichtung der Seite.

Es können die folgenden Eigenschaften für das Layout der Seite festgelegt werden (siehe **Bild 7.33**):

- Format der Seite (Papiergröße, Ausrichtung, Seitenränder)
- Kopfzeile (Beschriftung und Schriftgröße)
- Fußzeile (Beschriftung und Schriftgröße)
- Design (klassisch, modern, einfach, farbig)

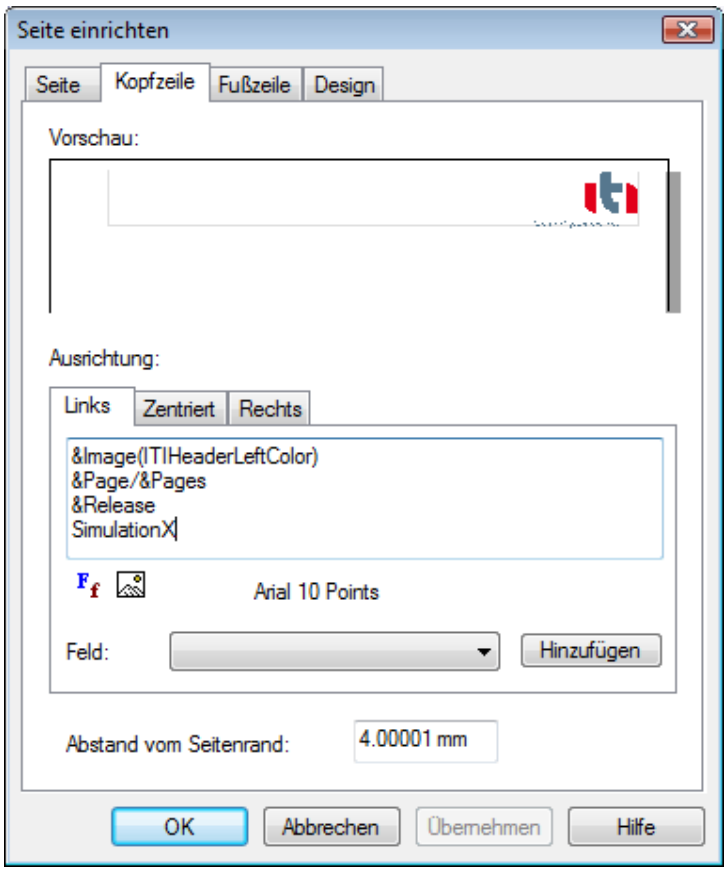



**Bild 7.34:** Dialog zur Einrichtung der Seite

Für die Beschriftung der Kopf-/Fußzeile steht eine Auswahl vordefinierter Feldtypen zur Verfügung.

Es können folgende Felder eingefügt werden:

| Name | Funktion |
|---|---|
| Page | Nummer der aktuellen Seite |
| Ungerade Seitenzahl | Nummer der ungeraden Seite |
| Gerade Seitenzahl | Nummer der geraden Seite |

| | |
|---|---|
| Gesamtseitenzahl | Anzahl der Gesamtseiten |
| Dateiname | Dateiname des Modells |
| Bearbeiter | Autor des Modells |
| Aktuelles Datum | aktuelles Datum |
| Aktuelles Datum mit Uhrzeit | wie vor, jedoch mit Angabe der aktuellen Zeit |
| Firmenname | |
| Copyright | |
| Programmversion | Versionsnummer von SimulationX |
| Release Nummer | Release Nummer von SimulationX |
| Release Nummer mit Datum | wie vor jedoch mit Datum |
| Grafik | es öffnet sich ein Dialog zur Auswahl einer Grafik |

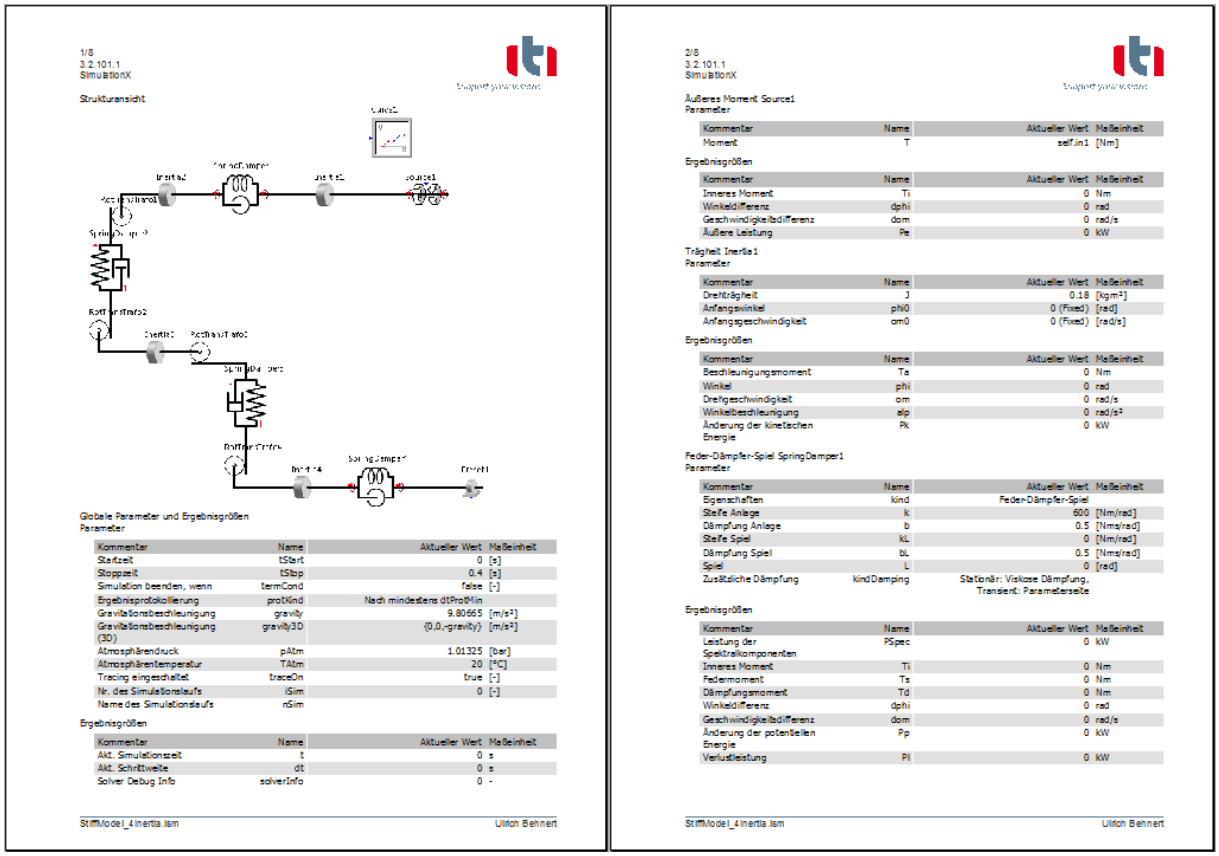


**Bild 7.35:** Dokumentation eines Modells

## 7.4.2 Benutzerdefinierter Druck

Zur benutzerdefinierten Druckausgabe kann die vorhandene PrintEngine über das COM-Interface gesteuert werden. Über die Erstellung von Skripten können Sie benutzerdefinierte Druckformulare generieren. Die Strukturierung der Dokumente

ist über Frame-Sets möglich, und es können Grafiken und Tabellen zur Gestaltung eingefügt werden.

Das folgende Skript erzeugt ein Modell und ein mehrseitiges Dokument, das in der Seitenansicht dargestellt wird. Sie finden dieses Beispiel unter „…\ITI SimulationX 3.5\Samples\Scripting\Print_simple_model.vbs“.

```
' Konstanten
simStopped = 16
filepath= "C:\Programme\ITI-Software\ITI SimulationX
3.5\Samples\Scripting\"
' Start von ITI-SimulationX mit einem neuen Dokument

Set sim = CreateObject("iti.sim3")
Set doc = sim.Documents.Add()
sim.visible=true

' Aufbau eines einfachen Simulationsmodells
Set simobj=doc.SimObjects.Add
("Mechanics.Translation.Mass", "Mass1", 90, 90)
Set simobj2=doc.SimObjects.Add
("Mechanics.Translation.Spring", "Spring1", 180, 90)
Set simobj3=doc.SimObjects.Add
("Mechanics.Translation.Damper", "Damper1", 180, 150)
doc.Connections.Add "Mass1.ctr2", "Spring1.ctr1",
"Connection1"
doc.Connections("Connection1").AddLine "Damper1.ctr1", 145,
105

' Parametrieren des Modells

doc.SimObjects("Mass1").Parameters("m").Value = 0.5
doc.SimObjects("Spring1").Parameters("k").Value = 1000
doc.Connections("Connection1").Parameters("x0").Value =
0.01
doc.Connections("Connection1").Results("x").Protocol = true
doc.Connections("Connection1").Results("v").Protocol = true

' Öffnen von Ergebnisfenstern
Set wnd = sim.ResultWindows.Add("Connection1.x")
wnd.SetMinMax 0.0, 1.0, -0.01, +0.01

' Rechnung Dämpfung b = 2

    doc.Parameters("Damper1.b").Value = 2
    doc.Reset
    doc.Start
```

```
    While (doc.SolutionState <> simStopped)
     wscript.Sleep(100)
    Wend
' Drucken des Modells
Set pe = doc.PrintEngine

If pe Is Nothing Then
    MsgBox "Error getting printengine"
Else
     ' Arrangieren der Druckobjekte
     pe.Objects.AddSimObject simobj
     pe.Objects.AddSimObject simobj2
     pe.Objects.AddSimobject simobj3
     pe.Objects.AddModelView

     'Erzeugen einer weiteren Seite
     set p=pe.Pages.Add

     ' Einfügen von Bildblöcken
     set imgblock=p.Objects.AddBlock
            ("ImageBlock","Parameters of mass1")
     imgblock.LoadImage filepath & "pic1.gif"
     ' imgblock.PrintName=false
     set imgblock1=p.Objects.AddBlock
            ("Imageblock",Chr(13) & "Paramter of Spring1")
     imgblock1.LoadImage filepath & "pic2.gif"
     set imgblock2=p.Objects.AddBlock
            ("Imageblock",chr(13) & "Parameter of Damper1")
     imgblock2.LoadImage filepath & "pic3.gif"
     ' Einfügen eines Ergebnisfensters
     p.Objects.AddResultWindow wnd

     ' Öffnen der Druckvorschau
     pe.PrintPreview

End If
```

Das Ergebnis des script-gesteuerten Druckes zeigt **Bild 7.35**. Die eingefügten Bilder auf Seite 2 wurden mit Hilfe eines Capture-Programmes erzeugt. Der Seitenaufbau entspricht der Reihenfolge der Notation im Script.

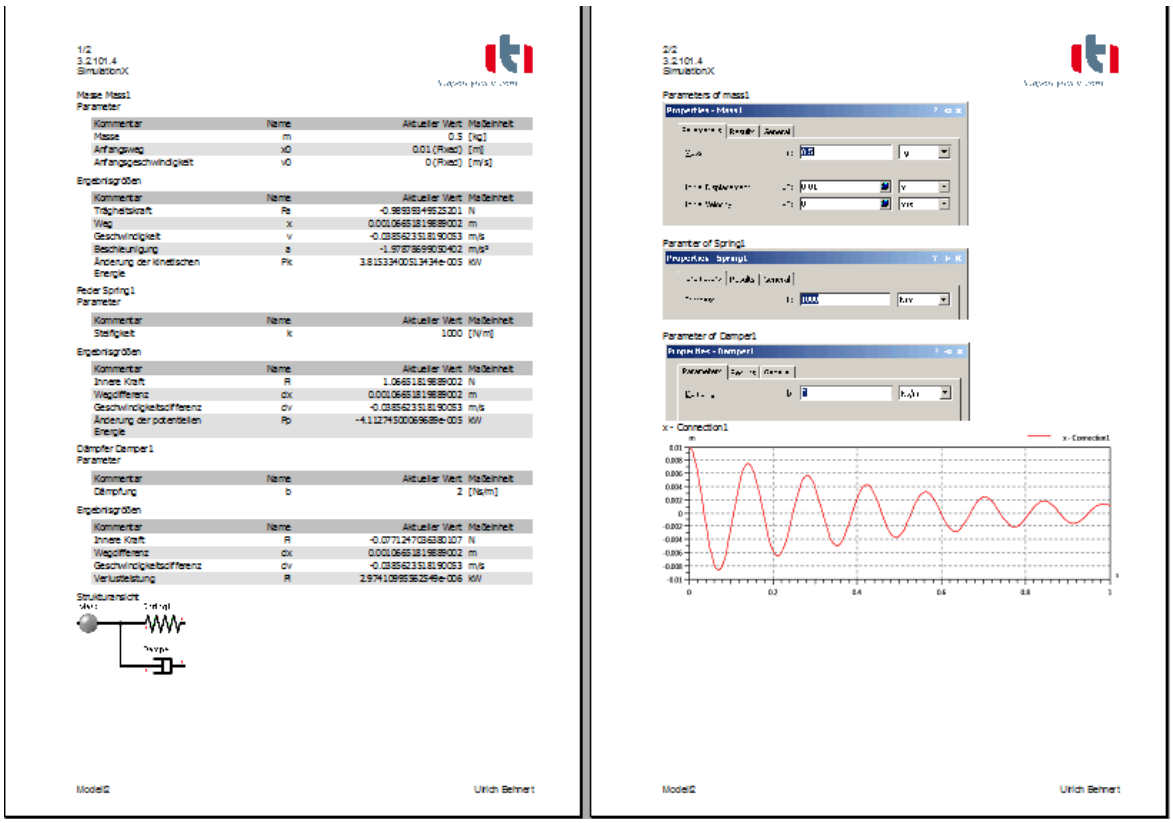


**Bild 7.36:** Benutzerdefiniertes Drucklayout

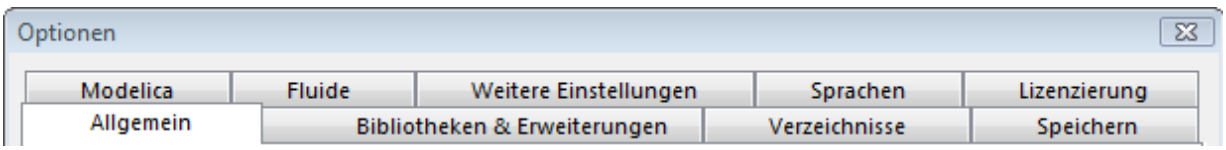
## 7.5 Schnittstellen zu MS-Office, Import/Export

SimulationX bietet eine Vielzahl von Möglichkeiten, Sollgrößen außerhalb des Programms zu erstellen bzw. Ergebnisse für andere Anwendungen bereitzustellen.

Mit MS-Office-Anwendungen (z. B. Excel, Access) sowie einer Vielzahl anderer Anwendungen können Werte, Kennlinien, Kennfelder oder Kurvenscharen erzeugt und für die Parametrierung einzelner Elemente (z.B. Signalgeber) genutzt werden. Ergebnisgrößen können über die Ergebnisfenster (siehe **Kap. 7.3.3**) abgespeichert und anschließend mit Office-Anwendungen weiter genutzt werden. Hier ist besonders das CSV-Format geeignet.

Über die vorhandene COM-Schnittstelle (siehe **Kap. 9.2**) können Parameter importiert und Ergebnisse ausgegeben werden. Dies ist hilfreich bei der Automation von Simulationsläufen.

## 7.6 Einstellungen zu SimulationX




Über das Menü ***Extras/Optionen*** sind weitere Einstellungen zu SimulationX verfügbar.

Bei der Installation von SimulationX werden diese auf Standardwerte eingestellt und können jederzeit geändert werden.

### 7.6.1 Allgemeine Einstellungen

Über dieses Register können Sie Einstellungen zum Programm-Start, zur Bibliotheksleiste und zur Animation vornehmen.

Alle vorgenommenen Einstellungen können mit der Schaltfläche Einstellungen zurücksetzen rückgängig gemacht werden.

Wenn Sie während der Simulation eine Visualisierung von Momentanzeigen, Modelica Grafikprimitiven oder Statecharts vornehmen wollen, muss das Control „***Animation der Strukturansicht während der Simulatioinsrechnung zulassen***“ im Register ***Allgemein*** gesetzt sein. Dies kann die Rechenzeit erhöhen.

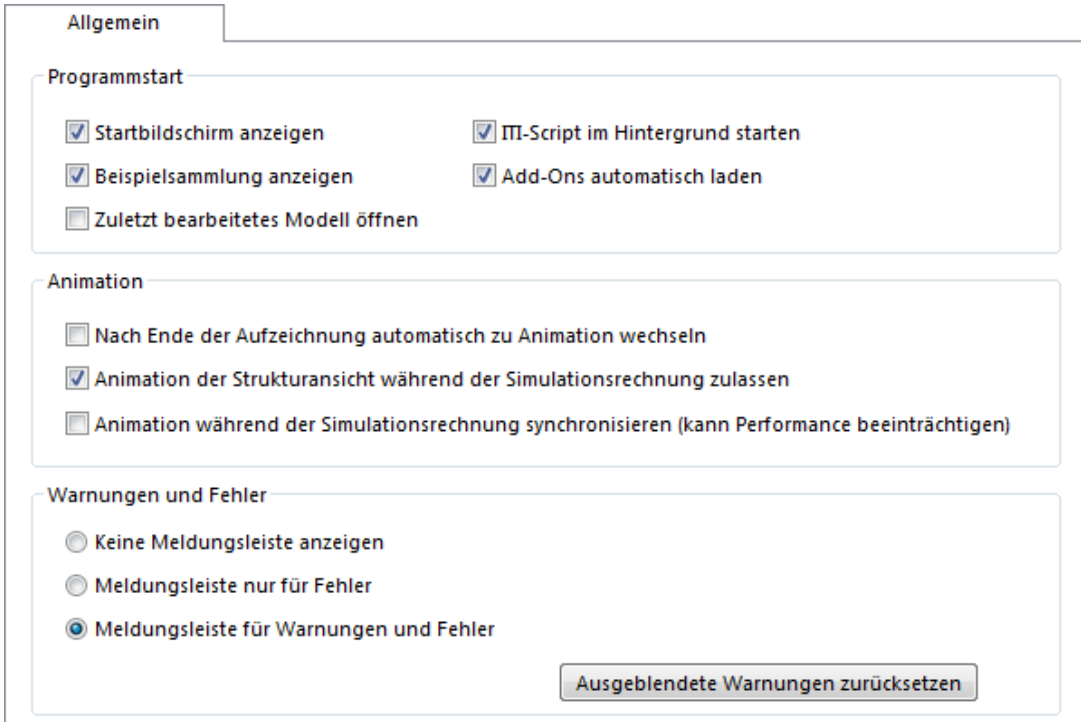



**Bild 7.37:** Allgemeine Einstellungen zum Programm

### 7.6.2 Bibliotheken und Erweiterungen

Mit dem Menüpunkt ***Optionen/Bibliotheken&Erweiterungen*** wählen Sie die Bibliotheken und Erweiterungen aus, die beim Starten des Programms geladen werden. Dies geschieht durch Setzen des Häkchens. Alle markierten Bibliotheken und Erweiterungen werden geladen.

Welche davon nutzbar sind, wird durch Ihre Lizenz zu SimulationX bestimmt. Indem Sie nicht lizenzierte oder benötigte Bibliotheken entfernen, schonen sie Ressourcen und reduzieren die Startzeit von SimulationX.

**Bild 7.38:** Auswahl der Bibliotheken

## 7.6.3 Verzeichnisse

In der Menüoption ***Optionen/Verzeichnisse*** besteht die Möglichkeit, die von SimulationX bei der Installation eingerichteten Pfade (siehe **Bild 7.39**) anzupassen.

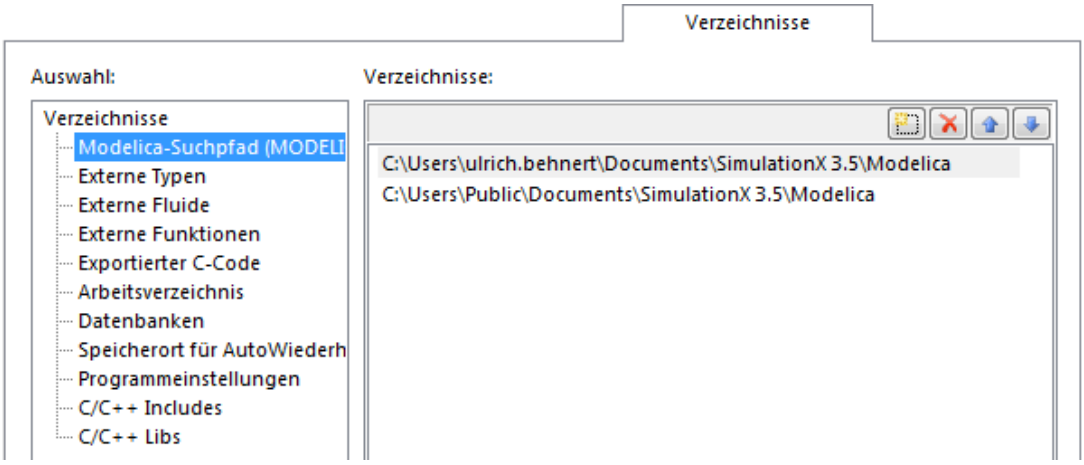



**Bild 7.39:** Einstellungen der Verzeichnisse

Mit den Schaltflächen können Sie

neue Verzeichnisse hinzufügen

den ausgewählten Verzeichniseintrag löschen und

Verzeichniseinträge verschieben bewegen.

## 7.6.4 Speichern

Standardmäßig erfolgt eine Speicherung der Ergebnisvariablen im Hauptspeicher. Optional besteht die Möglichkeit zur Speicherung der Daten auf der Festplatte. Die entsprechende Einstellung erfolgt unter ***Optionen/Speichern***.

Des Weiteren besteht die Möglichkeit das Standardspeicherformat (*.isx oder *.ism) für das Speichern der Modelle festzulegen.

Die Option Modelica konforme Speicherung bedeutet, ein Speichern als *.mo Datei ohne Ergebnisse und ohne SimulationX-Berechnungsumgebung.

Das Speichern von gerechneten Modellen mit Ergebnisgrößen führt zu einem hohen Speicherbedarf auf der Festplatte. Dies kann mit der Option „*Standardmäßig ohne Berechnungsergebnisse speichern*“ verhindert werden.

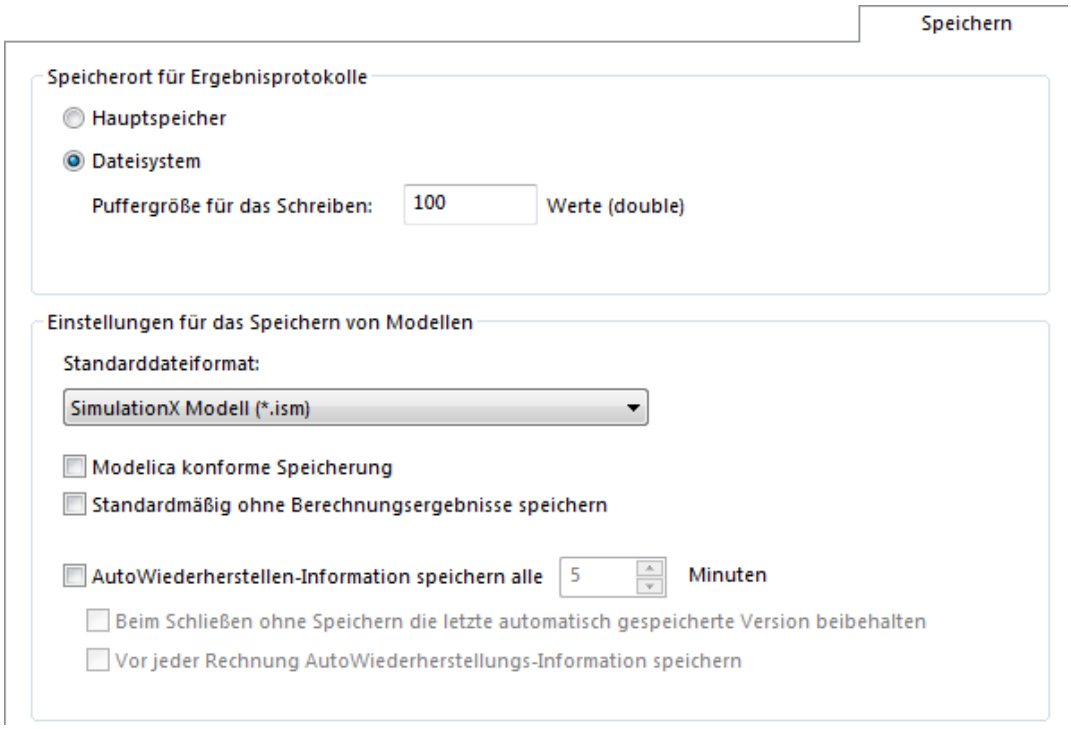



**Bild 7.40:** Einstellung für das Speichern der Ergebnisvariablen

Mit dem Feature **AutoWiederherstellen** können Sie Ihre Arbeit zusätzlich schützen. Mit diesem Feature wird in regelmäßigen Abständen eine temporäre Kopie der Datei erstellt, an der Sie gerade arbeiten. Wenn Sie Ihre Arbeit nach einem Stromausfall oder einem ähnlichen Problem wiederherstellen möchten, muss das Feature **AutoWiederherstellen** vor Auftreten des Problems aktiviert gewesen sein. Sie können für die automatische Wiederherstellung ein Speicherintervall von 5 Minuten (Standardeinstellung) oder größer festlegen.

Wenn bei aktiviertem Feature **AutoWiederherstellen** SimulationX nicht mehr reagiert und Sie Dateien geöffnet haben, bekommen Sie beim Neustart von SimulationX über das Dialogfeld **Wiederherstellen von Dateien** alle wieder herstellbaren Dateien angezeigt.

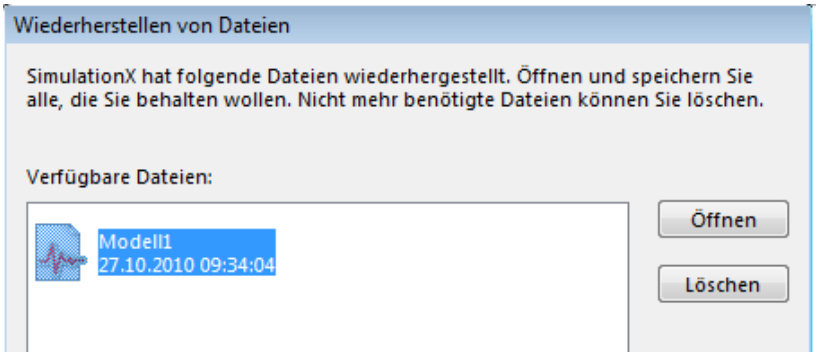



**Bild 7.41:** Wiederherstellen-Dialog

Die Informationen in den Dateien spiegeln den Stand nach der letzten automatischen Speicherung wieder. **AutoWiederherstellen** ersetzt nicht das manuelle Speichern oder das Sichern von Backups Ihrer Dateien.

### 7.6.5 Modelica

Hier können die Einstellungen für den Umgang mit Modelica-Dateien (*.mo) erfolgen.

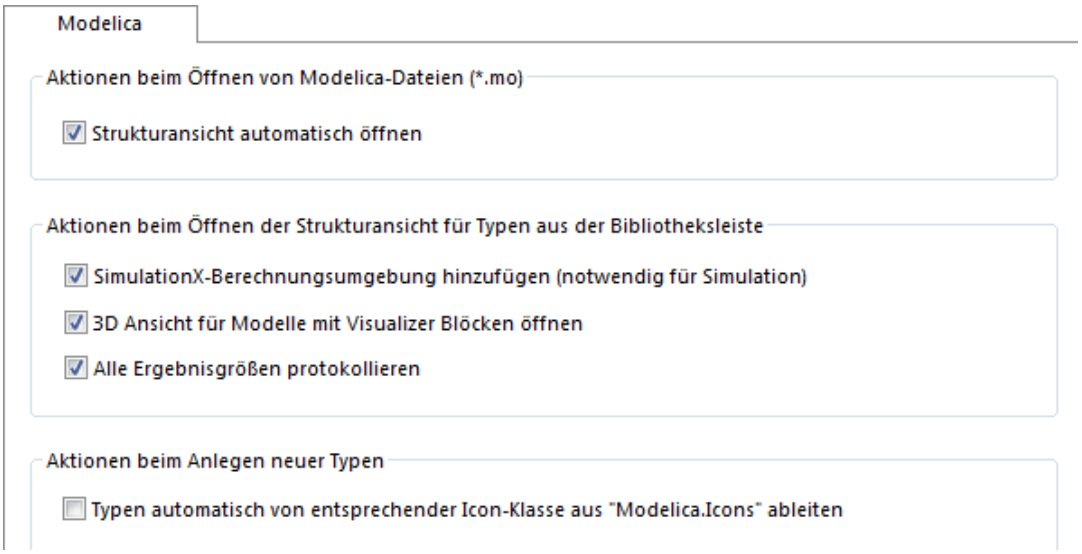



**Bild 7.42:** Modelica Einstellungen

Zu Durchführung einer transienten Simulation ist es notwendig die SimulationX-Berechnungsumgebung hinzuzufügen.

### 7.6.6 Fluide

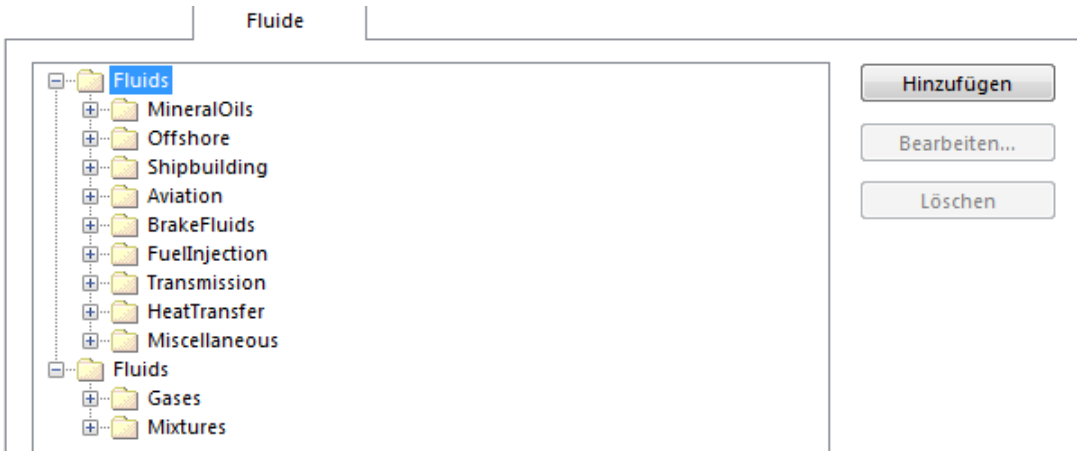



**Bild 7.43:** Auswahl des Mediums

Hier können Sie das zu verwendende Medium für Fluid-Anwendungen auswählen. Mit dem Button ***Hinzufügen*** wird der FluidDesigner zur Erweiterung der Fluid-datenbank gestartet. Näheres dazu finden Sie unter Kapitel 8.3.Beschriftung

Die Beschriftung der Elemente und Verbindungen ist individuell den Erfordernissen anpassbar und ist in **Kapitel 5.2.3.3** erklärt.

### 7.6.7 Sprachen

Über diesen Menüpunkt ist eine Umstellung der Sprache von SimulationX möglich. Es werden die verfügbaren Sprachen zur Auswahl angezeigt. Klicken Sie Ihre Auswahl mit der Maus an und übernehmen diese, wird die Sprache für die Bedienoberfläche von SimulationX umgestellt.

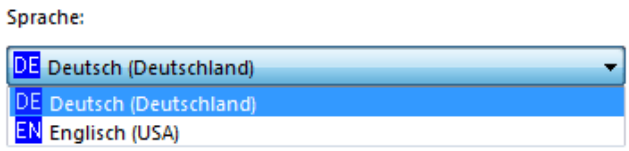



**Bild 7.44:** Einstellung der Sprache

### 7.6.8 Lizenzierung

Das Umschalten auf eine andere Lizenz erfolgt über das Menü ***Extras/Optionen/ Lizenzierung***. Betätigen Sie dazu die Schaltfläche ***Edition ändern in…*** und wählen Sie die entsprechende Edition aus. Siehe auch **Kapitel 2.**

### 7.6.9 Programmanpassung

Mit dem Menü ***Extras/Anpassung*** können Sie die Bedienoberfläche von SimulationX (Befehle, Symbolleisten, Menüs, u. a.) anpassen.

Sollten nach einem Update von SimulationX Symbole in den Symbolleisten nicht ordentlich dargestellt werden hilft meist ein ***Zurücksetzen***.

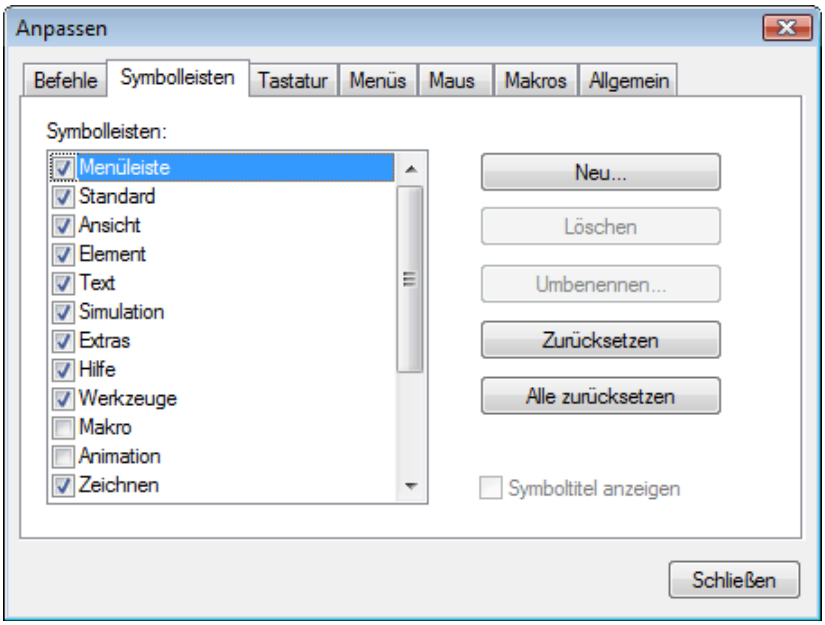



**Bild 7.45:** Anpassen der Oberfläche von SimulationX

SIMULATION X®

Powered by ITI



# Customization and Extensions

# 8 Erweiterte Modellierung

In diesem Kapitel finden Sie Wissenswertes zu Erstellung eigener Typen und das Einfügen von nutzerdefinierten Fluiden für die Hydraulik und Pneumatik, sowie den Statechart Designer.

## 8.1 Verwaltung von Externen Typen in SimulationX

Unter dem Begriff Externe Typen oder auch eigene Typen sind Element-Typen (in Modelica® Modell genannt) zu verstehen, die der Anwender von SimulationX selbst erstellt. Die Arbeit mit Modelica-Klassen wird in einem separaten Kapitel beschrieben. Einige der folgenden Ausführungen gelten aber auch dafür.

Ein **Element-Typ** ist die **Definition aller Eigenschaften**, die ein Element später in einem Modell besitzen soll, also angefangen vom Symbol, den Parametern und Ergebnisvariablen sowie dem Verhalten (Gleichungen..).
Ein Element-Typ enthält Komponenten von Typen. Das kann ein Parameter von Typ Real sein oder auch ein Element vom (Element-)Typ Mechanics.Translation.Mass in einer Substruktur.

Definiert der Anwender einen Element-Typ, so geschieht das im Allgemeinen mittels des integrierten TypeDesigners. Im Verlaufe der Definition wird auch festgelegt, in welcher Form und an welchem Ort diese Definition gespeichert werden soll sowie, welche Schutzmechanismen benutzt werden. Beide Eigenschaften bestimmen, welche Manipulationen der Anwender selbst oder ein Dritter mit diesem Element-Typ machen kann.

Zunächst wird unterschieden zwischen **globalen Element-Typen** und **lokalen Element-Typen**.

**Globale Element-Typen** findet man innerhalb von SimulationX in der Bibliotheksleiste unter dem Eintrag ***Externe Typen***. Sie sind global im gesamten Programm, d.h. für jedes Modell verfügbar, analog zu den Element-Typen in den Bibliotheken. Globale Element-Typen werden als separate Datei gespeichert. Der Speicherort wird durch die Einstellungen in ***Extras/Optionen/Verzeichnisse/ExterneTypen*** festgelegt.

**Lokale Element-Typen** werden innerhalb eines Modells gespeichert. Sie sind im Modellexplorer unter der Registerkarte Typen zu finden. Sie sind nur innerhalb dieses Modells (lokal) verfügbar.

Typ in Typ

Grundsätzlich kann jeder Element-Typ als separate Einheit definiert werden:

- Eine Datei für jeden globale Typen

- lokale Typen jeweils als ein Eintrag in der Liste im Modellexplorer

Es ist aber auch möglich, Element-Typen innerhalb von anderen, bereits existierenden Element-Typen zu speichern. Es entsteht dann eine Baumstruktur. Diese Abhängigkeit kann aber Einschränkungen in der Flexibilität zur Folge haben.

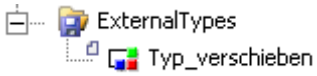
Verschieben und Kopieren innerhalb SimulationX

Der Speicherort von Element-Typen innerhalb von SimulationX kann geändert werden. Meist ist damit auch eine Änderung der physischen Speicherung (Datei) verbunden. Element-Typen können dabei verschoben oder kopiert werden.

Diese Funktion verwendet man, um globale Element-Typen in lokale umzuwandeln und umgekehrt oder um den Platz eines Element-Typen innerhalb der Baumstruktur zu verändern. Die Bedienung ist im **Kapitel 4** Bedienoberfläche beschrieben.
Das Kopieren oder Verschieben wird durch Verschlüsselung des Element-Typen eventuell eingeschränkt.

Das Verschieben eines Element-Typs aus der Bibliotheksleiste in die (Typen-) Baumstruktur des Modellexplorers bzw. umgekehrt ändert die Referenz im Element (Modellkomponente). Die Referenz muss geändert werden, weil beim Verschieben der Element-Typ nicht mehr am alten Ort vorhanden ist und die Referenz nicht mehr gültig wäre.

Beispiel:
Wird der globale Element-Typ




im Modell verwendet, so kann wird die Referenz im Eigenschaftsfenster, Seite ***Allgemein*** oder im Modelica Code des Modells angezeigt werden.
*ExternalTypes.Typ_verschieben*

Verschiebt man den Element-Typ per *Drag & Drop* in das Modell,

so ändert sich die Referenz vom Element zum Element-Typ in *Modell2.Typ_verschieben,* wobei die Referenz im Modelica Code den Modellnamen nicht enthält.

In der Bibliotheksleiste ist der Element-Typ jetzt nicht mehr vorhanden. Er kann nicht mehr für andere Modelle verwendet werden.
Modelle, die vor dem Verschieben erzeugt wurden und während des Verschiebens nicht geöffnet sind, enthalten immer noch die Referenz *ExternalTypes.Typ_verschieben*. Diese Modelle werden nun nicht mehr korrekt geöffnet. Es wird eine Meldung ausgegeben, dass der entsprechende Element-Typ fehlt.

Element-Typen lassen sich auch aus dem Modell in die Bibliotheksleiste unter ExternalTypes verschieben. Auch dabei wird die Referenz geändert.

Werden Element-Typen per *Drag & Drop* **kopiert**, so wird am Modell, d.h. an der Referenz zum Element-Typ nichts geändert. Für das Modell wird weiterhin der ursprüngliche Element-Typ benutzt.
Beim Kopieren von Element-Typen sollte immer berücksichtigt werden, dass bei diesem Vorgang **unabhängige Kopien** entstehen. Diese Kopien können trotz gleich lautendem Namen verschiedene Inhalte haben.

Lokale Element-Typen lassen sich außerdem mit einer Exportfunktion in die Bibliotheksleiste *ExternalTypes* verschieben oder kopieren. Klick man im Modellexplorer mit der rechten Maustaste auf den Eintrag des Typen, so kann man in dem Kontextmenü die Funktion *Exportieren* auswählen.

Überschreiben schon vorhandener Typen

In *ExternalTypes* oder im Modell vorhandene Element-Typen lassen sich durch die Verschiebe- oder Kopierfunktion innerhalb von SimulationX nicht überschreiben. Der zu ersetzende Typ müsste vorher gelöscht und der neue an dessen Stelle verschoben oder kopiert werden.

Modelle mit externen Typen

Wenn Modelle mit externen globalen Element-Typen aufgebaut wurden und diese Modelle anschließend auf einem anderen Computer berechnet werden sollen, so müssen die verwendeten Element-Typen auf dem anderen Computer vorhanden sein. Das ist auf drei verschiedenen Wegen möglich:

1. Die Dateien *.mo der externen Element-Typen sind in einem Netzwerk so gespeichert, dass SimulationX von beiden Computern aus dieselben Dateien lädt und verwendet.

2. Die Dateien für die externen Element-Typen müssen auf beiden Computern vorhanden sein, wobei:
   - Name und Inhalt identisch sein sollten und
   - die Struktur der Bibliothek *ExternalTypes* auf jedem Computer identisch sein muss.

Bei Änderungen an den Element-Typen sind die *.mo Dateien auszutauschen. Das kann auch während der Laufzeit von SimulationX geschehen, allerdings ist vor einer erneuten Rechnung die Funktion ***Neu Laden*** aufzurufen (siehe **Kapitel 4.2.2.1**).

3. Die Element-Typen sind als lokale Element-Typen im Modell gespeichert und werden mit dem Modell weitergegeben.

Um die Element-Typen als lokal im Modell zu speichern, stehen spezielle Funktionen bereit. Diese kopieren den globalen Element-Typ in das Modell (lokalen Typ erzeugen) und ändern automatisch die Referenz vom Element zum Element-Typ. Der globale Element-Typ in der Bibliothek bleibt erhalten.

- Menü ***Bearbeiten / Auflösen Externer Referenzen***
- Menü ***Datei / Export***
- Menü ***Datei / Senden***

Alle drei Funktionen erstellen einen speziellen Eintrag im Modellexplorer - ImportedTypes. Unter diesem Eintrag befinden sich dann die nun lokalen Element-Typen. Außerdem können dort auch noch importierte Fluid-Typen gespeichert sein.

Laden von externen Element-Typen in die Bibliothek ExternalTypes

Globale Element-Typen werden beim Start von SimulationX bzw. nach Öffnen des Ordners *ExternalTypes* in der Bibiotheksleiste aus den eingestellten Verzeichnissen in den Arbeitsspeicher geladen und dort gehalten.
Wenn die Datei *.mo eines geänderten Element-Typen in das entsprechende Verzeichnis kopiert und damit eine vorhandene Datei überschrieben wird, so hat das zunächst keine Auswirkung auf den Element-Typen in der Bibliothek *ExternalTypes*. Erst nach Aufruf der Funktion ***Neu Laden*** steht die aktualisierte Definition zur Verfügung.
Beim Ändern von Element-Typen in SimulationX wird nach ***Fertigstellen*** im TypeDesigner bzw. ***Speichern*** nach dem Ändern der Struktur die Datei geändert.

In allen Fällen, in den mehrere Nutzer externe Element-Typen gemeinsam benutzen, ist eine Verwaltung durch einen der Nutzer empfehlenswert.

Verwaltung außerhalb von SimulationX

Globale externe Element-Typen werden in Dateien mit der Erweiterung *.mo* abgespeichert.
Der Dateiname setzt sich zusammen aus dem Namen des Element-Typs, einem Doppelkreuz # und einem Versionszähler. Normalerweise sind die Dateien als Textformat gespeichert. Wenn der Element-Typ verschlüsselt bzw. mit Passwort gespeichert wird, so wird ein Binärformat verwendet.

Wird ein vorhandener Element-Typ bearbeitet, so wird beim Speichern eine Sicherungskopie angelegt. Die Sicherheitskopie bekommt die Dateinamenserweiterung * .mo1. Bei weiteren Änderungen wird dieser Zähler immer um 1 erhöht. Die Datei mit dem höchsten Index ist die jüngste Kopie.

Wird in der Bibliothek *ExternalTypes* eine neue Package gebildet, so wird ein neues Verzeichnis angelegt. Bei der Eingabe des Namens, der auch der Verzeichnisname wird, wird automatisch auf eine für SimulationX gültige Syntax geachtet. Berücksichtigt wird dabei die Menge der bereits vorhandenen Namen. Es darf auf gleicher Ebene keine Package und keinen Typen mit demselben Namen geben.
Innerhalb des neuen Verzeichnisses wird eine Datei *package.mo* angelegt.

Den Zusammenhang zwischen Organisation der Bibliothek *ExternalTypes* und der entsprechenden Dateistruktur zeigt **Bild 8.1**

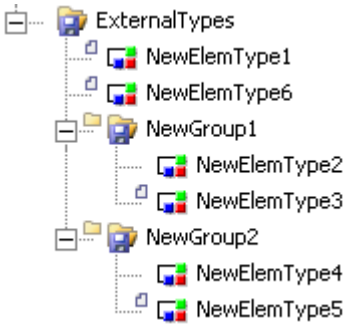
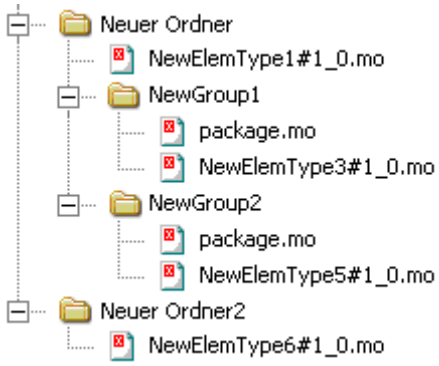
| Typhierarchie in SimulationX | Verzeichnisaufbau im Dateisystem |
|---|---|
|  |  |

**Bild 8.1:** Typ- und Dateistruktur für globale externe Element-Typen

Das Verzeichnis ..\Neuer Ordner ist das für Externe Typen eingestellte Verzeichnis (***Extras/Optionen/Verzeichnisse***). Dieses Verzeichnis muss im Dateisystem existieren und kann auch einen anderen Namen haben.

Das Verzeichnis ..\Neuer Ordner2 ist ein weiteres für Externe Typen eingestelltes Verzeichnis

NewGroup1 und NewGroup2 wurden über ***Neu/Package*** durch Rechtsklick auf den Bibliothekseintrag *ExternalTypes* erzeugt und sind deshalb Verzeichnisse. Bei der Abfrage wurde …\Neuer Ordner als Stammverzeichnis ausgewählt. In beiden Verzeichnissen wird eine Datei *package.mo* angelegt.

Der Element-Typ *NewElemTyp1* wurde durch Rechtsklick auf *ExternalTypes* und nach entsprechender Auswahl in den Ordner …\Neuer Ordner gespeichert.
Der Element-Typ *NewElemTyp6* wurde durch Rechtsklick auf *ExternalTypes* und nach entsprechender Auswahl in den Ordner …\Neuer Ordner2 gespeichert.

Die Anzeige beider Element-Typen auf gleicher Ebene erfolgt, weil die beiden Verzeichnisse als Verzeichnis für Externe Typen eingestellt wurden.

NewElemType2 und NewElemType4 sind Element-Typen, die mit dem Attribut „Speicherung innerhalb der Umgebung“ erzeugt wurden und deren Quelltext die Datei package.mo enthält.

NewElemType3 und NewElemType5 sind Element-Typen, die als Datei gespeichert wurden.

*Packages* werden standardmäßig als Verzeichnis gespeichert. Packages können jederzeit nachträglich in eine Datei überführt werden, wobei alle darunter liegenden Packages und Element-Typen in die Umgebung, sprich in die Datei übernommen/aufgenommen werden.

Es ist möglich, die Verzeichnisstruktur für die Bibliothek ExternalTypes außerhalb von SimulationX anzulegen. Dabei ist aber auf die Syntax zu achten um Probleme beim Laden zu vermeiden.

Versionen globaler Element-Typen

SimulationX unterstützt unterschiedliche Versionen für globale Element-Typen sowohl für die SimulationX-Bibliotheken als auch für externe (nutzerdefinierte) Element-Typen. Legt man eine neue Version eines Element-Typs name#1_0.mo (nur externe Typen erlaubt), so wird dafür eine neue Datei name#2_0.mo erzeugt. Innerhalb von SimulationX steht für die Bearbeitung im TypeDesigner stets nur die neueste Version zur Verfügung. Soll die neueste (oder eine andere) Version gelöscht werden, so ist das nur auf Dateiebene möglich. Andererseits ist es beim Kopieren/Versenden von externen Element-Typen als Dateien notwendig, alle Dateien name#i_0.mo zu berücksichtigen. Fehlt eine Datei aus der Serie, kann möglicherweise ein Modell, dass genau diese Version verlangt, nicht geöffnet werden.

Spezielle nutzerdefinierte Typen

**Fluid Typen**
FluidTypen werden mit dem FluidDesigner definiert. Sie werden jeweils in einer Datei Name.mo im über ***Extras/Optionen*** eingestellten Verzeichnis abgespeichert.

**SimulationX Modell**
Per Definition ist auch ein SimulationX Modell ein Typ, ähnlich einem Element-Typ, aber mit einer zusätzlichen Umgebung (Informationen, Parametern.), die für die Abarbeitung einer Simulationsrechnung notwendig ist. Demzufolge kann auch ein Modell mit dem TypDesigner bearbeitet werden. Der Zugang dafür ist im Modellexplorer auf der Seite Typen. Auf diese Weise können zusätzliche Para-

meter oder Variablen dem Modell hinzugefügt werden. (Siehe auch **Kapitel 4.2.4**).

Der Name des Modells wird in SimulationX automatisch vergeben (Modell1, Modell2). Im Laufe einer Sitzung wird für jedes geöffnete Modell der Index erhöht. Dieser Modellname wird im Modell mit gespeichert, kann aber beim Öffnen automatisch einen anderen Index bekommen, wenn der enthaltene bereits von einem geöffneten Modell verwendet wird. Aus diesem Grund sollte der Modellname nicht in Referenzen auf Variablennamen verwendet werden.

## 8.2 TypeDesigner

Der TypeDesigner ist der in SimulationX integrierte spezielle Editor für die Definition nutzerdefinierter Bibliothekselemente. Der TypeDesigner erschließt dem Anwender den kompletten Umfang der Modellierung und Programmierung, die SimulationX und die Sprache Modelica bieten.
Er unterstützt den Nutzer bei der Erstellung und Bearbeitung aller Arten von Klassen des Modelica® Standards.

- Modell (Element-Typ)
- Connector
- Record
- Block
- Funktion
- Package

In Klammern sind die SimulationX -spezifischen Begriffe angegeben. Dieses Handbuch trennt an dieser Stelle die Beschreibung in die traditionelle Arbeitsweise mit Element-Typen und Packages in der Bibliothek ExternalTypes und in die Modellierung auf der Basis der Modelica Standard Library. Letztere sowie der Sprachstandard werden in einem separaten Kapitel beschrieben. Bei der Arbeit mit externen Element-Typen wird hier stillschweigend davon ausgegangen, dass die Verwendung der SimulationX-Bibliotheken mit den Definitionen der *connections* die Basis für die Modellerstellung ist. Letztendlich ist das aber keine Begrenzung. Im Folgenden wird beschrieben, wie der TypeDesigner gestartet wird. Dann wird an Hand eines Beispiels die Definition eines externen Element-Typs beschrieben.

Neue Typen können auf unterschiedlicher Weise angelegt werden.

1. Erzeugung eines neuen Typs (siehe **Kapitel 8.2.1**)
2. Ableitung eines vorhandenen Typs (siehe **Kapitel 8.2.3**)
3. Erzeugung eines zusammengesetzten Typs (siehe **Kapitel 8.2.4**)

### 8.2.1 Aufruf und Oberfläche

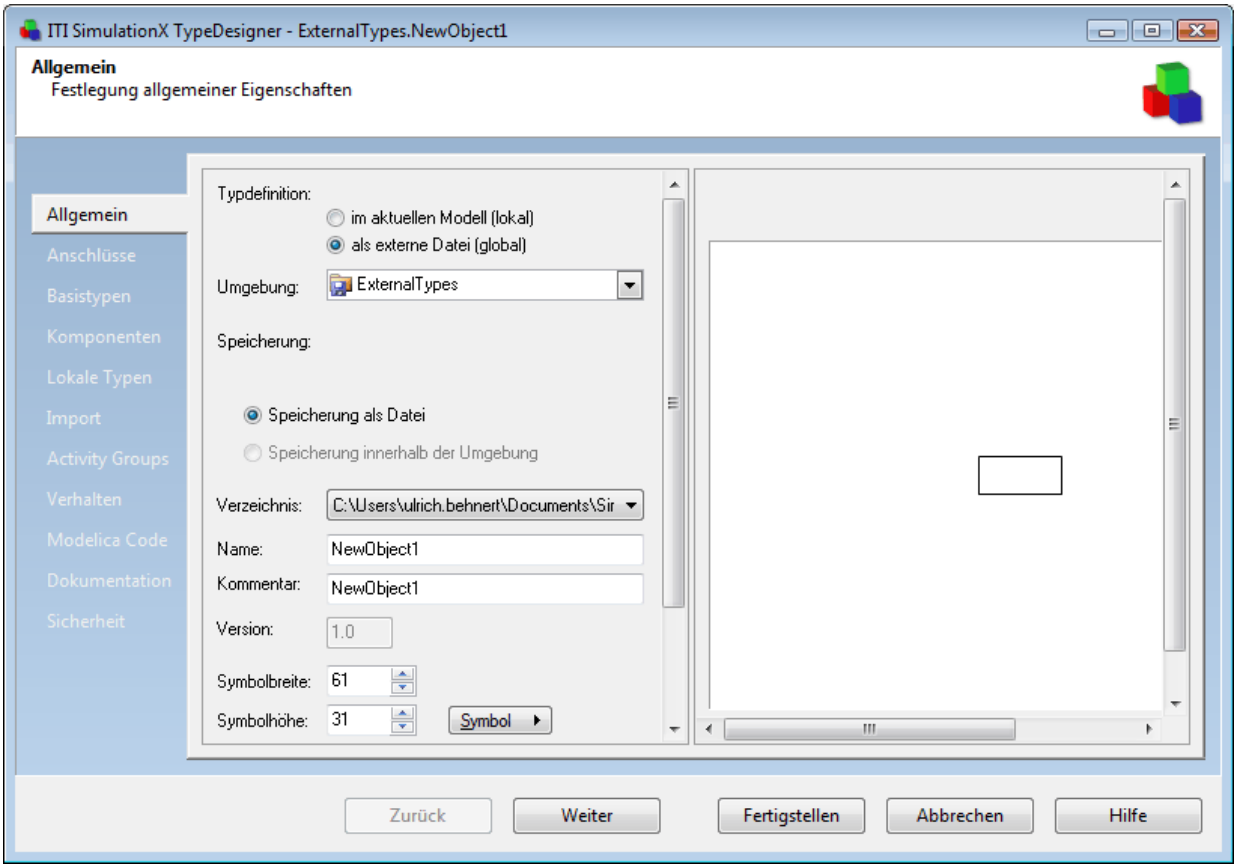



**Bild 8.2:** Ansicht TypeDesigner

Der TypeDesigner kann an allen Stellen in SimulationX gestartet werden, an denen ein neuer Typ angelegt wird oder ein bestehender bearbeitet wird (siehe auch **Kapitel 5.2.2**).

- Bibliotheksleiste: Klicken Sie in der Baumansicht mit der rechten Maustaste auf *SimulationX* bzw. ein untergeordnetes Package. Im Kontextmenü wählen Sie ***Neu/Model*** um einen **globalen** Typ zu erzeugen.
- Modellexplorer: Klicken Sie in der Baumansicht auf der Seite *Typen* mit der rechten Maustaste auf den Wurzeleintrag. Im Kontextmenü wählen Sie ***Neu/Model…*** um einen **lokalen** Typ zu erzeugen.
- Menü ***Elemente/Zusammenfassen*** oder ***Elemente/Ableiten*** , dazu müssen entweder ein oder mehrere Elemente in einem Modell markiert sein, bei der Ableitung nur eines
- Bibliotheksleiste: Klick mit der rechten Maustaste auf einen bestehenden Element-Typ (SimulationX-Bibliotheken oder *ExternalTypes*) und Auswahl der Funktion ***Neue Ableitung...*** aus dem Kontextmenü
- Modellexplorer: Klick mit der rechten Maustaste auf einen bestehenden Element-Typ (SimulationX-Bibliotheken oder *ExternalTypes*) und Auswahl der Funktion ***Neue Ableitung...*** aus dem Kontextmenü
- Klick mit der rechten Maustaste auf einen bestehenden Typ und Auswahl *Bearbeiten*, ein Doppelklick hat dieselbe Funktion

Der *TypeDesigner* ist zur Bearbeitung der textlichen Seite eines Element-Typs gedacht. Will man die Struktur bearbeiten, muss man aus dem Kontextmenü die Funktion ***Öffnen*** aufrufen.

## 8.2.2 Beschreibung der Funktionen des TypeDesigner

Die folgenden Kapitel beschreiben anhand eines Beispiels die Eigenschaftsseiten.
**Kapitel 8.2.3** beschreibt die Seite *Basistypen* bei der Ableitung eines Typs.
**Kapitel 8.2.4** beschreibt die zusätzlichen Eigenschaftsseiten beim Zusammenfassen von Komponenten.

### 8.2.2.1 Allgemein

Speicherung globaler Typen

Wurde der TypeDesigner in der Bibliotheksleiste "*SimulationX*" oder im Modell-Explorer über das Kontextmenü ***Neu/Modell*** gestartet, so ist damit der Ort, an dem die Typdefinition gespeichert wird, festgelegt. Je nach Art der Umgebung steht dann nur die Speicherung in der Umgebung oder auch als Datei zur Verfügung.

Ein Element-Typ kann nur in einem bestehenden Verzeichnis (Package) als Datei gespeichert werden.

Wurde der TypeDesigner aus der Menüsteuerung von SimulationX (Elemente Zusammenfassen, Elemente Ableitung) oder von einem Element-Typ einer (internen) SimulationX-Bibliothek gestartet (Ableitung), so muss der Ort der Speicherung noch definiert werden:

- lokale (im Modell) oder globale (Bibliotheksleiste) Speicherung
- die Umgebung; Package oder bestehender Element-Typ (Auswahl aus der Baumstruktur)

Ableitungen von internen Element-Typen können nicht in der Bibliothek des Basistypen gespeichert werden, da die SimulationX-Bibliothek nicht durch den Anwender veränderbar sind.

Sind in den Einstellungen (Menü ***Extras/Optionen/Verzeichnisse***) mehrere Verzeichnisse für externe Element-Typen eingestellt, so ist abhängig von der Wahl der Umgebung auch noch das Verzeichnis auszuwählen. Nach **Bild 8.1** wäre zwischen …\Neuer Ordner oder …\Neuer Ordner2 zu entscheiden, wenn *ExternalTypes* als Umgebung ausgewählt wurde. Bei allen anderen Umgebungen ist das Verzeichnis automatisch festgelegt.

Name und Kommentar

Geben Sie im Feld „Name“ einen eindeutigen Bezeichner für den neuen Element-Typ ein. Beachten Sie dabei die Regeln zur Namensbildung (siehe **Kapitel 10**). Der Typ kann mit einem Kommentar näher beschrieben werden.

Der Name des Element-Typen wird für die Dateinamensbildung, aber auch für die automatische Erzeugung der Elementnamen im Modell verwendet.

Darstellung

Mit der Definition des Symbols wird die Darstellung des Elementes im Modell definiert. Es gibt die Optionen Modelica-Grafik (Vektorgrafik), Bitmap (Rastergrafik) und Standardrahmen.

Vor dem Definieren einer Grafik sollte die Symbolgröße (Breite, Höhe) eingestellt werden. Diese wirkt sich auf die Größe des Standardrahmens und des Bitmaps aus. Die Vektorgrafik ist immer 60x60 Pixel groß.

Durch Klick auf die Schaltfläche ***Symbol*** öffnen Sie ein Kontextmenü, in dem Sie Menüpunkte zum Anlegen (***Neue Modelica-Grafik***, ***Neues Bitmap***) von Symbolen finden. Der Menüpunk ***Standardrahmen*** löscht eine eventuell vorhandene Grafik und erzeugt einen leeren Rahmen in Symbolgröße.

Zur Bearbeitung von Bitmap Grafiken wählen Sie den entsprechenden Menüpunkt unter der Schaltfläche ***Symbol*** (***Bitmap Öffnen***, ***Bitmap Bearbeiten***, ***Zwischenablage Importieren***). Zum Bearbeiten des Symbols sollten Sie das Editierfenster soweit vergrößern, dass Sie das komplette Symbol sehen.

Modelica-Grafiken können direkt im Fenster auf der rechten Seite bearbeitet werden. Über die Werkzeugleiste können Sie grafische Objekte erzeugen. Bitmaps können in Modelica-Grafiken integriert werden.

---

**Tipp:**

Achten Sie schon beim Entwurf der Symbole auf die spätere Lage der Anschlusspunkte. Optimale Ergebnisse erzielen Sie, wenn sich die Anschlüsse im 15-Pixel-Raster an den Rändern des Symbols befinden. Beim Laden eines Bitmaps werden Pixel der Farbe Magenta (Red=255, Green=0, Blue=255) als transparent interpretiert.

---

### 8.2.2.2 Anschlüsse

Durch die Definition von Anschlüssen (auch Ports oder Connectoren genannt) legen Sie die Schnittstelle eines Element-Typs fest. Für jeden zu erstellenden Anschluss führen Sie die folgenden Arbeitsschritte aus:

1. Wählen Sie den gewünschten Anschlusstyp (s. **Bild 8.3**) aus.

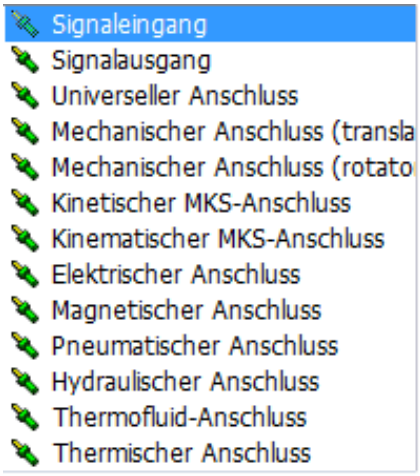



**Bild 8.3:** Auswahlbox Anschlüsse

2. Fügen Sie einen Anschluss vom gewählten Typ über die nebenstehende Schaltfläche hinzu. Der neue Anschluss wird im Vorschaufenster dargestellt.
3. Der Name und der Kommentar können geändert werden.
4. Mittels *Drag & Drop* (linke Maustaste) können Sie den Anschluss innerhalb des Vorschaufensters an die gewünschte Position am Symbolrahmen verschieben.

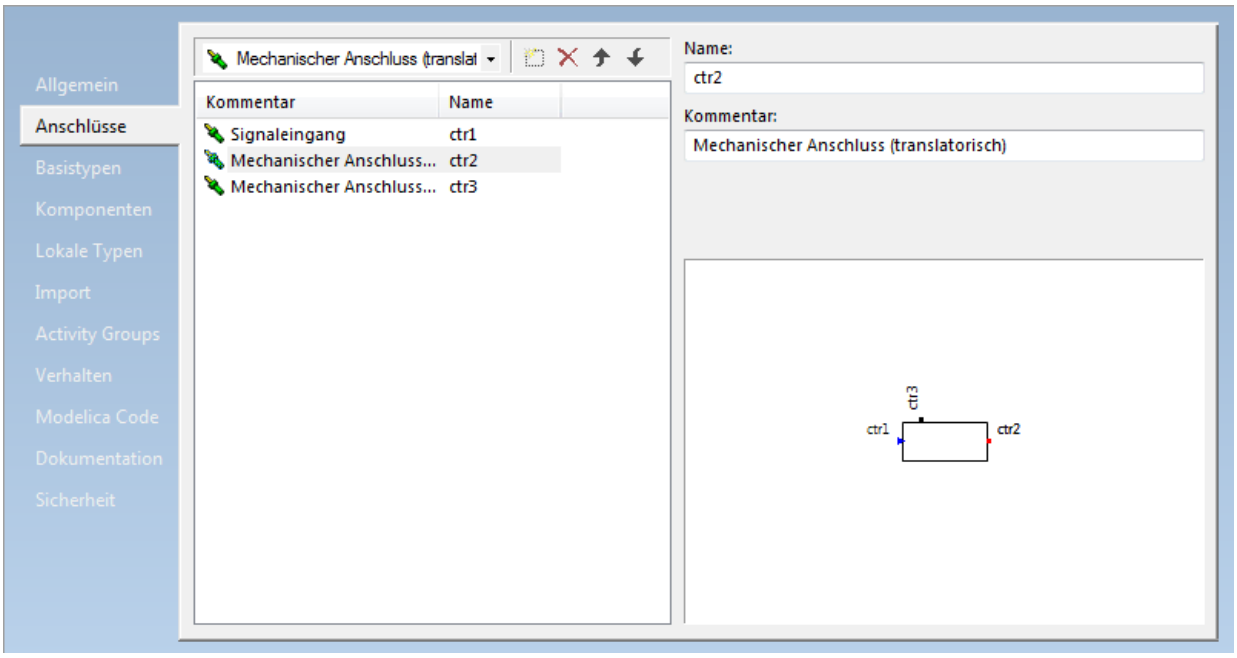



**Bild 8.4:** Dialog Anschlüsse

Die Anzahl der möglichen Anschlüsse ist lediglich durch die Größe des Symbols begrenzt. Durch Vergrößern des Elementsymbols (Breite und/oder Höhe) können Sie Raum für weitere Anschlüsse schaffen. Wenn ein Bitmap gezeichnet wurde, müssen sie vor der Größenänderung das Bitmap entfernen.

Mit den Schaltern kann die Reihenfolge der Anschlüsse verändert werden. Mit wird ein Anschluss gelöscht.

### 8.2.2.3 Basistypen

Erläuterungen dazu finden Sie im Abschnitt Ableitungen (siehe **8.2.3**).

### 8.2.2.4 Komponenten

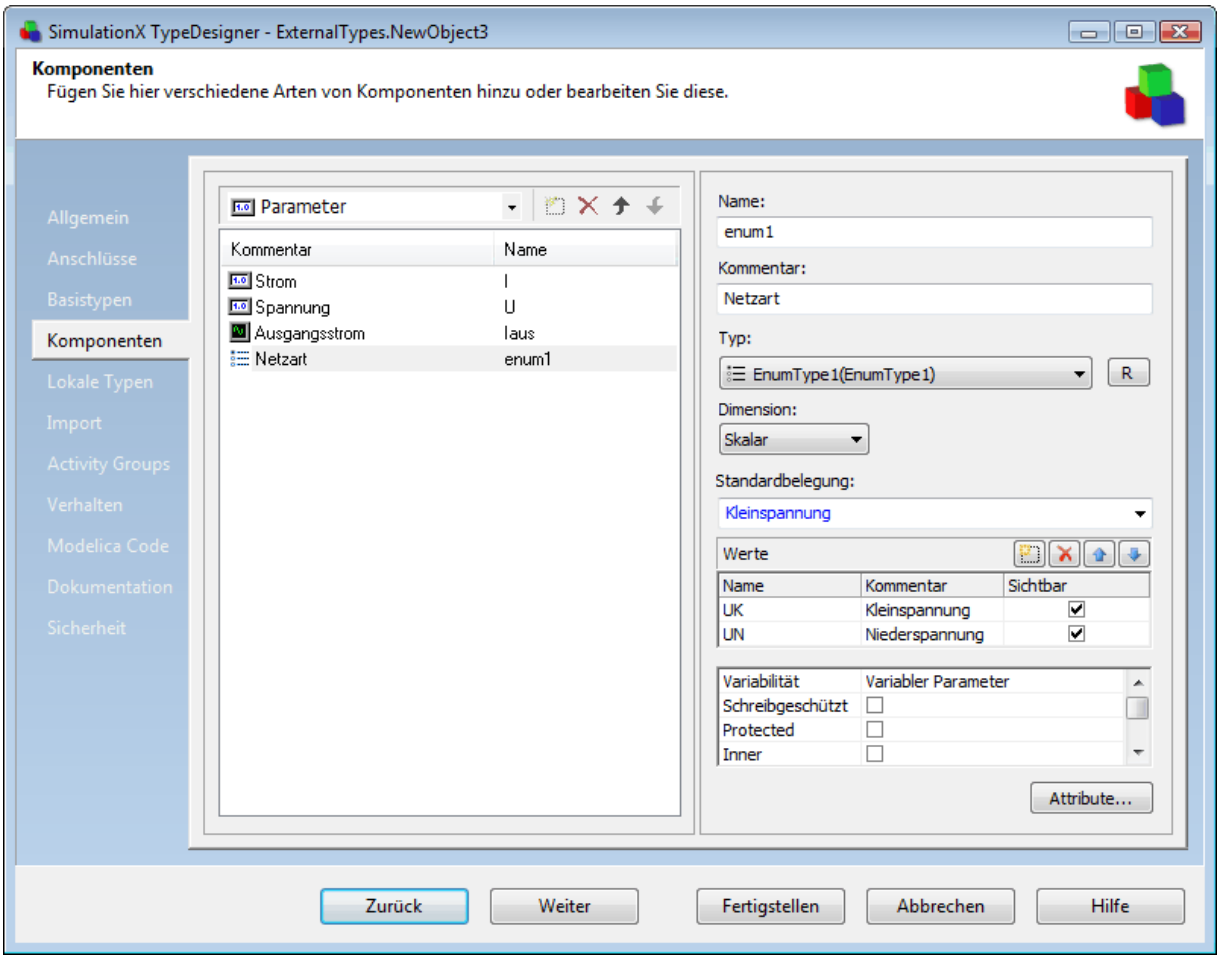



**Bild 8.5:** Dialog Komponenten

Fügen Sie hier Komponenten hinzu und bearbeiten Sie deren Eigenschaften.

Führen Sie jeweils die folgenden Arbeitsschritte aus:

1. Wählen Sie den gewünschten Komponententyp (siehe **Bild 8.6**). Eine Übersicht über die wichtigsten Eigenschaften gibt **Tabelle 8.1**.
2. Fügen Sie Komponenten des gewählten Typs über die nebenstehende Schaltfläche hinzu. Bei den Komponententypen SimBlock, Enumeration, Record oder Model werden Sie zur Auswahl des konkreten Typen aufgefordert.

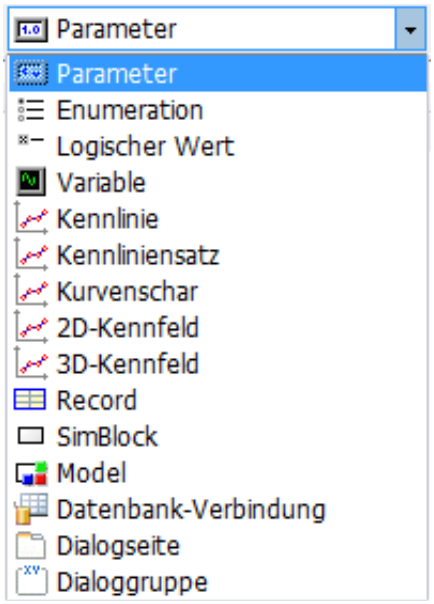



**Bild 8.6:** Auswahlbox Komponenten

3. Verschiedene Eigenschaften wie Name und Kommentar können bearbeitet werden.
4. In Abhängigkeit von der gewählten Komponente können Sie weitere Eigenschaften (z. B. Typ, Physikalische Größe, Dimension oder Standardbelegung) angeben.

**Typ:** Wichtige Einstellungen sind Real, Integer, Boolean. Alle anderen Einstellungen können auch direkt in der Auswahlbox Komponenten ausgewählt werden

**Physikalische Größe:** Für die Komponente werden die Attribute *quantity* und *displayUnit* gesetzt, die für die angezeigte Einheit im Eigenschaftsdialog und für die Standardbelegung verwendet werden.

**Dimension:** Skalar oder mehrdimensionaler Wert (Vektor...)

**Weitere Einstellungen in der unteren Tabelle:**
Die möglichen Einstellungen hängen vom eingestellten Typ der Komponente ab. Für Parameter ist z. B. die Einstellung als Startwert interessant.
Markieren Sie eine Komponente als „schreibgeschützt“, wenn der Anwender die Komponente zwar sehen aber nicht verändern soll, beispielsweise fest vorgegebene Verzeichnisse oder Ergebnisse aus anderen Programmen.
Markieren Sie z. B. eine Variable als „protected“ (geschützt), wenn sie für den Anwender des Typs im Eigenschaftsdialog unsichtbar bleiben soll, beispielsweise bei Variablen für Zwischenergebnisse, welche im Algorithmus benötigt werden.
Als Diskret markierte Komponenten ändern sich nur innerhalb von Events.

**Deklarationsgleichung (bei Variablen):**
Das Verwenden einer Deklarationsgleichung wird hier nicht empfohlen. Prinzipiell hat diese das gleiche Ergebnis wie die Gleichung im Equationteil der Verhaltensbeschreibung. Sollte dort aber auch eine Gleichung für diese Variable

aufgeschrieben werden, ist für diese Variable eine Gleichung zu viel vorhanden. Beim Start der Simulation wird das auch im Ergebnisfenster über eine Meldung angezeigt. Die Analyse wird erschwert, weil die Ursache auf zwei verschiedene Stellen verteilt ist.

**Startwert (bei Variablen):**
Wenn eine Variable Zustandsgröße wird, so sollte der Startwert beeinflussbar sein. Im Eingabefeld für den Startwert der Variable sind alle vorhandenen Parameter mit diesem speziellen Attribut sichtbar und der entsprechende Startwert kann ausgewählt werden.
Dazu ist eine Komponente *Parameter* anzulegen und in der Tabelle der weiteren Einstellungen die Auswahl *Startwert* zu aktivieren. In der Liste der angelegten Komponenten bekommt der Parameter ein rotes (statt blau) Symbol. Der Parameter sollte die gleiche physikalische Größe haben wie die Variable, für die er vorgesehen ist. Im Eigenschaftsdialog in SimulationX wird dieser Parameter dann mit dem speziellen Symbol für die Fixierung (Pin) angezeigt.

**Attribute:**
Hier können die Einstellungen für die Attribute vordefinierter Typen vorgenommen werden. Die Beschreibung erfolgt im **Kapitel 10.4.3.2**. Bei der Parametrierung von Komponenten im Modell können diese geändert werden (**Kapitel 5.3.4**).

| Komponente | Verwendung |
|---|---|
| Parameter | Eingabe von Zahlenwerten und mathematischen Ausdrücken vom Typ Real, Integer, Boolean oder String. (siehe **Tabelle 8.2**)<br>Parameter vom Typ Real können als Startwert für Zustände verwendet werden. |
| Enumeration | Auswahl einer Option aus einer Aufzählung - erfordert die Auswahl oder Erzeugen eines Enumerationtyp (siehe Lokale Typen/**Enumeration**) |
| Logischer Wert | Auswahl einer Option |
| Variable | Ergebnisgröße (siehe **Tabelle 8.2**) |
| Kennlinie | Eingabe einer Kennlinie f(x) |
| Kennliniensatz | Eingabe einer Gruppe von Kennlinien $f_i(x)$ |
| Kurvenschar | Eingabe einer Kurven-Schar |
| 2D-Kennfeld | Eingabe eines 2D-Kennfeldes |
| 3D-Kennfeld | Eingabe eines 3D-Kennfeldes |
| Datenbankverbindung | Verknüpfen von Parametern des Element-Typs mit einer Datenbank |

| | |
|---|---|
| SimBlock | Wieder verwendbare Berechnungsvorschrift – erfordert Auswahl oder Erzeugen eines Simblocktyp (siehe Lokale Typen/**SimBlock**) |
| Model | Erweiterung eines Compounds (siehe **Kapitel 8.2.4**) |
| Record | Menge zusammengehöriger Daten mit unterschiedlichen Typen - erfordert die Auswahl eines Recordtypen (siehe Lokale Typen/**Record**) |
| Dialoggruppe | Einrichten einer Gruppe zur Strukturierung des Eigenschaftsdialoges, das Ergebnis sind nutzerdefinierte Blöcke im Eigenschaftsdialog. |
| Dialogseite | Einrichten einer Seite zur Strukturierung des Eigenschaftsdialoges, das Ergebnis sind nutzerdefinierte Registerkarten im Eigenschaftsdialog. |

**Tabelle 8.1:** Komponententypen

Die Eigenschaft der Variabilität bestimmt, ob eine Komponente als Parameter oder als Variable verwendet wird. Die folgende Tabelle zeigt die möglichen Alternativen und Auswirkungen.

| Variabilität | Darstellung im Eigenschaftsdialog | Erlaubte Ausdrücke |
|---|---|---|
| Variable | Ergebnisgröße (protokollierbar) | keine (nur Definition eines Startwerts erlaubt) |
| Variabler Parameter | Parameter | während der Simulation nicht konstante Ausdrücke |
| Konstanter Parameter | Parameter | während der Simulation konstante Ausdrücke |
| Konstante | Parameter (deaktiviert) | während der Simulation konstante Ausdrücke, im Eigenschaftsdialog nicht änderbar |

**Tabelle 8.2:** Variabilität

Parameter werden im Eigenschaftsdialog von Elementen angezeigt und erlauben die Eingabe von Ausdrücken. Variablen können als Ergebnisgrößen protokolliert werden.

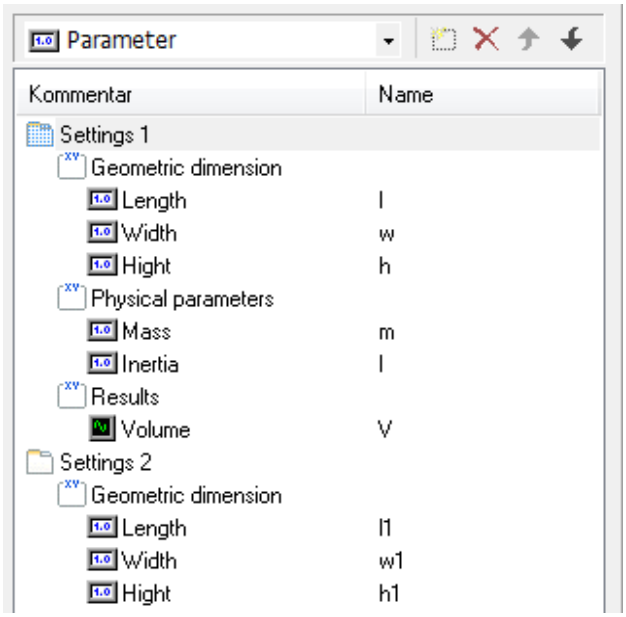
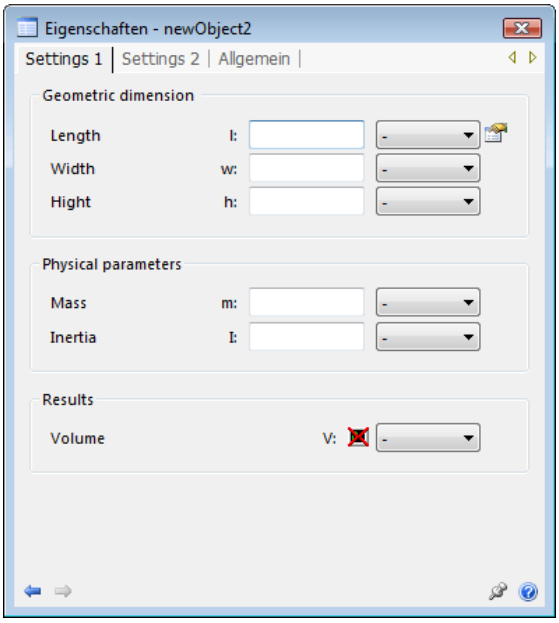
**Einfügen von Dialogseiten und -gruppen**

Wählen Sie zuerst Dialogseite bzw. Dialoggruppe als Komponententyp und fügen Sie über nebenstehende Schaltfläche die Dialogseite/-gruppe hinzu. Auf der rechten Seite können Sie den Namen vergeben.

Mit Hilfe der Pfeiltasten können Sie die Position des Eintrags verändern. Die

untergeordneten Komponenten werden entsprechend eingerückt dargestellt.

Sollen Komponenten keiner Gruppe oder Seite untergeordnet sein, dann verschieben Sie diese Komponenten vor die erste Gruppe bzw. Seite. Für diese Komponenten werden automatisch Dialogseiten erzeugt. Leere Dialogseiten und -gruppen werden nicht im Typ gespeichert.

| Definition im TypeDesigner | Eigenschaftsdialog |
|---|---|
|  |  |





**Hinweis:**
Dialogseiten und -gruppen haben nur auf die Darstellung des Eigenschaftsdialoges Einfluss. Name bzw. Identifikator eines Symbols wird durch die Gruppenzugehörigkeit nicht verändert.

### 8.2.2.5 Lokale Typen

Lokale Typen werden im aktuell bearbeiteten Typ gespeichert. Es stehen alle Arten von Modelica® Typen zur Auswahl:

- Enumeration (Aufzählungstyp)
- Funktion (Externe-/ Interne Funktion, Partielle Ableitung)
- Record
- SimBlock
- Model (Element Typ)
- Connector (Anschlusstyp, nicht notwendig bei ausschließlicher Nutzung der SimulationX-Bibliotheken)
- Typdefinition
- Package

Das Definieren lokaler Typen an dieser Stelle ist zwar möglich und wird durch die Philosophie der Modellbeschreibungssprache Modelica unterstützt. Es entsteht eine Konstruktion, die im einleitenden Abschnitt dieses Kapitels unter dem Stichwort **Typ in Typ** beschrieben wurde. Für eine übersichtlichere Modellierung erscheint es jedoch sinnvoll, solche Strukturen nur in Ausnahmefällen zu verwenden. Eine Ausnahme bilden Aufzählungstypen (Enumerations), die meist an einen bestimmten Typen gebunden sind und hier definiert werden müssen.

Um einen neuen lokalen Typ zu erzeugen, wählen Sie zuerst einen der oben aufgezählten Typen aus und fügen ihn über die nebenstehende Schaltfläche hinzu. Auf der linken Seite können *Name*, *Kommentar* und (je nach gewähltem Typ) weitere Eigenschaften bearbeitet werden. Für die Bearbeitung von strukturierten Typen (Funktionen, SimBlöcke, Records, Elementtypen, Connectoren Packages) wird über die Schaltfläche ***Bearbeiten…*** eine Instanz des TypeDesigners geöffnet.

**Enumeration**
Der Typ einer Enumeration definiert die Menge der Alternativen und den Standardwert. Wenn Sie eine Komponente mit dem Enumerationtyp erstellen, dann legen Sie hier die Sichtbarkeit der Alternativen und einen Standardwert fest.

**Funktion** (Definition einer zustandsfreien Berechnungsvorschrift)
Funktionen sind wieder verwendbare Module mit einem Algorithmus. Die Argumente (Eingabe- und Ausgabeparameter) bilden die Schnittstelle nach außen. Eine Funktion kann eigene Komponenten und lokale Typen beinhalten. Externe Funktionen berechnen die Ausgabeparameter über einen Funktionsaufruf in einer DLL. Interne Funktionen besitzen einen Algorithmus für die Berechnung. (siehe auch **Kapitel 8.2.5**)

**Record**
Records fassen eine Menge von zusammengehörigen Daten unterschiedlichen Typs zusammen. Dies vereinfacht zum Beispiel die Schreibweise bei Zuweisungen.

**SimBlock** (Definition einer zustandsbehafteten Berechnungsvorschrift)
SimBlock-Typen sind wiederverwendbare Module mit eigenen Algorithmen und Gleichungssektionen. Die Argumente (Eingabe- und Ausgabeparameter) des SimBlocks bilden die Schnittstelle nach außen.

**Model** (Element Typ)
Model Typen bilden die Vorlage für die Elemente eines Simulationsmodells. Sie können eine zustandsbehaftete Verhaltensbeschreibung definieren und die Schnittstelle wird durch Anschlüsse definiert.

**Connector**
Connectortypen erlauben die Definition eigener Anschlusstypen.

**Typdefinition**
Typdefinitionen erlauben die Definition abgeleiteter Typen in verkürzter Schreibweise, wenn die Struktur der Ableitung unverändert bleibt. Die Eigenschaften der geerbten Komponenten aus der Basisklasse können jedoch verändert werden. Wenn Sie die Basisklasse ausgewählt haben, dann können Sie die Modifikationen vornehmen.

**Package**
Ein Package fasst Typ- und Konstantendefinitionen z. B. eines Themas zusammen und macht den Quelltext damit leichter lesbar. Package vermeiden zusätzlich Namenskonflikte. Beispiel: Pneumatics.Valves vs. Hydraulics.Valves.

### 8.2.2.6 Import

Importe machen die öffentlichen Definitionen eines Package im aktuellen Typ verfügbar und erlauben damit eine kürzere Schreibweise. Zusätzlich wird durch die Import-Anweisung die Abhängigkeit des Typs vom importierten Package explizit festgelegt.

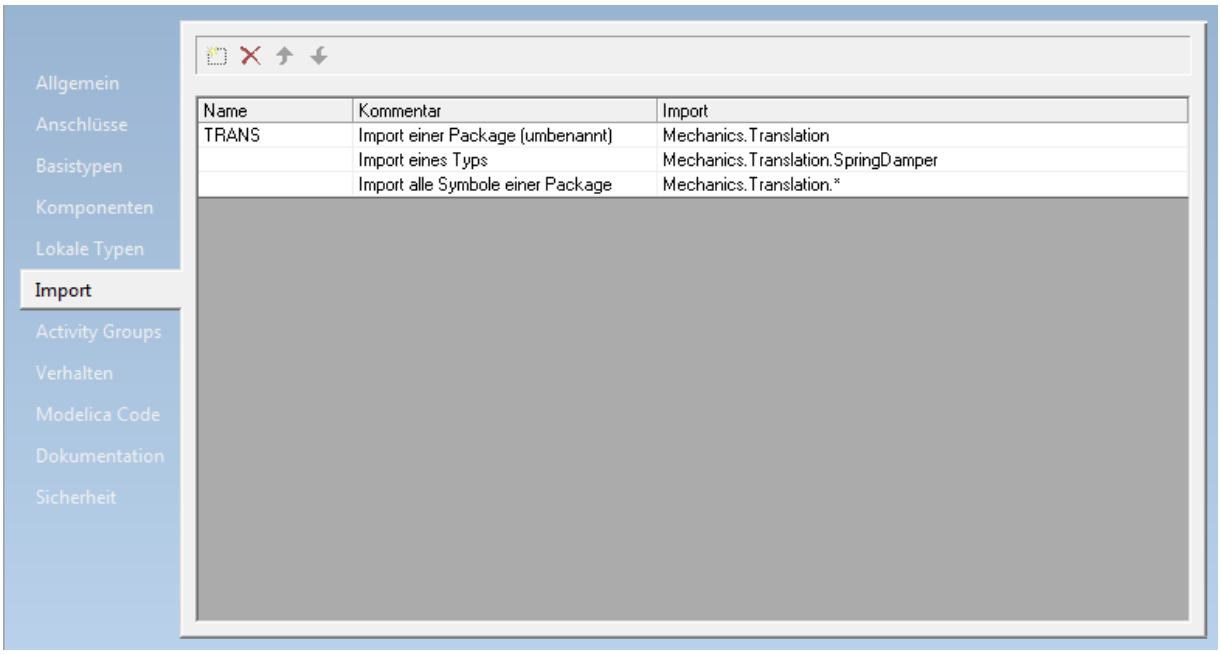



**Bild 8.7:** Auswahl Import

Mit der Schaltfläche können Import-Anweisungen hinzugefügt werden. Über einen Dialog können Sie die zu importierende Definition auswählen.

Sie können den Inhalt des Package auf drei verschiedene Arten importieren. Die Unterschiede verdeutlicht die folgende Tabelle.

| Import | Schreibweise für *Mechanics.Translation.Spring* | Umbenennen möglich ? |
|---|---|---|
| *Mechanics.Translation* | *Translation.Spring* | ja |
| *Mechanics.Translation.Spring* | *Spring* | ja |
| *Mechanics.Translation.** (Importiere alle Definitionen) | *Spring* | nein |

**Tabelle 8.3:** Import

Die ersten beiden Importarten erlauben das Umbenennen der importierten Definition innerhalb des aktuellen Typs. Die Definition kann dann innerhalb des Typs mit dem neuen Namen verwendet werden, damit können z. B. Namenskonflikte mit bereits vorhandenen Definitionen beseitigt werden.

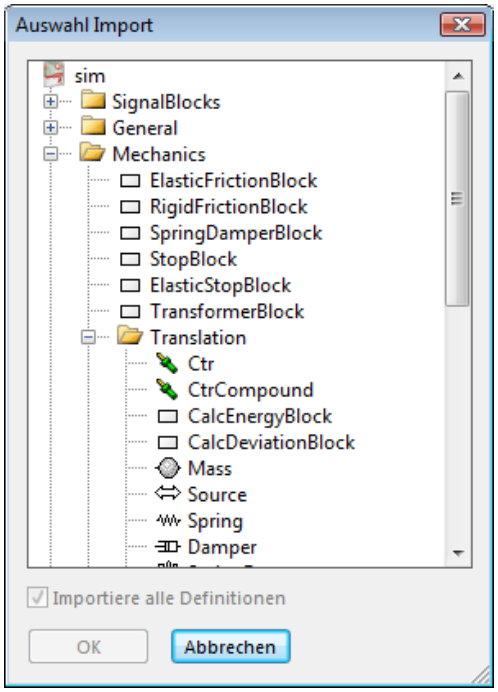



**Bild 8.8:** Auswahl Import

Wenn Sie die Option ***Importiere alle Definitionen*** aktivieren, dann werden alle Definitionen der ausgewählten Package importiert. Sie können dies am Stern am Ende des Imports in der Import Seite erkennen (siehe **Bild 8.7**).

Im Quelltext des Typs werden alle Namen von importierten Definitionen in kurze Schreibweise umgewandelt. Wenn ein Import gelöscht wird, dann werden wieder die vollständigen Identifikatoren verwendet.

### 8.2.2.7 Activity Groups

Activity Groups dienen dazu, dynamische Eigenschaftsdialoge zu erzeugen. Dynamisch heißt, dass ihr Inhalt von der aktuellen Parameterbelegung abhängt.

Activity Groups ermöglichen das Festlegen von Bedingungen, wann gewisse Parameter sichtbar oder unsichtbar sind.
Um mit Activity Groups arbeiten zu können, muss zunächst die Checkbox ***„Activity Groups unterstützen"*** aktiviert werden.
Mit dem Schalter erzeugen Sie eine neue Gruppe und können diese bearbeiten. Es wird eine Liste von Tupeln {Name, Bedingung, Liste der zugehörigen Parameter} verwaltet. Jedes Tupel bildet **eine** Activity Group. In der Parameterliste sind die Namen der Parameter aufgeführt, die von dieser Activity Group beeinflusst werden.
Falls eine Bedingung den Wert „false" zurückgibt, sind die zugehörigen Parameter im Eigenschaftsdialog unsichtbar und liefern den Wert „0". Wenn ein Parameter in mehreren Activity Groups vorkommt, werden die einzelnen Bedingungen mit ODER verknüpft, d. h. wenn eine Bedingung wahr ist, dann ist dieser Parameter sichtbar.

**Beispiel:**
Es soll ein Element-Typ erstellt werden, der einen Querschnitt und einen Durchfluss als Auswahlkriterium für verschiedene Ansätze nutzt. Es gibt zwei Enumerations, die die Art des jeweiligen Ansatzes auswählen.

| | Enumeration | Ansatz |
|---|---|---|
| Querschnitt | A = {a, b } | Rechteck |
| | A={ d } | Kreis |
| Durchfluss | Q = { p1 } | Ansatz1 |
| | Q = { p21, p22 } | Ansatz2 |
| | Q = { p31,p32,p33 } | Ansatz3 |

**Tabelle 8.4:** Übersicht Parameter

Der Element-Typ enthält dann **einen** Parameter vom Typ ActivityGroups und dessen Wert besteht aus folgenden Einträgen:

| Name | Bedingung | Liste der Parameter |
|---|---|---|
| Rechteck | A==1 | a,b |
| Kreis | A==2 | d |
| Ansatz1 | Q==1 | p1 |
| Ansatz2 | Q==2 | p21,p22 |
| Ansatz3 | Q==3 | p31,p32,p33 |

**Tabelle 8.5:** Bedingungen für A und Q

Umsetzung im TypeDesigner:

1. Erzeugen Sie je eine Komponente für Querschnitt und Durchfluss sowie die dazugehörigen Parameter.
   Dazu wählen Sie zunächst eine Komponente vom Typ „Enumeration“ aus und fügen diese hinzu.

   Danach vergeben Sie den entsprechenden Namen (im Beispiel „A“) und den Kommentar („Querschnitt“). Als nächstes werden die Werte (im vorliegenden Fall „Rechteck“ und „Kreis“) hinzugefügt. Wiederholen Sie diese Anweisungen für die Enumeration Q („Durchfluss) mit den Werten „Ansatz1“, „Ansatz2“ und „Ansatz3“. Anschließend vereinbaren Sie alle weiteren Parameter.

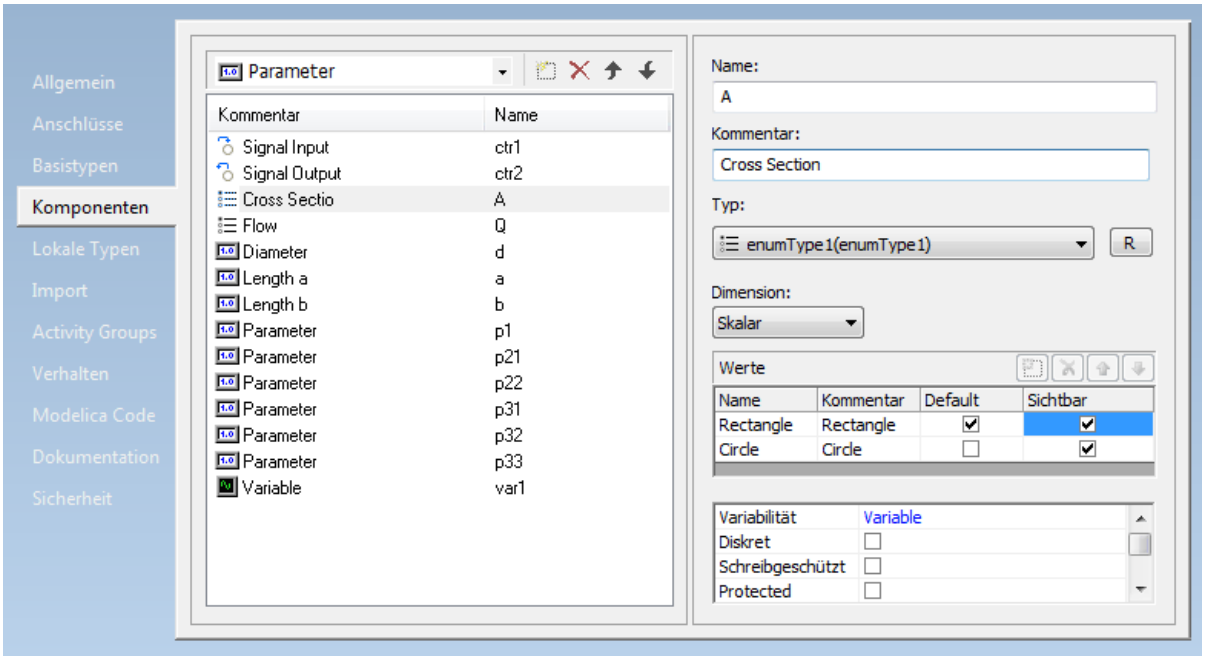



**Bild 8.9:** Dialog Komponenten

   Die Reihenfolge der Einträge entspricht der Darstellung im Eigenschaftsdialog des neuen Element-Typs und sollte für das Beispiel, wie in **Bild 8.9** dargestellt, übernommen werden.

2. ActivityGroup anlegen
   Zuerst aktivieren Sie die Activity Groups über die Checkbox *“**Activity Groups unterstützen**”* (oben links). Danach betätigen Sie den Button , vereinbaren für die Enumerations die Bedingungen und tragen die Parameter ein.
   **Beachte:**
   - Inaktive Parameter liefern 0 zurück
   - Enumerations werden über einen Integer-Wert (Bedingung) abgefragt
3. Test der Bedingungen der Activity Group:
   Dazu erstellen Sie ein Element des neuen Typs in der Strukturansicht und öffnen den Eigenschaftendialog. Unter ***Parameter1*** finden Sie die vorher definierten Auswahlkriterien und können diese testen.

Um eine Activity Group zu löschen, markieren Sie diese und nutzen den ***Löschen***-Button . Die Reihenfolge der Activity Groups kann mit den Schaltern geändert werden.

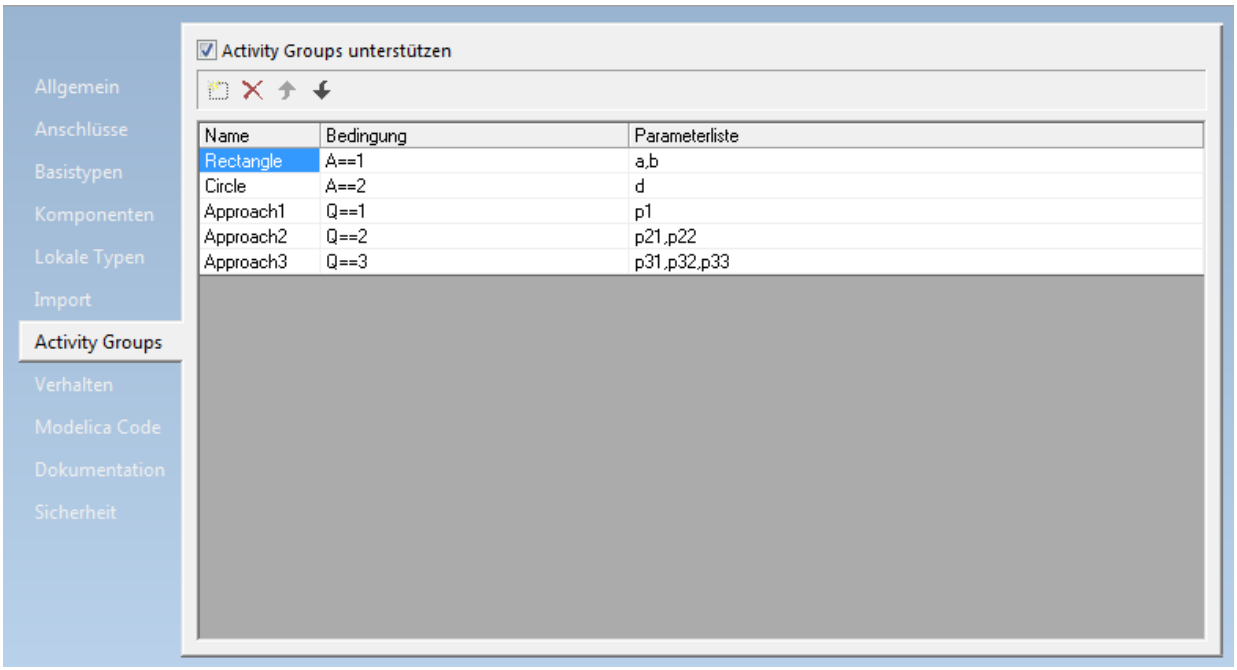



**Bild 8.10:** Activity Groups

Die Testergebnisse sind in (**Bild 8.11**) beispielhaft für Rechteck und Ansatz1, sowie für Kreis und Ansatz3 aufgezeigt.

Bei mehr als 10 Parametern wird ein neuer Dialog genutzt. Im vorliegenden Fall wird bei Ansatz3 der Parameter `p33` im Dialog ***Paramter2*** dargestellt.

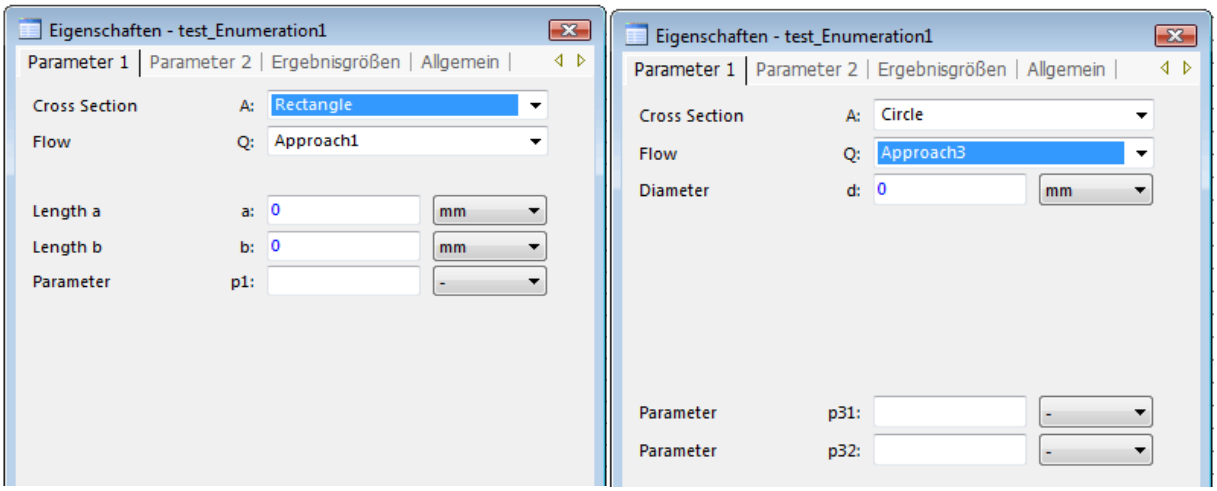



**Bild 8.11:** Parameterdialoge für unterschiedliche Bedingungen

### 8.2.2.8 Verhalten

Hier kann durch Algorithmen und Gleichungen (engl. Equations) das Verhalten des Elementtyps beschrieben werden. Jeder Typ besitzt wenigstens eine Gleichungs- oder Algorithmensektion. Über ein Auswahlfeld können Sie weitere

Algorithmen- oder Gleichungssektionen hinzufügen analog der Erstellung von Komponenten.
Während der Simulation werden die Sektionen in der festgelegten Reihenfolge abgearbeitet. Die Reihenfolge der Sektionen können Sie mit den Pfeiltasten oder per Drag & Drop der Reiter festlegen.
Auf der linken Seite sind in einer Baumstruktur die Namen verfügbarer Komponenten und lokaler Funktionstypen zur Information dargestellt. Außerdem sind die vorhanden Alternativen an Enumerations sichtbar. Auf der rechten Seite ist die aktuell ausgewählte Sektion zum Bearbeiten geöffnet.
Geben Sie hier eine Anweisungsfolge ein, die das Verhalten des neuen Element-Typs beschreibt. Zur Verfügung stehen neben Anweisungen zur Wertzuweisung und arithmetischen Operatoren auch die in Programmiersprachen gebräuchlichen Steuerungskonstrukte. Zugriff auf die Anschlüsse, Parameter und Ergebnisgrößen erhalten Sie über deren Namen. Per *Drag & Drop* können Sie die Symbole aus der Liste in die aktuell ausgewählte Sektion verschieben. Über die Reiter am unteren Rand wechseln Sie zwischen den Sektionen.

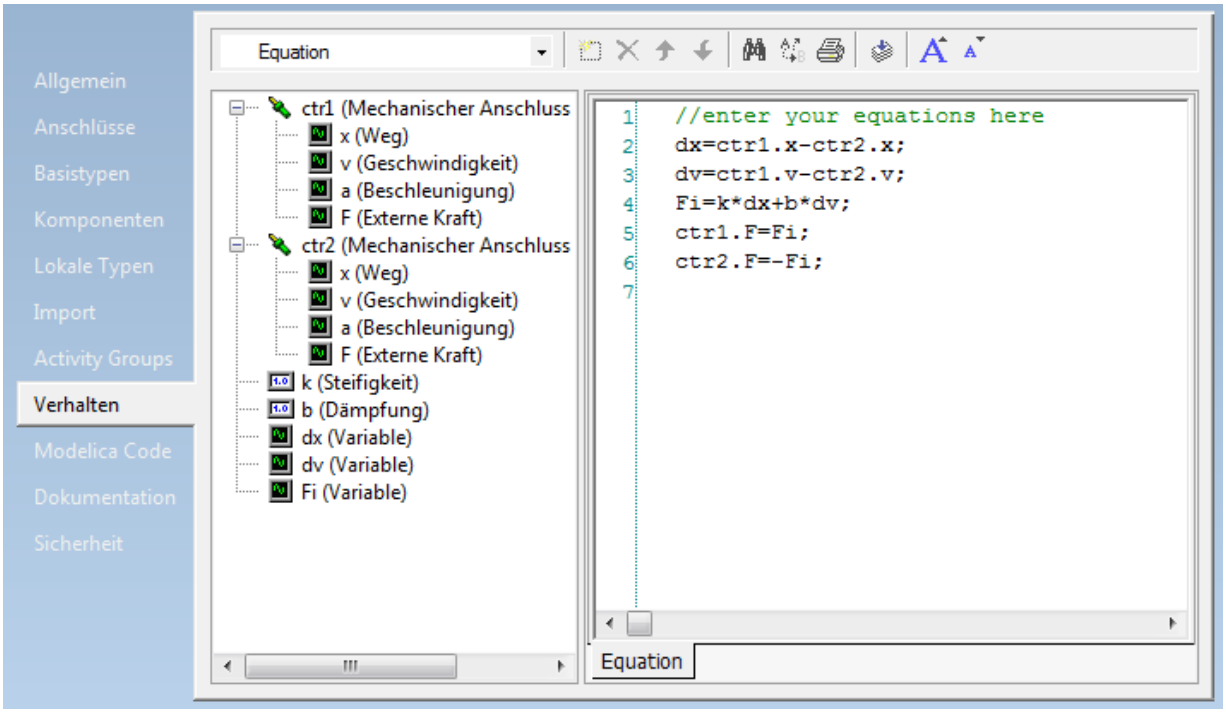



**Bild 8.12:** Verhaltensbeschreibung mit Algorithmen und Gleichungen (Equation)

Wenn möglich, sollten Sie das Verhalten mit **Gleichungen** beschreiben. Jede Gleichung besitzt eine linke und eine rechte Seite, die durch ein "=" miteinander verbunden sind. Die Gleichung ist mit einem ";" abzuschließen. Nur Gleichungen können durch die *symbolische Analyse* optimiert werden. **Bild 8.12** zeigt ein Beispiel hierfür.
Mit einem **Algorithmus** kann das Verhalten Ihres Element-Typs durch Zuweisungen ":=" beschrieben werden. Jede Zuweisung ist mit einem ";" abzuschließen. Die Reihenfolge der Abarbeitung wird durch den Algorithmus fest vorgegeben.

Zur Erleichterung der Arbeit bei der Notation von Algorithmen und Gleichungen sind folgende Schalter vorhanden:

Suchen von Variablen, Parametern (Textsuche)

Suchen und Ersetzen (Textsuche)

Syntaxcheck

Das **Kapitel 10** enthält eine weitere Beschreibung der Nutzung von Variablen in einem Algorithmus und in Gleichungen.
Zusätzlich zur Beschreibung des Typverhaltens in der Simulation können Sie spezielle Sektionen zur Anfangswertberechnung einfügen. Zur Verfügung stehen mit "Initial Algorithm" und "Initial Equation" ebenfalls beide Möglichkeiten der Verhaltensbeschreibung. Die „Initial“-Sektionen werden nur bei der Berechnung des Anfangszustands des Modells ausgewertet. Die Verwendung von Gleichungssystemen oder Algorithmen zur Initialisierung ist nur notwendig, wenn Anfangswerte nicht explizit berechnet werden können.
Eine weitere Möglichkeit, das Verhalten eines Elements zu beschreiben, ist die Nutzung von **Statechart-Diagrammen** (siehe **Kapitel 8.4**).

### 8.2.2.9 Modelica Code

Auf dieser Seite kann der Modelica® Quellcode überprüft und bearbeitet werden. Um versehentliche Änderungen zu vermeiden, ist standardmäßig nur der Ansichtsmodus aktiviert. Der Quelltext kann noch nicht verändert werden, sondern es muss erst in den Bearbeitungsmodus umgeschaltet werden ( ).

Bei Rückkehr in den Ansichtsmodus können alle Änderungen entweder übernommen oder rückgängig gemacht werden. Fehler oder Warnungen bei der Analyse des Codes werden unter dem Bearbeitungsfenster angezeigt und können nachträglich berichtigt werden. Nach erfolgreicher Zuweisung des neuen Quelltexts werden alle anderen Seiten im Typ Designer entsprechend aktualisiert.

Erlaubt das Umschalten zwischen Ansichts- und Bearbeitungsmodus des Modelica® Quellcodes.

Macht die letzte Änderung im Quelltext rückgängig.

Stellt rückgängig gemachte Änderung wieder her.

Startet Text Suche.

Startet Suchen und Ersetzen von Text.

Führt Modelica® Syntax-Überprüfung durch.

Vergrößerung der Schriftgröße.

Verkleinerung der Schriftgröße.

Zur besseren Übersicht können Sie einzelne Abschnitte des Quelltexts ein- und ausblenden. Durch Klick auf das Minus-Symbol auf der linken Seite wird der entsprechende Text ausgeblendet. Durch Klick auf ein Plus-Zeichen machen Sie ihn wieder sichtbar.

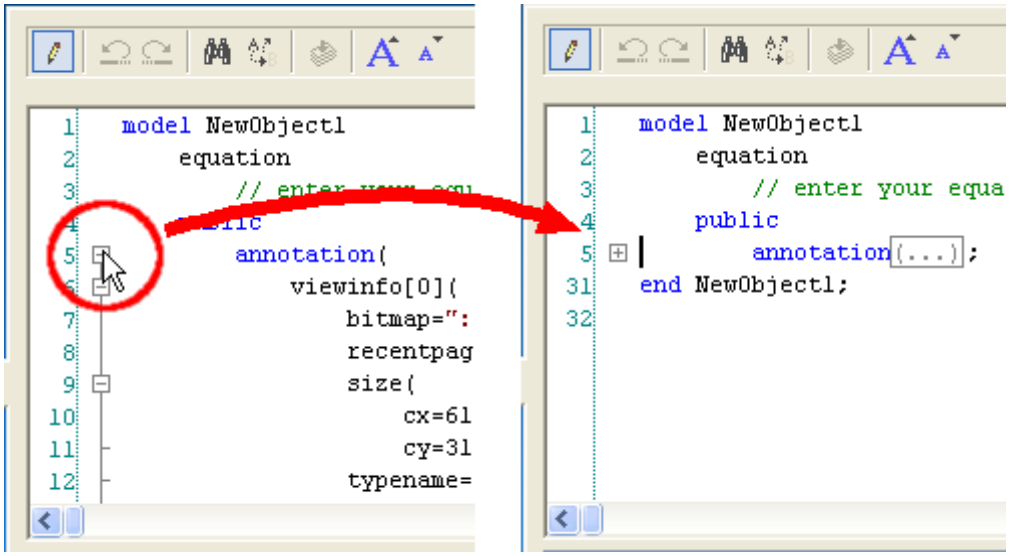



**Bild 8.13:** Ausblenden eines Quellcode Abschnitts

### 8.2.2.10 Dokumentation

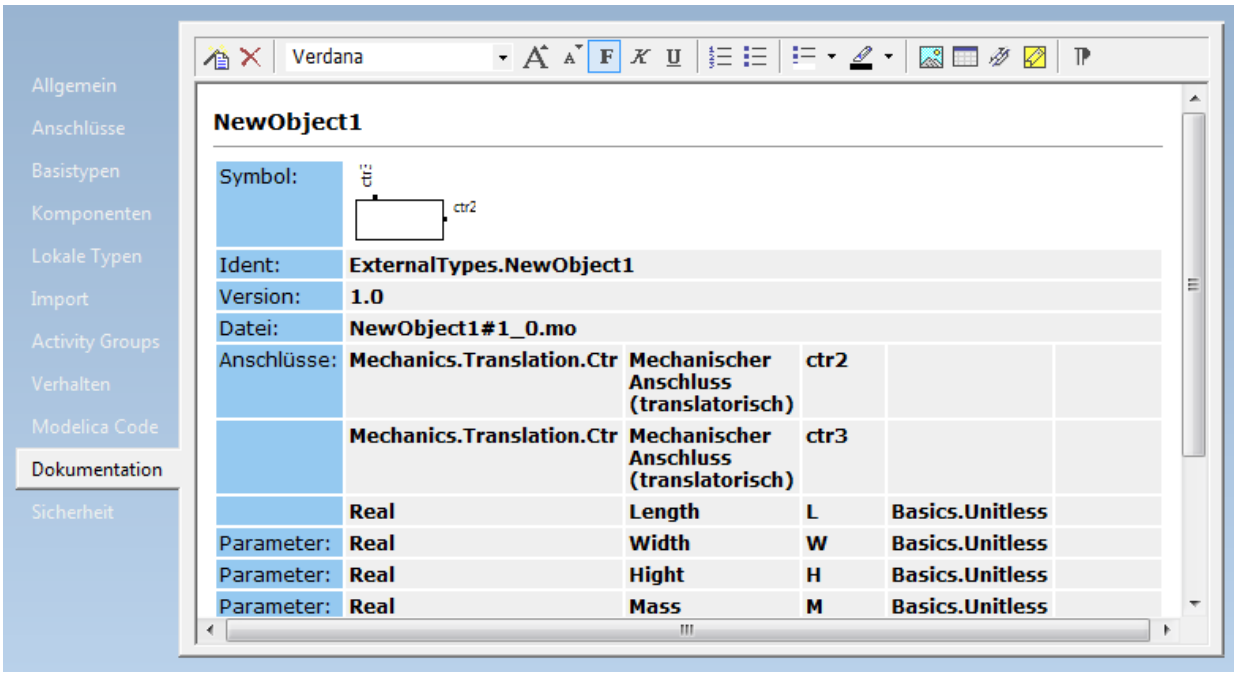



**Bild 8.14:** Ansicht Dokumentation

Um die Nutzung und Wiederverwendbarkeit eines neuen Typs zu vereinfachen, sollten Sie Ihre Arbeit dokumentieren. Die Dokumentation erfolgt in Form einer HTML-Datei.

Sie können ein Gerüst (siehe **Bild 8.14**) mit allen wesentlichen Bestandteilen zur Dokumentation des neuen Typen erzeugen lassen.

Ihre Aufgabe besteht darin, das entstandene Gerüst mit all jenen Informationen zu vervollständigen, die eine sachgerechte Nutzung ermöglichen. Dazu stehen Ihnen umfangreiche Eingabe- und Editiermöglichkeiten zur Verfügung.
Die erstellte Dokumentation ist gleichzeitig die Hilfe zu dem entsprechenden Typ.

Für die Gestaltung steht eine Reihe von Werkzeugen zur Verfügung. In **Bild 8.15** sind diese erklärt.

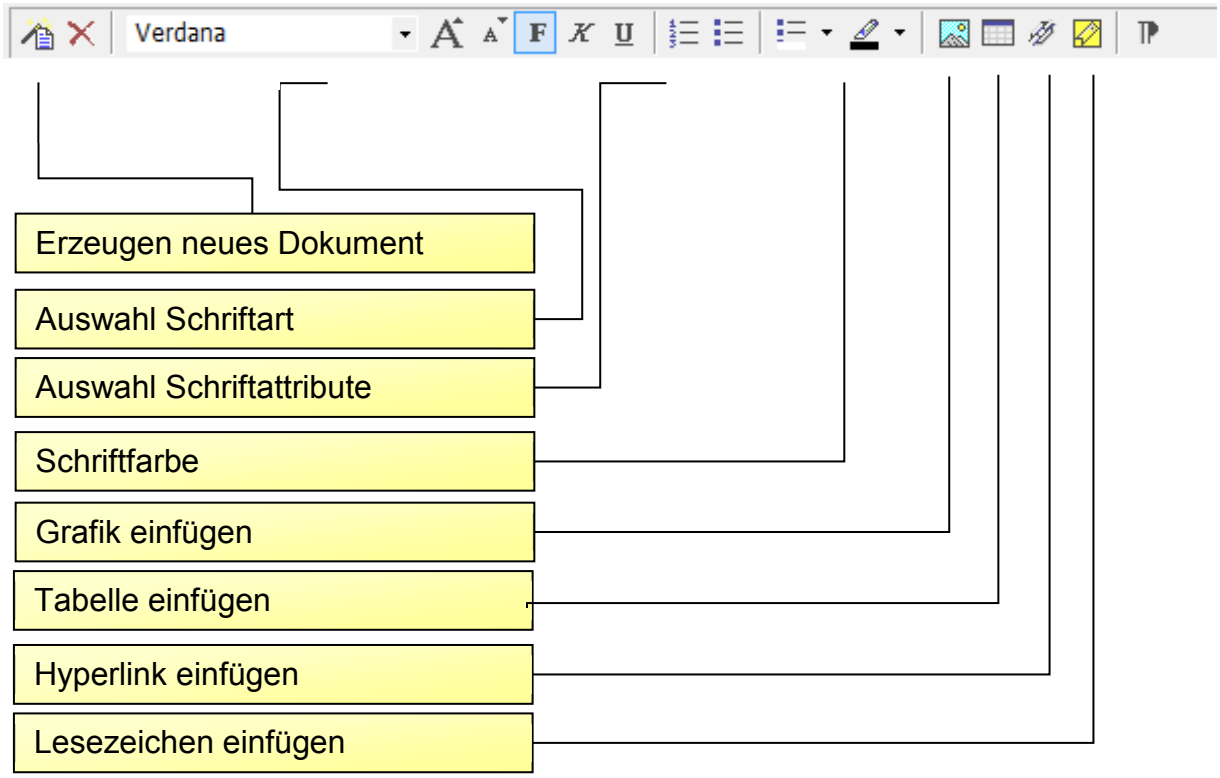



**Bild 8.15:** Toolbar zur Gestaltung der Dokumentation

### 8.2.2.11 Sicherheit

SimulationX bietet ein zweistufiges Sicherheitskonzept (siehe **Bild 8.16**) zum Schutz globaler Typen vor unberechtigtem Zugriff oder Veränderungen.

Normalerweise werden globale Typen in der Modellbeschreibungssprache Modelica® als Textdatei gespeichert und sind für alle Anwender frei zugänglich. Durch verschlüsselte Speicherung kann der Einblick in diese Dateien unterbunden werden. Trotzdem können diese Modelle und Typen von SimulationX weiterhin ohne Einschränkung geladen werden. Um dies zu verhindern, können Sie die Verschlüsselung optional mit einem Kennwortschutz ergänzen.

Für die Benutzung der so geschützten globalen Typen in SimulationX ist beim Laden der betreffenden Typ- bzw. Packagedatei die Eingabe des korrekten

Kennworts erforderlich. Die so eingegebenen Kennwörter werden in einer rechnerspezifischen und ebenfalls verschlüsselten Datei auf Ihrem Computer gespeichert, um eine wiederholte Abfrage des Kennworts bei jedem Start von SimulationX zu vermeiden. Diese Datei kann nicht auf einen anderen Rechner übertragen werden und wird von SimulationX nach dem Austausch von Hardwarekomponenten wie Festplatte oder Netzwerkkarte neu angelegt, wobei die erforderlichen Kennwörter erneut einzugeben sind.

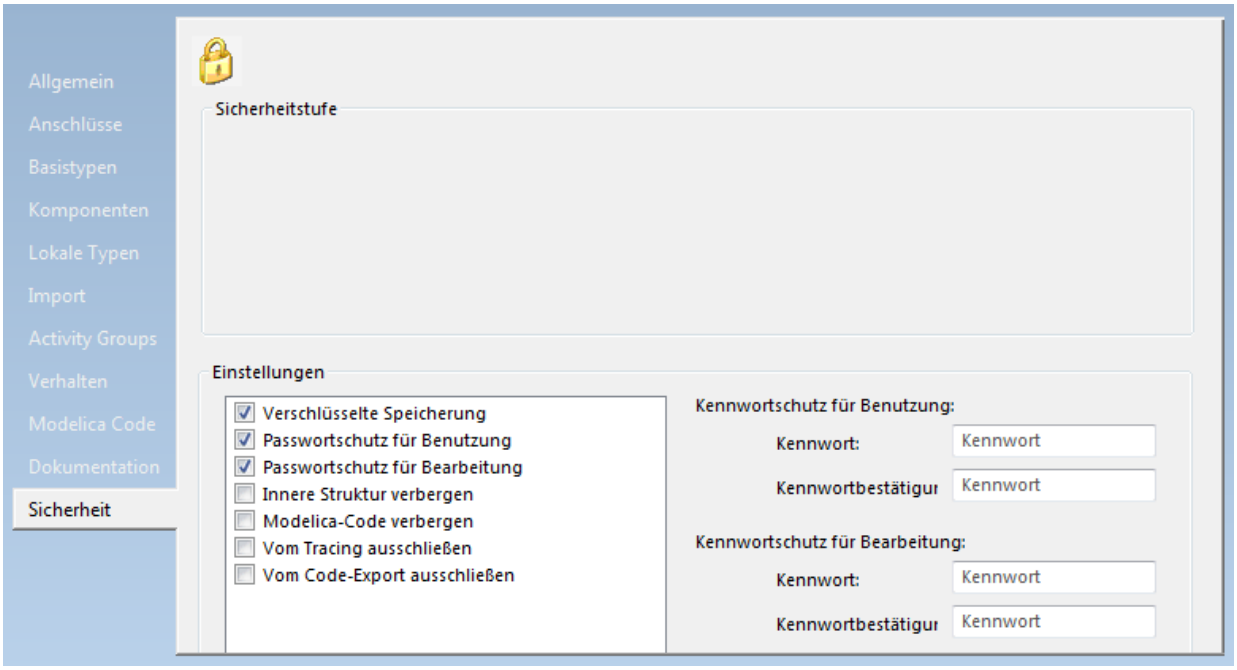



**Bild 8.16:** Dialog Sicherheit

Globale Typen sollen oftmals für andere Anwender zur Nutzung freigegeben werden. Dabei kann es wünschenswert sein, den Einblick in die Wirkungsweise sowie jegliche Veränderungen an dem Verhalten des Typs zu unterbinden. Dies kann durch die Angabe eines zweiten Kennworts (siehe **Bild 8.17**) erreicht werden.

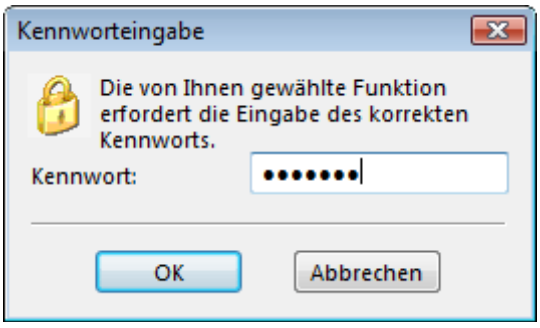



**Bild 8.17:** Dialog Kennworteingabe

Für die Ausführung sämtlicher Aktionen, die das Verhalten des Typs offen legen (Konvertierung in lokale Typen) oder eine Veränderung am Typ gestatten (Aufruf des TypeDesigners) ist die Eingabe dieses Kennworts obligatorisch. Eine Speicherung dieses Kennworts erfolgt nicht.

Die Option zur Verschlüsselung sowie der zweistufige Kennwortschutz kann nur von voll autorisierten Anwendern aufgehoben werden.

**Tipp:**
Sorgen Sie für die sichere Aufbewahrung Ihres Kennworts. Ohne ein gültiges Kennwort ist das Öffnen bzw. die Bearbeitung eines entsprechend geschützten globalen Typs nicht mehr möglich!

### 8.2.3 Ableiten von Typen (Basistypen)

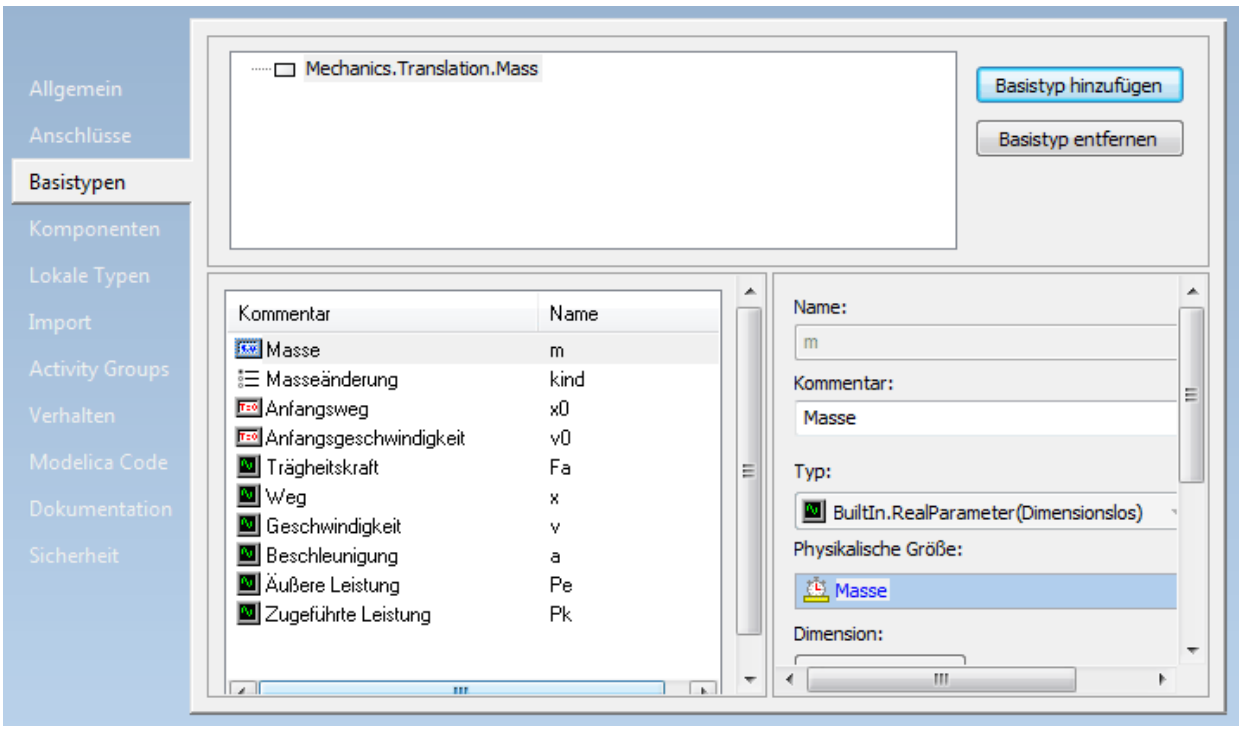



**Bild 8.18:** Basistypen

Mit Hilfe des TypeDesigners können Sie aus einem bereits bestehenden Typ einen neuen Typ ableiten. Sie haben mehrere Möglichkeiten eine Ableitung zu erzeugen:

- Wählen Sie den Menüeintrag ***Neue Ableitung*** im Kontextmenü des gewünschten Basistypen.
- Erzeugen Sie einen neuen Typen, öffnen Sie den TypeDesigner und fügen auf der Seite Basistypen die gewünschten Basisklassen hinzu.
- Erzeugen Sie den abgeleiteten Typ aus einem bereits im Modell vorhandenen Element. Markieren Sie zuerst das Element, dessen Typ als Vorlage für den neu zu erstellenden Typ dienen soll. Danach erzeugen der Ableitung (Menü ***Elemente/Ableiten*** oder -Button in der Toolbar).

Die Seite *Basistypen* zeigt alle direkten und indirekten Basisklassen eines Typs als Baumhierarchie. Je nach gewählter Basisklasse enthält der untere Teil des Dialogs alle Komponenten deren Eigenschaften im aktuellen Typ modifiziert werden können.

Wenn die Eigenschaft einer geerbten Komponente nicht modifiziert ist, dann wird immer der entsprechende Wert aus dem Basistyp verwendet. Änderungen der Eigenschaften von Komponenten im Basistypen wirken sich auf entsprechenden Eigenschaften der Komponenten in den abgeleiteten Typen aus. Modifizierte Eigenschaften bleiben auch nach Änderung im Basistypen unverändert. Komponenteneigenschaften ohne Modifikation besitzen eine blaue Textfarbe. Modifizierte Komponenten werden durch schwarzen Text angezeigt. Eine Modifikation wird gelöscht, wenn der Text im Eingabefeld gelöscht wird.

Um einen Komponente auszuwählen, führen Sie die folgenden Arbeitsschritte aus:

1. Wählen Sie den entsprechenden Basistypen
2. Wählen Sie die gewünschte Komponente aus der Liste
3. Modifizieren Sie die veränderbaren Eigenschaften.

Activity Groups und Parametergruppen werden von Basistypen geerbt und sind nicht veränderbar. Sie können jedoch eigene Activity Groups und Parameter-gruppen hinzufügen.

Das Verhalten des Basistyps wird **nicht** angezeigt. Während der Simulation wird das Verhalten des Basistyps immer **vor** dem Verhalten der Ableitung ausgeführt. Sie können also auf Ergebnisse des Basistyps zurückgreifen oder diese verändern.

### 8.2.4 Zusammenfassen von Teilmodellen (Compound)

Ein komplettes Simulationsmodell oder ein Teilmodell können für die Nutzung in anderen Modellen zu einem neuen Element-Typ zusammengefasst werden.
Dazu führen Sie die folgenden Arbeitsschritte aus:

1. Markieren Sie den Modellteil, den Sie zu einem neuen Element-Typ zusammenfassen wollen (siehe **Kap. 5.2.3** Auswahl mehrer Elemente).

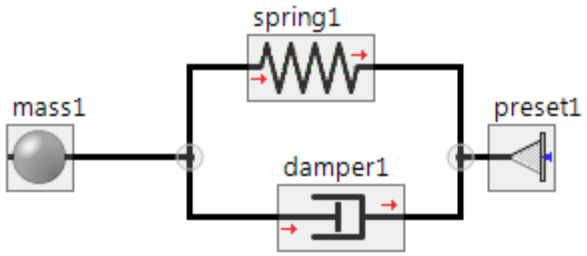



**Bild 8.19:** Markierte Elemente

2. Öffnen Sie den TypeDesigner (***Elemente/Zusammenfassen...*** oder ).Die Seiten des TypeDesigners entsprechen den in **Kapitel 8.2.1** beschriebenen - erweitert um die Seiten „Innere Anschlüsse“ und „Innere Komponenten“.

**Hinweis für ITI-SIM 3.x Anwender:**
Das Zusammenfassen von Elementen zu einem neuen Element-Typ ist vergleichbar mit dem Erstellen eines Makros in ITI-SIM 3.x. Der wichtigste Unterschied der Verwendung von Typen gegenüber Makros wird deutlich, wenn man die Auswirkungen betrachtet, die eine spätere Änderung mit sich bringt. Während die Modifikation innerhalb eines Makros von Hand auf jedes vorkommende Objekt übertragen werden muss, wirken sich Änderungen an einem Typ automatisch auf Komponenten dieses Typs aus.

Beispiel:
Entwickeln Sie eine Modellstruktur entsprechend **Bild 8.19**. Markieren Sie die Elemente und fassen diese zu einem neuen Typ zusammen.
In den folgenden Abschnitten werden Sie das Compound vervollständigen.

### 8.2.4.1 Innere Anschlüsse

Für den neuen Element-Typ können Sie beliebige Anschlüsse der zusammengefassten Elemente nach außen führen.

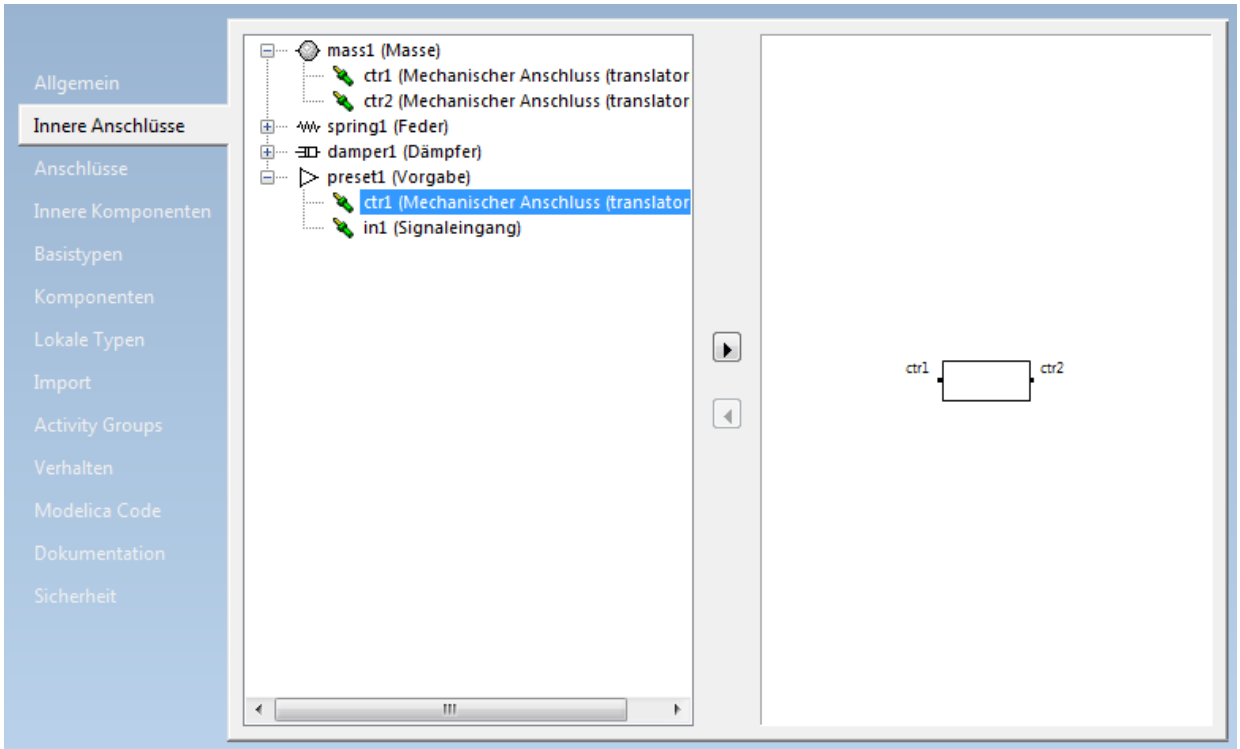



**Bild 8.20:** Auswahl der Anschlüsse

Für jeden so zu übernehmenden Anschlusspunkt führen Sie die folgenden Arbeitsschritte aus:

1. Wählen Sie in der Baumansicht den nach außen zu führenden Anschluss mit einem Mausklick aus.

2. Mit der Schaltfläche Hinzufügen wird der neue Anschluss erstellt und im Vorschaufenster dargestellt.

3. Mittels *Drag & Drop* können Sie nun den Anschlusspunkt innerhalb des Vorschaufensters an die gewünschte Position verschieben.

Wählen Sie für das Beispiel den Anschluss `ctr1.Mass1` aus.

Die ausgewählten inneren Anschlüsse sind dann für den zusammengefassten Element-Typ unter *Anschlüsse* verfügbar (siehe **Bild 8.21**).

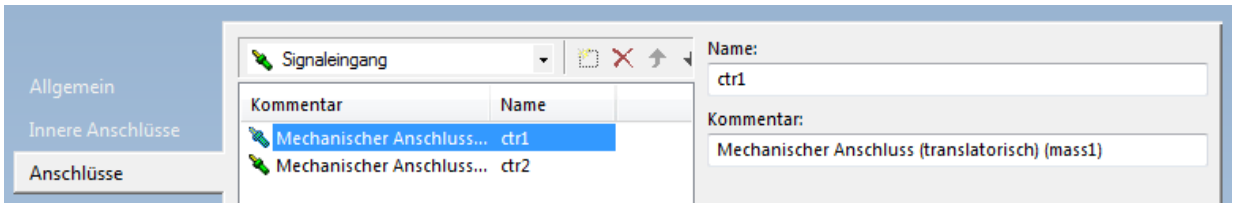



**Bild 8.21:** Anschlüsse des neuen Typ

**Tipp:**
Sie können Anschlüsse direkt aus der Baumansicht mittels *Drag & Drop* in das Vorschaufenster ziehen und an der gewünschten Position platzieren.

Um einen nach außen geführten Anschluss zu löschen, markieren Sie diesen mit einem Mausklick und betätigen die Schaltfläche *Entfernen* .

### 8.2.4.2 Innere Komponenten

Auf dieser Seite können beliebige Komponenten des zusammengefassten Typs zur Ansicht und Bearbeitung im Eigenschaftsfenster zugänglich gemacht werden.

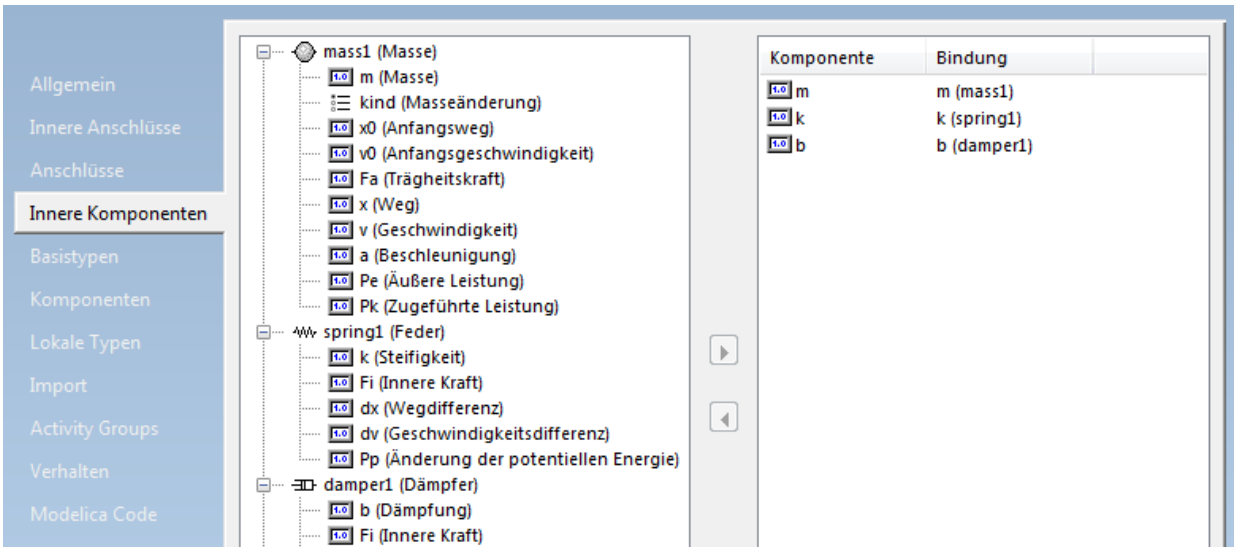



**Bild 8.22:** Innere Komponenten

Um einen Parameter auszuwählen, führen Sie die folgenden Arbeitsschritte aus:

1. Wählen Sie in der Baumansicht die betreffende Komponente mit einem Mausklick aus.
2. Übernehmen Sie die Komponente mit der Schaltfläche *Hinzufügen* in die Vorschauliste.

Beispiel:
Übernehmen sie die folgenden Komponenten:

- Parameter `Mass1.m`
- Parameter `Spring1.k`
- Parameter `Damper1.b`

Das Ergebnis ist in **Bild 8.22** sichtbar.

**Tipp:**
Sie können Komponenten auch aus der Baumansicht mittels Drag & Drop in die Vorschauliste ziehen.

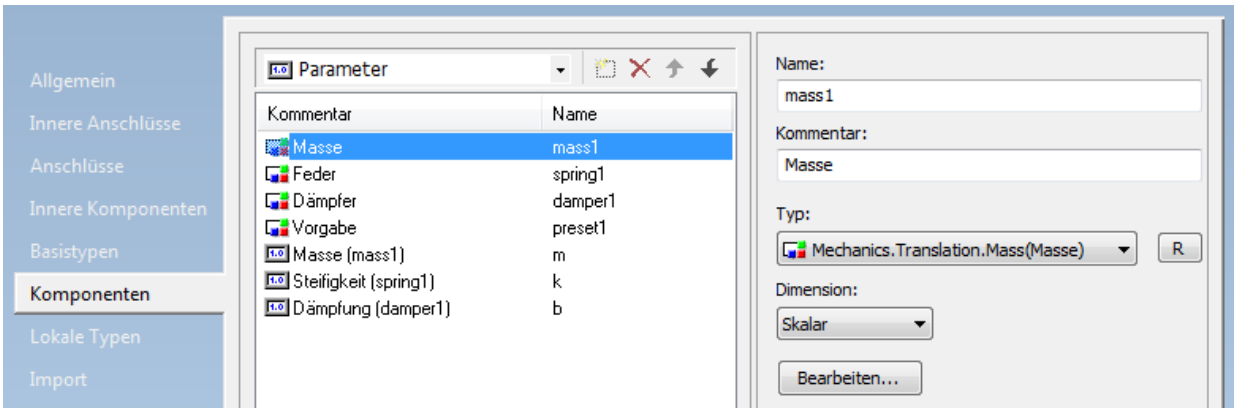
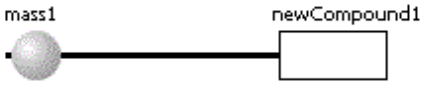



**Bild 8.23:** Komponenten mit geänderten Eigenschaften

Um eine übernommene Komponente zu löschen, markieren Sie diese in der Vorschauliste und betätigen die Schaltfläche *Entfernen*. Die übernommenen inneren Komponenten stehen dann auf der Dialogseite „Komponenten“ zur Verfügung und es können bestimmte Eigenschaften (z. B. der Kommentar, physikalische Größe) geändert werden.
Im Beispiel werden die *Default* -Eigenschaften beibehalten (siehe **Bild 8.23**).

Bei Nutzung des Compound können Sie die Komponenten zur Parametereingabe nutzen (s. **Bild 8.24**).
Da im Beispiel keine weiteren Einstellungen (Activity Groups, Verhalten) notwendig sind, wird jetzt der Button Fertigstellen gedrückt. Damit ist das Compound erstellt.

Beispiel:
Nutzen Sie das erstellte Compound (NewCompound1) und erweitern es um eine Masse (mass1).
Folgende Parameter werden vereinbart:




Für das Beispiel nutzen Sie folgende Parameter (**Bild 8.24):**

`mass1.m` = 200 g
`mass1.x0` = 10 mm

`newCompound1.m` = 50 g
`newCompound1.k` = 1 N/mm
`newCompound1.b` = 2 Ns/m

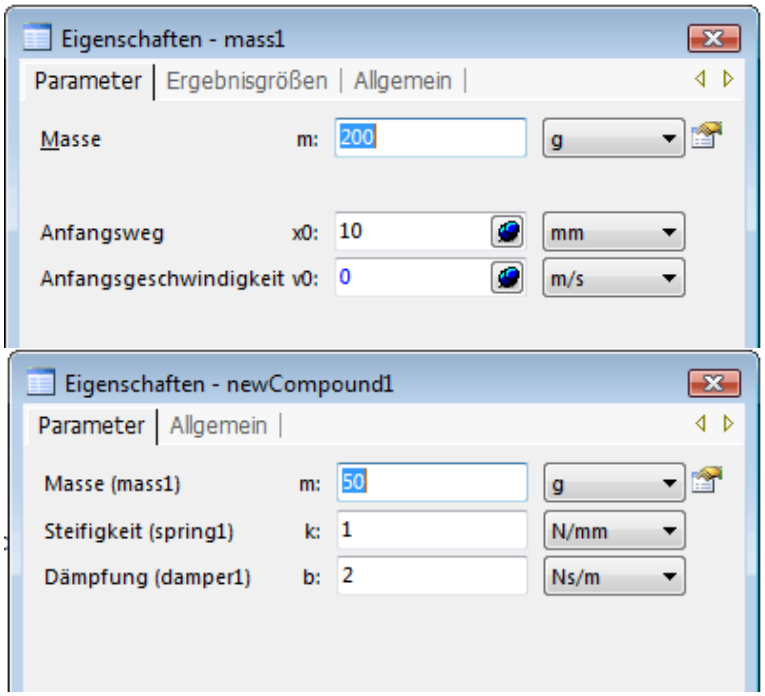



**Bild 8.24:** Parameter der Masse und des Compound

Starten Sie die Simulation und lassen Sie sich den Weg der Masse (mass1.x) in einem Ergebnisfenster anzeigen.
Vergleichen Sie das Ergebnisdiagramm (**Bild 8.25**) mit den Ergebnissen von **Kapitel 3**.

Auf die erzeugten Typen können alle Operationen, wie sie in **Kapitel 4** und **Kapitel 5** beschrieben sind, angewandt werden (d. h. Verschieben, Spiegeln, Drehen...).
Mittels der Schaltfläche in der Toolbar kann für einen zusammengefassten Typen in der Strukturansicht die innere Struktur dargestellt werden. Diese Ansicht wird mit geschlossen.

Eine Änderung des Typs ist jedoch nur im TypeDesigner möglich!

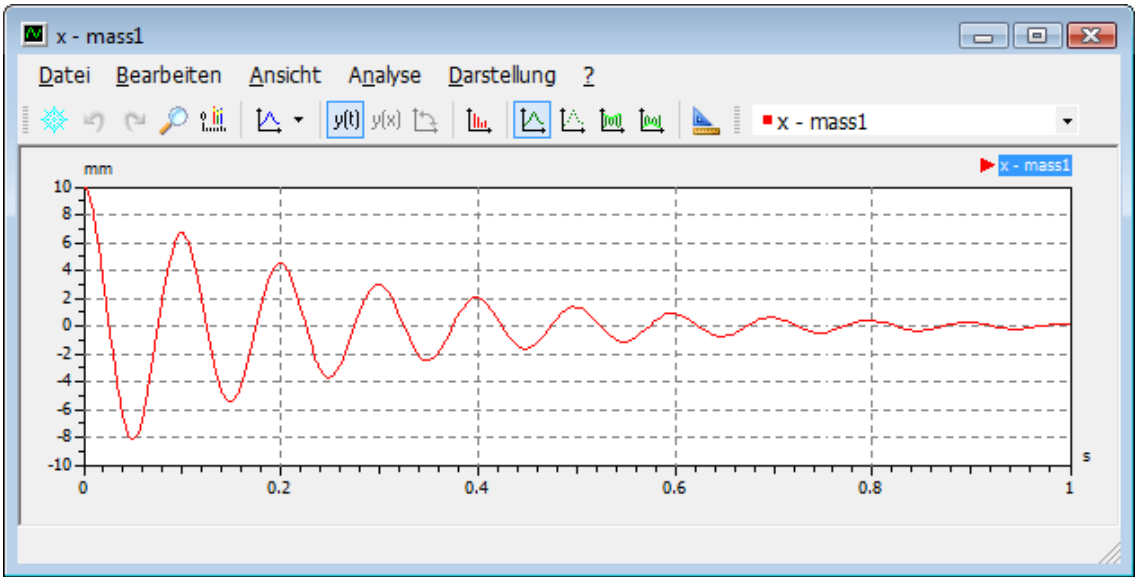



**Bild 8.25:** Ergebnisfenster für Compound-Anwendung

## 8.2.5 Anlegen von Bibliotheken oder Packages (SimulationX)

Hinweise hierzu finden Sie in **Kapitel 8.1** und **Kapitel 10**.

## 8.2.6 Funktionen

### 8.2.6.1 Interne/Externe Funktionen

Funktionen dienen zur Definition einer zustandsfreien Berechnungsvorschrift. Die Reihenfolge der Eingabe- und Ausgabeargumente entspricht der Argumentliste und ist beim Aufruf der Funktion zu berücksichtigen. Das Verhalten wird bei internen Funktionen durch einen Algorithmus und bei externen Funktionen durch Aufruf einer, von einer DLL-Bibliothek exportierten, Funktion definiert.

Die Schaltfläche ***Ableitung…*** erlaubt die Definition einer Funktion zur Berechnung der Ableitung einer bestimmten Ordnung. Weitere Informationen können Sie der Modelica®-Spezifikation **[5]** entnehmen.

Sie können bei der Erstellung von externen Funktionen auch Tracing nutzen, welches dann bei der Trace-Auswertung in SimulationX genutzt werden kann. Siehe dazu Abschnitt 12.8.6 der Modelica®-Spezifikation **[5].**

### 8.2.6.2 Partielle Ableitungsfunktionen

Partielle Ableitungsfunktionen definieren die Ableitung einer Funktion bezüglich eines oder mehrerer Eingabeargumente der referenzierten Funktion. Weitere Informationen können Sie der Modelica®-Spezifikation **[5]** entnehmen.

### 8.2.6.3 Beispiel Externe Funktion

Zunächst sollen die grundlegenden Schritte, die zum Aufruf externer Funktionen notwendig sind, erläutert werden. In einem begleitenden Beispiel werden diese Schritte demonstriert. Danach erhalten Sie weitere Informationen, die für spezielle Anwendungsfälle externer Funktionen von Bedeutung sind.

**Beispiel:** Einbindung der externen Funktion `times_two_c` aus dem folgenden Quelltext in ein Modell.

```
/*
  Simple example of an external function.

  Compilation with the Microsoft C++ Compiler:
  cl.exe times_two.cpp -o times_two_lib.dll -link -dll

  Compilation with gcc in the cygwin environment:
  g++.exe times_two.cpp -o times_two_lib.dll \
          -mno-cygwin -shared
 */

// Compiler dependent settings:
#ifdef _MSC_VER
#define DLL_EXPORT __declspec(dllexport)
#else
#define DLL_EXPORT
#endif
/************* External function *************/
extern "C" DLL_EXPORT double times_two_c(double x)
{
  return 2.0*x;
}
```

Mit der Beispielsammlung zu SimulationX werden folgenden Dateien bereitgestellt:

| | | |
|---|---|---|
| `times_two.cpp` | – | Quelltext |
| `times_two_lib.dll` | – | kompilierte Bibliothek |
| `times_two_model.ism` | – | Modell mit internem Typ times_two, der die Funktion times_two_c nutzt. |

**Schritt 1:**

Als Schnittstellen zu externen Funktionen dienen lokale Typen „f(x) Funktion“. Fügen Sie für die externe Funktion, die Sie in der Verhaltensbeschreibung eines Elementtyps nutzen möchten, eine Funktion f(x) zu den lokalen Typen des Elementtyps hinzu und geben Sie dieser einen Namen (siehe auch Abschnitt **8.2.2.3**).

Wählen Sie den Funktionstyp *Externe Funktion* aus.

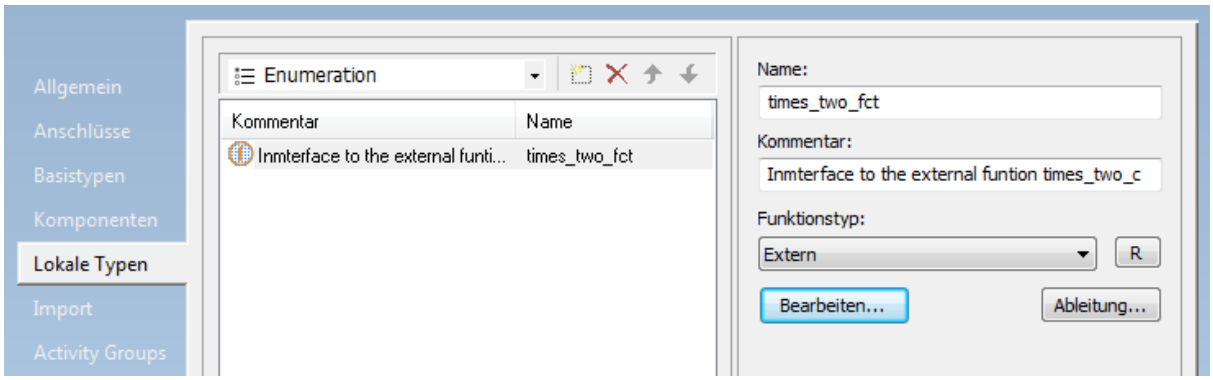



**Bild 8.26:** Hinzufügen der Komponente times_two_fct

In **Bild 8.26** sehen Sie unter den Lokalen Typen des Elementtyps `times_two` die Funktion `times_two_fct`, die als Schnittstelle zur externen Funktion `times_two_c` dient.

**Schritt 2:**

Teilen Sie SimulationX mit, wie die externe Funktion aufzurufen ist. Klicken Sie dazu auf den Button ***Bearbeiten*** der neu erstellten *Funktion*. Es öffnet sich der Dialog zur Funktion (**Bild 8.27**).

Auf der Seite für die *Argumente* (siehe **Bild 8.27**) legen Sie die Ein- und Ausgänge der Funktion fest. Die Bedienung und die Möglichkeiten der Einstellungen entsprechen denen der Parameter unter **8.2.2.3**.

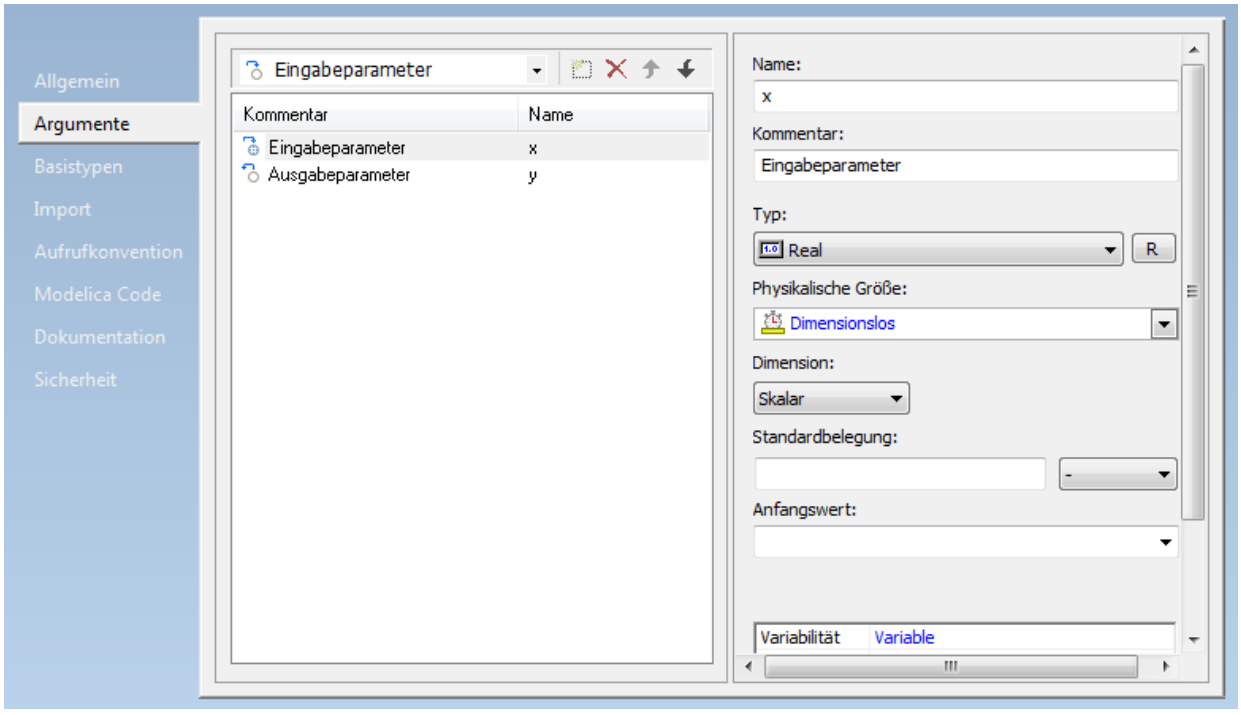



**Bild 8.27:** Argumentendefinition

Die externe Funktion times_two_c aus dem Beispiel hat ein Argument vom Typ *double* und einen Rückgabewert vom Typ *double*. Das entspricht in SimulationX einem Eingabeparameter vom Typ *Real* (vgl. **Bild 8.27**) und einem Ausgabeparameter vom Typ *Real*.

Die eigentliche Vereinbarung zur externen Funktion erfolgt unter *Aufrufkonvention* (**Bild 8.28**).

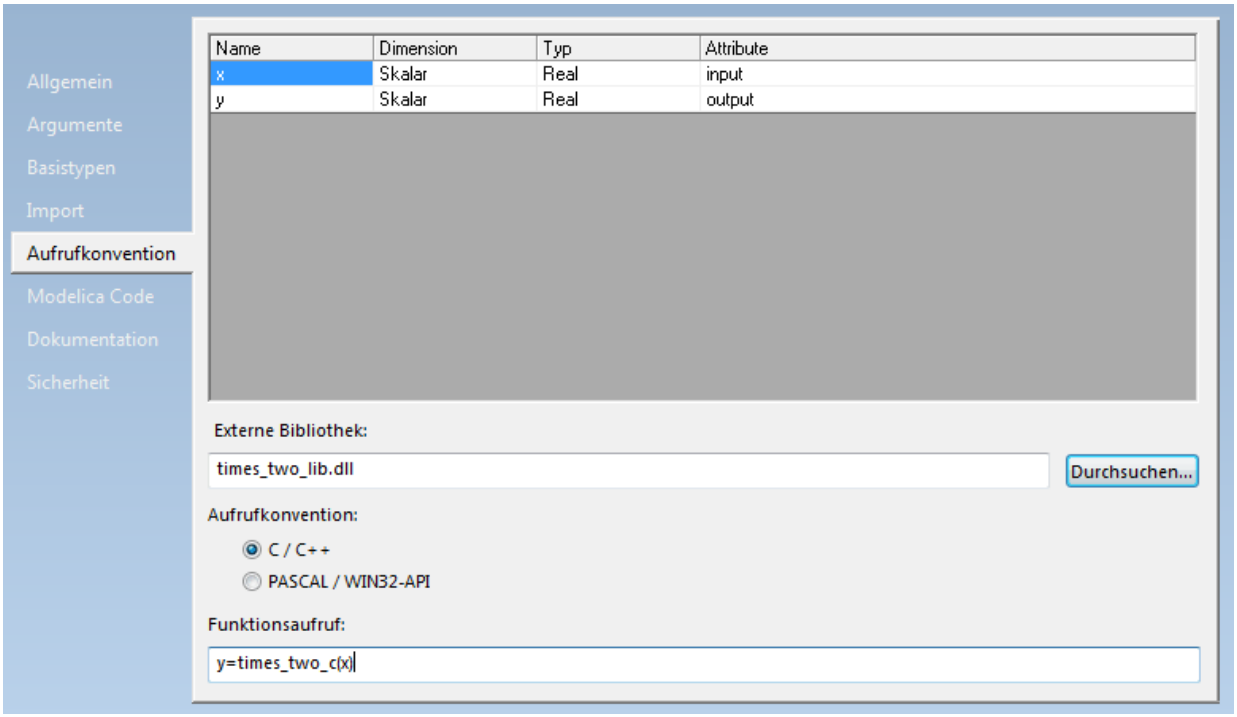



**Bild 8.28:** Konventionen zur Nutzung der Funktion

In das Eingabefeld *Externe Bibliothek* tragen Sie den Namen der DLL-Datei ein, die die externe Funktion exportiert.

Mit dem Umschalter *Aufrufkonvention* teilen Sie SimulationX mit, welcher Aufrufkonvention die externe Funktion folgt ("C/C++" oder "PASCAL/WIN32-API").

Im Eingabefeld *Funktionsaufruf* wird festgelegt, wie SimulationX die externe Funktion aufrufen soll. Die beim Funktionsaufruf einsetzbaren Argumente und Rückgabewerte werden in der Tabelle oben auf der Seite *Aufrufkonvention* (**Bild 8.28**) noch einmal angezeigt.

Entsprechend der Definition der externen Funktion `times_two_c` steht bei der Funktion `times_two_fct` im Feld Funktionsaufruf die Zeichenkette `y=times_two_c(x)`. Die externe Funktion erhält als Argument den Wert des Eingangs `x` und weist ihren Rückgabewert dem Ausgang `y` zu.

**Schritt 3:**
Nachdem Sie in den Schritten 1 und 2 eine Funktion zu den lokalen Typen eines Elementtyps hinzugefügt und konfiguriert haben, können Sie diese in der Verhaltensbeschreibung des Typs nutzen. Das ist in **Bild 8.29** für die Funktion `times_two_fct` aus dem Beispiel demonstriert. Hier erfolgt eine Vereinbarung in der Kategorie *Algorithm*.

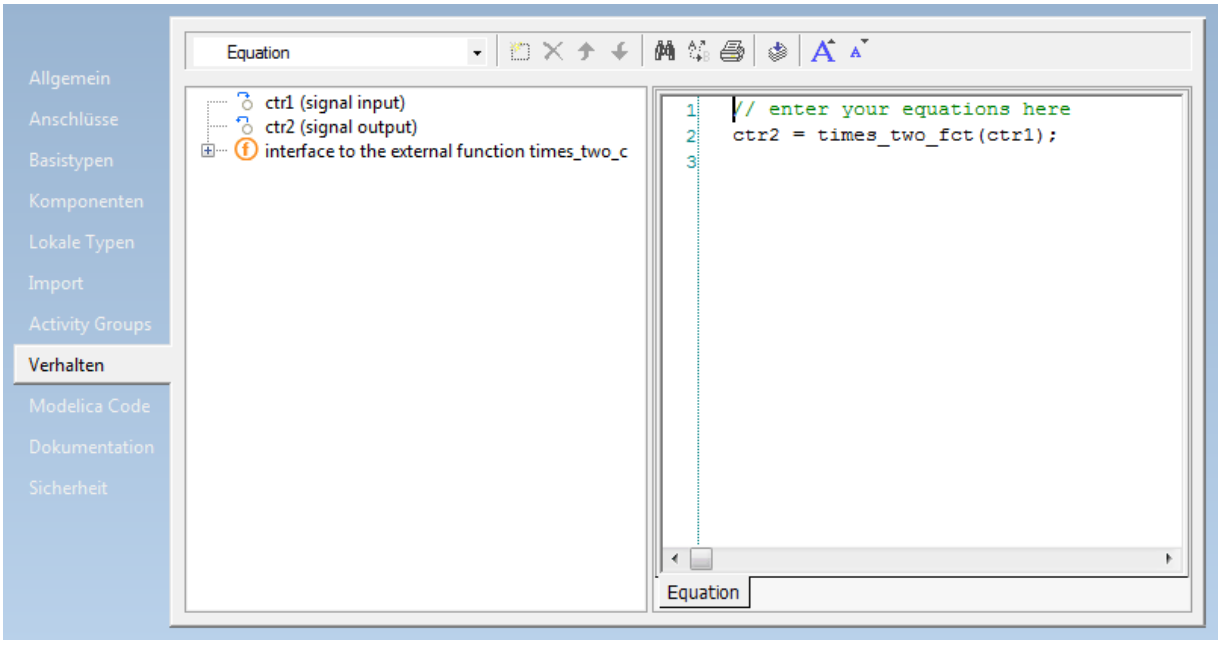



**Bild 8.29:** Verwendung der Funktion times_two_fct in der Verhaltensbeschreibung des Elementtypen times_two

**Wichtige Details:**

Hinweise zur Seite Argumente (**Bild 8.27**):

Diese Seite ähnelt sehr der Seite *Komponenten* im TypeDesigner (**8.2.2.4**) und ist analog zu bedienen. Als Typen für die Argumente stehen jedoch nur *Eingabeparameter* und *Ausgabeparameter* zur Verfügung. *Eingabeparameter* können Sie auf der Seite *Aufrufkonvention* im Feld *Funktionsaufruf* als Argumente und *Ausgabeparameter* als Rückgabewerte der externen Funktion benutzten.

**Name:** Mit dem hier eingetragenen Namen referenzieren Sie den Ein- oder Ausgabeparameter im Feld *Funktionsaufruf* auf der Seite *Aufrufkonvention*.
In der Verhaltensbeschreibung des Elementtyps dient dieser Name auch als Bezeichner bei einer Argumentangabe per Name (nähere Erläuterung weiter unten).

**Typ:** Modelica-Typ des Ein- oder Ausgabeparameters.
Im begleitenden Beispiel hat die Funktion *times_two* ein Argument vom Typ *double* und einen Rückgabewert vom Typ *double*. Das korrespondiert zu einem Eingabeparameter vom Typ *Real* und zu einem Ausgabeparameter vom Typ *Real*.

**Dimension:** Es ist möglich reellwertige (Real), ganzzahlige (Integer) oder wahrheitswertige (Boolean) Spaltenvektoren und Matrizen als Eingabe- oder Ausgabe-

parameter an externe Funktionen zu übergeben. Mögliche Dimensionseinstellungen sind:

- ***Skalar*:** Bei Verwendung als Eingabeparameter oder Rückgabewert wird der Wert direkt übermittelt. Also wird z. B. ein Modelica-Skalar vom Modelica-Typ *Real* als C-Wert vom C-Typ *double* übergeben.
  Im Gegensatz dazu erfolgt bei Verwendung eines Skalars als Ausgabeparameter *in der Argumentliste* die Übergabe mittels Zeiger auf den Speicherplatz der Ausgabevariablen. Zum Beispiel wird für einen Real-Ausgabeparameter in der C-Argumentliste ein Zeiger mit dem C-Typ *(double*)* übergeben.
- ***Vektor:*** Bei Modelica-Vektoren wird an die C-Funktion immer ein Zeiger auf den Speicherbereich des Vektors übergeben. Also liefert SimulationX zum Beispiel für einen Real-Vektor der gerufenen C-Funktion einen Zeiger mit dem C-Typ *(double*)*. Wählt der Anwender den Dimensionstyp *Vektor*, so ist von ihm zusätzlich die Größe des Vektors als natürliche Zahl (größer null) einzutragen. Bitte berücksichtigen Sie in diesem Zusammenhang auch die weiter unten stehenden Hinweise zum Dimension-Typ *frei*.
- ***Matrix2D:*** Grundsätzlich gelten auch hier die Aussagen für *Vektor*. Die Modelica-Matrix wird in dem reservierten Speicherfeld zeilenweise abgelegt, also zuerst alle Elemente der ersten Zeile, danach alle Elemente der zweiten Zeile u.s.w.
- ***Matrix3D:*** Auch hier gelten die Aussagen für *Vektor*. Dem Element M[i,j,k] der Modelica-Matrix entspricht das Element mit dem Index size(M,1)*(size(M,2)*i + j) + k - 1 in dem an die C-Routine übergebenen Speicherfeld.
- ***frei:*** In vielen Anwendungsfällen ist die Größe eines Vektors oder einer Matrix bei der Definition einer externen Funktion nicht bekannt. Zum Beispiel kann es sein, dass sich die Größe des Vektors bzw. der Matrix erst durch die Parametrisierung eines Objekts des neu erstellten Typs oder durch Signalverbindungen im Gesamtmodell ergibt.
  In diesem Fall ist statt der Dimensions-Typen *Vektor, Matrix2D* oder *Matrix3D* der Dimensions-Typ *frei* zu wählen. Es wird dann ein Eingabefeld sichtbar, in das die *freie Dimensionsdefinition* einzutragen ist. Die *freie Dimensionsdefinition* ist eine kommaseparierte Liste von Dimensionsangaben für das mehrdimensionale Datenfeld (*Vektor* oder *Matrix*). Für die Definitionsangaben kann man alle dafür gültigen Modelica-Terme nutzen. Zum Beispiel spezifiziert man mit der freien Dimensionsdefinition ":,:" (ohne die einschließenden Hochkommas) eine zweidimensionale Matrix, bei der die Größe an dieser Stelle noch nicht festgelegt wird. Weiterhin kann man sich mit Hilfe des *size*-Operators auf die Dimensionsgrößen der anderen Eingabeparameter beziehen.

***Die anderen Felder der Aufrufkonventionen***

Beim Eingabefeld *Externe Bibliothek* können sie den Pfad der DLL-Bibliothek absolut oder relativ angeben. Eine relative Pfadangabe bezieht sich auf die Verzeichnisse, die unter *Extras/Optionen/Externe Funktionen* eingetragen sind und die Standardsuchpfade von WINDOWS. Das sind:

- die in der Umgebungsvariable `PATH` angeführten Verzeichnisse;
- das Verzeichnis, in dem sich das Programm SimulationX (`simx.exe`) befindet (standardmäßig: `c:\Programme\ITI-Software\SimulationX`).

Alternativ zur direkten Eingabe des Pfades der DLL-Bibliothek können Sie nach Klicken auf das Ordnersymbol den Verzeichnisbaum nach der DLL-Bibliothek durchsuchen.

DLL-Bibliotheken, die aus Fortran-Quellen erstellt wurden, genügen in der Regel auch den *Aufrufkonventionen* von "C/C++". Für skalare Argumente werden jedoch von Fortran-Funktionen Zeiger auf Variablen vom Typ Integer oder Real erwartet. Verwenden Sie also in diesem Fall für skalare Eingänge 1x1-Datenfelder.

Im Eingabefeld *Funktionsaufruf* geben Sie den Namen der externen Funktion gefolgt von der in runden Klammern eingeschlossenen kommaseparierten Argumentliste an. Zusätzlich können Sie die Funktion `size` benutzen.

`size(`*`Array`*`,` *`n`*`)` gibt die Größe der *n*-ten Dimension von *Array* zurück.

**Achtung:** Eine falsche Angabe der Argumente kann einen Programmabsturz zur Folge haben.

Beachten Sie außerdem die Hinweise im nächsten Abschnitt.

***Datenformate der Argumente externer Funktionen***

Skalare Ausgangsvariablen können direkt als Rückgabewert der externen Funktion definiert werden.

Beispiel in C: `double times_two(double x)`

In das Eingabefeld *Funktionsaufruf* schreibt man dann den Ausgang, dem dieser Rückgabewert zugewiesen werden soll, durch ein Gleichheitszeichen getrennt vor den Funktionsaufruf.
Beispiel: `y = times_two(x)`

Alternativ können Sie Ausgangsvariablen in die Argumentliste der externen Funktion aufnehmen. In diesem Fall werden an die externe Funktion Zeiger (in C: `double*` oder `int*`) auf die Datenfelder übergeben, in die die Werte der Ausgangsvariablen abzuspeichern sind.

Beispiel: In dem folgenden Kopf einer C-Routine ist `y` die Ausgangsvariable, an die ein Feld als Rückgabewert übergeben wird.

```
void mat_mul(int n, int m, double *y, double* A,
double* x)
```

In dem Eingabefeld Funktionsaufruf schreiben Sie die Ausgangsvariable dann an der richtigen Stelle mit in die Argumentliste:

Beispiel: `mat_mul(size(A,1),size(A,2),y,A,x)`

Eingangsvariablen mit skalarem Wert werden als Skalare an die externe Funktion übergeben (in C: `double` und `int` für die Typen `Real` bzw. `Integer`). Bei Eingangsvariablen mit einem Feld als Wert wird an die externe Funktion ein Zeiger auf das Datenfeld übergeben (in C: `double*` oder `int*`).

### *Verwendung von Funktionen in der Verhaltensbeschreibung eines Typen*

In der Algorithmussektion der Verhaltensbeschreibung nutzen Sie eine Funktion als rechte Seite einer Zuweisung. Bei einer Zuweisung sind linke und rechte Seite durch ":=" getrennt.
In der Gleichungssektion nutzen Sie Funktionen als rechte Seite einer Gleichung, bei der linke und rechte Seite durch "=" getrennt sind.

Hat die Funktion nur einen Rückgabewert, so steht auf der linken Seite die Variable, der der Rückgabewert zugewiesen werden soll.

Beispiel: `y = f(a1,a2);`

Hat die Funktion mehrere Ausgangsvariable, so steht auf der linken Seite eine in runde Klammern eingeschlossene und durch Kommas separierte Liste der Variablen, denen die Rückgabewerte zugewiesen werden sollen.

Beispiel: `(y1,y2) = f(a1,a2);`

Auf der rechten Seite steht der Funktionsaufruf, das heißt, der Name der Funktion gefolgt von einer in runden Klammern eingeschlossenen und durch Komma separierten Liste von Argumenten.

Es gibt zwei Möglichkeiten die Funktionsargumente anzugeben: *per Name* und *per Reihenfolge*.

Bei der Angabe *per Name* wird zuerst der Name des Eingangs und dann ein Gleichheitszeichen "=" gefolgt von dem zuzuweisenden Wert angegeben.

Beispiel: `f(a1=x1, a2=1.0)`

Geben Sie die Eingänge exakt in der Reihenfolge an, in der sie in der Komponententabelle der Funktion stehen, so können Sie die Namen der Eingänge und das Gleichheitszeichen weglassen, was auch als Argumentangabe *per Reihenfolge* bezeichnet wird.

Beispiel: `y=f(x1,1.0)`

Sie können die zwei Varianten auch kombinieren. Dabei müssen Sie zuerst die Argumente per Reihenfolge schreiben und dahinter die Argumente per Name.
Beispiel: `y=f(x1, a2=1.0)`

### 8.2.7 Simulationsspezifische Funktionen

Folgende Funktionen können in die Verhaltensbeschreibung integriert werden. Sie sind teilweise nur in SimulationX verfügbar (mit * gekennzeichnet) und sind nicht Teil der Modelica Sprachebeschreibung

**`abs(x)`**
Wird ersetzt durch (if x>=0 then x else –x).
Außerhalb von noEvent(...) oder der when Anweisung werden Ereignisse zur Behandlung von Unstetigkeiten ausgelöst.

**`analysisType()`***
Gibt eine Zeichenkette zurück, welche den Kontext beschreibt, in welchem das Modell evaluiert wird.
Mögliche Werte sind:

- `dynamic` – Simulation über die Zeit
- `linear` – Lineare Systemanalyse
- `static` – Berechnung des statischen Gleichgewichts
- `stationary` – Stationäre Simulation (im Frequenzbereich)

**`analysisTypeDetail()`***
Gibt eine Zeichenkette zurück, welche den Kontext genauer beschreibt, in welchem das Modell evaluiert wird.
Mögliche Werte sind:

- `residuals` – Berechnung der Residuen am aktuellen Zeitpunkt.
- `residualsChangedStates` – Berechnung der Residuen mit vom Löser gering veränderten Werten der Zustände und/oder der aktuellen Simulationszeit
- `residualsIntegral` – Berechnung der Residuen mit vom Löser veränderten Werten der Zustände für Integralgleichungen
- `jacobian` – Berechnung der Jacobi-Matrix

| | |
|---|---|
| | `event` – ein Event-Schritt (diskrete Variablen können ihren Wert ändern)<br>`animation` – während der Animation<br>`acceptStates` – während der Ausführung des accept-Algorithmus'<br>`other` – sonst |
| **`analysisTypeDepend()`** * | erlaubt das Einbringen zusätzlicher Abhängigkeiten in eine Gleichung, um dem Löser das Verlassen eines ansonsten singulären Bereiches zu ermöglichen. |
| **`assert(Bedingung, Zeichenkette)`** | Zusicherung, dass die Bedingung während der Simulation erfüllt ist. Wenn nicht, wird die Zeichenkette ausgegeben und die Simulation abgebrochen (Fehler). Über die Trace-Einstellungen lässt sich das verhindern. |
| **`cardinality( A )`** | Gibt die Anzahl der Anschlüsse zurück, die mittels einer Verbindung mit A verbunden sind. Gibt 0 zurück, wenn A nicht verbunden ist. |
| **`change(x)`** | Wird ersetzt durch (x <> pre(x)). Liefert **`true`**, wenn sich der Wert der Variable ändert.<br>Das Argument muss eine diskrete Variable sein. |
| **`ceil(x)`** | Gibt den kleinsten ganzzahligen Wert zurück, der nicht kleiner als x ist.<br>Das Argument muss ein diskreter Ausdruck sein. |
| **`der(x)`** | Ableitung der Variable x . (= dx/dt) |
| **`der(y,x)`** * | Ableitung der Variable y nach x . (= dy/dx) |
| **`delay(Ausdruck, dt[ , dtmax])`** | Liefert den Wert des Ausdrucks um dt verzögert. Solange die aktuelle Zeit kleiner als dt ist, liefert es den Anfangswert.<br>dtmax muss eine Konstante sein, und es muss gelten<br>0 <= dt <= dtmax. Wenn es nicht angegeben wird, muss dt konstant sein. |

| | |
|---|---|
| **div(x,y)** | Berechnet den ganzzahligen Quotienten x/y, indem es x/y berechnet und alle Nachkommastellen abschneidet (Rundung in Richtung der 0).<br>Die Argumente müssen diskrete Ausdrücke sein. |
| **dontSolveFor(expr)** * | Verhindert das Umstellen einer Gleichung nach einer in expr enthaltenen Variablen. |
| **edge(b)** | Wird ersetzt durch (b and not pre (b)) für die boolean Variable b. Liefert **true**, wenn der Wert der Variable von **false** nach **true** wechselt.<br>Das Argument muss eine diskrete Variable sein. |
| **floor(x)** | Gibt den größten ganzzahligen Wert zurück, der nicht größer als x ist.<br>Das Argument muss ein diskreter Ausdruck sein. |
| **forCodeGen()** * | Liefert true, wenn eine symbolische Analyse für die Code-Generierung durchgeführt wird. |
| **impact(state)** * | liefert einen Wert ≠ 0, wenn der Zustand state einen Impuls hatte. Das Vorzeichen gibt die Richtung des Impulses an. Kann nur in spezieller Algorithmussektion verwendet werden (accept algorithm) |
| **inAnimation()** * | Liefert true während der Animation. |
| **integer(x), int(x)** | Gibt den größten ganzzahligen Wert zurück, der nicht größer als x ist.<br>Das Argument muss ein diskreter Ausdruck sein. |
| **integral(x,z0)** * | Integral der Variable x ($= \int x dt + z_0$ ). $Z_0$ ist der Anfangswert des Integrals. Er wird nur einmal ausgewertet. |
| **integral(y,x,z0)** * | Das Zeitintegral der Variable y über x $= \int y dx + z_0$ . $Z_0$ ist der Anfangswert des Integrals. Er wird nur einmal ausgewertet. |
| **initial()** | Gibt am Anfang der Simulation den Wert **true** zurück. |

| | |
|---|---|
| **isConstant(expr)** * | Liefert true, wenn der übergebene Ausdruck konstant ist. |
| **message(…)** * | siehe `terminate` |
| **noEvent(Ausdruck)** | Der Ausdruck wird direkt ausgewertet. Eventuell enthaltene Relationen oder unstetige Funktionen lösen keine Ereignisse für die Behandlung von Unstetigkeiten aus. |
| **language( lang, string)** * | Liefert string zurück, wenn lang der eingestellten Sprache entspricht, sonst die leere Zeichenkette.<br>Mögliche Werte für lang sind:<br>"**de**" für Deutsch oder "**en**" für Englisch.<br>Damit lassen sich sprachabhängige Meldungen erzeugen. |
| **last(x)** * | Liefert den Wert der Variablen x aus dem letzten gültigen Integrationsschritt. Während der Ereignisiteration bleibt last(x) konstant. |
| **objectIdent()** * | Liefert den Bezeichner des Objekts, zu dem die Gleichung gehört (für Meldungen). |
| **pre(x)** | Bei der Ereignis-Iteration an Unstetigkeitsstellen liefert der Operator den Wert der Variable x aus dem letzten Iterationsschritt. Außerhalb von Unstetigkeitsstellen gilt immer x = = pre(x).<br>Das Argument muss eine diskrete Variable sein. |
| **reinit(x, Ausdruck)** | Reinitialisiert den Zustand einer Variablen x mit Ausdruck (nur an Unstetigkeitsstellen). Voraussetzung ist, dass x in einem der(x) Operator verwendet wird. |
| **rem(x,y)** | Berechnet den ganzzahligen Rest der Division x/y, so dass gilt:<br>x = div(x,y)*y + rem(x,y).<br>Die Argumente müssen diskrete Ausdrücke sein. |
| **sample(Start, Interval)** | Erzeugt Zeitereignisse zu den Zeitpunkten $Start + \sum_i Interval_i$. Zu diesen Zeitpunkten gibt die Funktion **true** zurück, sonst **false.** Der Wert von Start wird nur einmal ausgewertet, Interval nur an den |

| | |
|---|---|
| | Zeitpunkten. Start >= 0 und Interval > 0 sind erforderlich. |
| **sign(x)** | Wird ersetzt durch (if x > 0 then 1 else if x < 0 then –1 else 0).<br>Außerhalb von noEvent(…) oder der when Anweisung werden Ereignisse zur Behandlung von Unstetigkeiten ausgelöst. |
| **smooth(n, Ausdruck)** | Gibt an, dass der Ausdruck n-fach stetig differenzierbar ist und liefert den Ausdruck zurück. Wenn der Ausdruck an bestimmten Stellen Ereignisiterationen erzeugt, so kann dadurch der dafür notwendige Aufwand reduziert werden. Siehe Abschnitt die noEvent() und smooth()-Funktion |
| **setImpact(state)** * | löst einen Impuls auf dem Zustand state aus, z. B. bei Anschlag. |
| **terminate (Zeichenkette)** | Die Zeichenkette wird ausgegeben und die Simulation (erfolgreich) beendet. |
| **trace(… )** * | s.o. |
| **warning(…)** * | s.o. |

**trace-Funktion**

Die trace-Funktion ermöglicht Anzeigen im Ausgabefenster zur Unterstützung bei der Fehlersuche.
Die Darstellung im Ausgabefenster besteht aus einer kurzen (wird immer dargestellt) und einer langen Zeichenkette (wird nur dargestellt, wenn auf das '+'-Zeichen vor der kurzen Meldung geklickt wird).
Die trace-Funktion kann mit einer variablen Anzahl von Parametern aufgerufen werden.

Das erste Argument ist eine Formatangabe. In ihr können Platzhalter der Form %n enthalten sein, wobei sich das n auf das n-te weitere Argument bezieht. Die Trennung von kurzer und langer Zeichenkette erfolgt durch das erste \n in der Formatangabe.
Die weiteren Argumente können beliebige Ausdrücke sein, deren aktueller Wert in die angegebenen Platzhalter gefüllt wird.
Der Rückgabewert der Funktion ist immer 1. Dadurch kann sie auch in Parametern gerufen werden.
Damit auch wirklich etwas im Ausgabefenster erscheint, muss im Menu unter Simulation/Einstellungen... auf der Seite Tracing unter Modelica die trace-

Funktion eingeschalten sein und der globale Parameter **traceOn** den Wert true liefern.

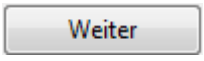
| Aufruf | Ausgabe (kurz) | Ausgabe (lang) | Kommentar |
|---|---|---|---|
| `trace("test")` | test | test | Einfache Zeichenkette |
| `trace("test",t)` | test | 0.23 | Zusätzliches Argument |
| `trace("time:%1, dt: %2",t,dt)` | time: 0.23, dt: 0.01 | time: 0.23, dt:0.01 | Formatangabe im ersten Argument |
| `trace("time: %1\ndt: %2",t,dt)` | time: 0.23 | dt: 0.01 | Lange und kurze Ausgabe ver-schieden |
| `trace("time+dt=%1", t+dt)` | time+dt = 0.24 | time+dt = 0.24 | Ausdruck als Argument |

Ähnlich wie trace funktionieren auch die Funktionen **message** und **warning** zur Ausgabe von Meldungen bzw. Warnungen im Ausgabefenster. Die globalen Einstellungen zum Freischalten der trace Funktion haben darauf jedoch keinen Einfluss.

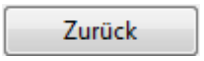
## 8.3 FluidDesigner

Mit dem FluidDesigner erhalten Sie ein Werkzeug zur Erstellung von nutzer-definierten Fluidtypen für die Bibliotheken Hydraulik und Pneumatik.

Die Werkzeugleiste im unteren Teil des FluidDesigners gilt für alle Seiten und hat folgende Funktionen:

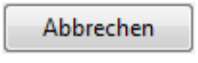


wechselt zur nächsten Seite



wechselt zur vorherigen Seite

erstellt das Fluid und beendet den FluidDesigner



bricht die Arbeit ab

blendet die Online-Hilfe ein

### 8.3.1 FluidDesigner *Hydraulik*

Ein Fluidtyp beschreibt alle für die Simulation relevanten physikalischen **Eigenschaften einer Hydraulikflüssigkeit (wie z. B. Viskosität, Dichte,** Kompressibilität usw.) in Abhängigkeit der Zustandsgrößen Druck, Temperatur

und Luftanteil. Zusätzlich zu den in der Bibliothek Hydraulik bereits intern vordefinierten Fluidtypen kann der Anwender damit selbst neue Fluidtypen definieren. Diese werden dann im Dialog der hydraulischen Verbindung mit angezeigt und können für die Simulation herangezogen werden.

Den **FluidDesigner** starten Sie:

- über das Menü ***Extras/Optionen/Fluide***
  Durch Betätigen des *Hinzufügen* –Schalters wird der FluidDesigner gestartet.

**Bild 8.30:** Fluidauswahl im Menü *Extras/Optionen*

- Aufruf des ***Eigenschaftsdialoges*** einer hydraulischen Verbindung
  Über die Auswahl ***Neues Fluid*** (siehe **Bild 8.31**) wird der FluidDesigner gestartet.

Zum Ändern und Löschen eines nutzerdefinierten Fluidtyps wählen Sie diesen im Menü ***Optionen/Fluide*** aus und betätigen die entsprechende Taste.

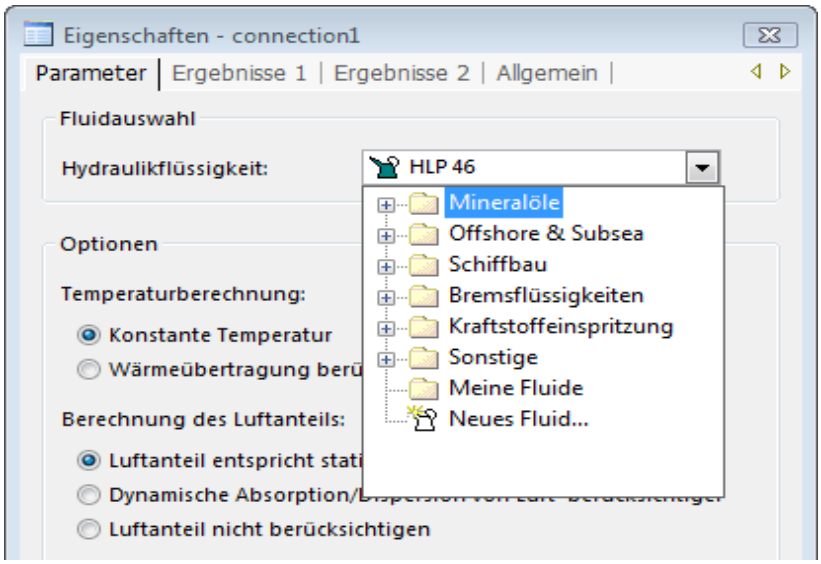



**Bild 8.31:** Eigenschaftsdialog der hydraulischen Verbindung

### 8.3.1.1 Allgemein

Auf dieser Seite benennen Sie den neu zu erstellenden Fluidtyp. Des Weiteren können Sie einen Kommentar einfügen, über eine Auswahl die Klassifikation der Hydraulikflüssigkeit (siehe **Bild 8.32**) vornehmen und Sicherheitseinstellungen ändern.

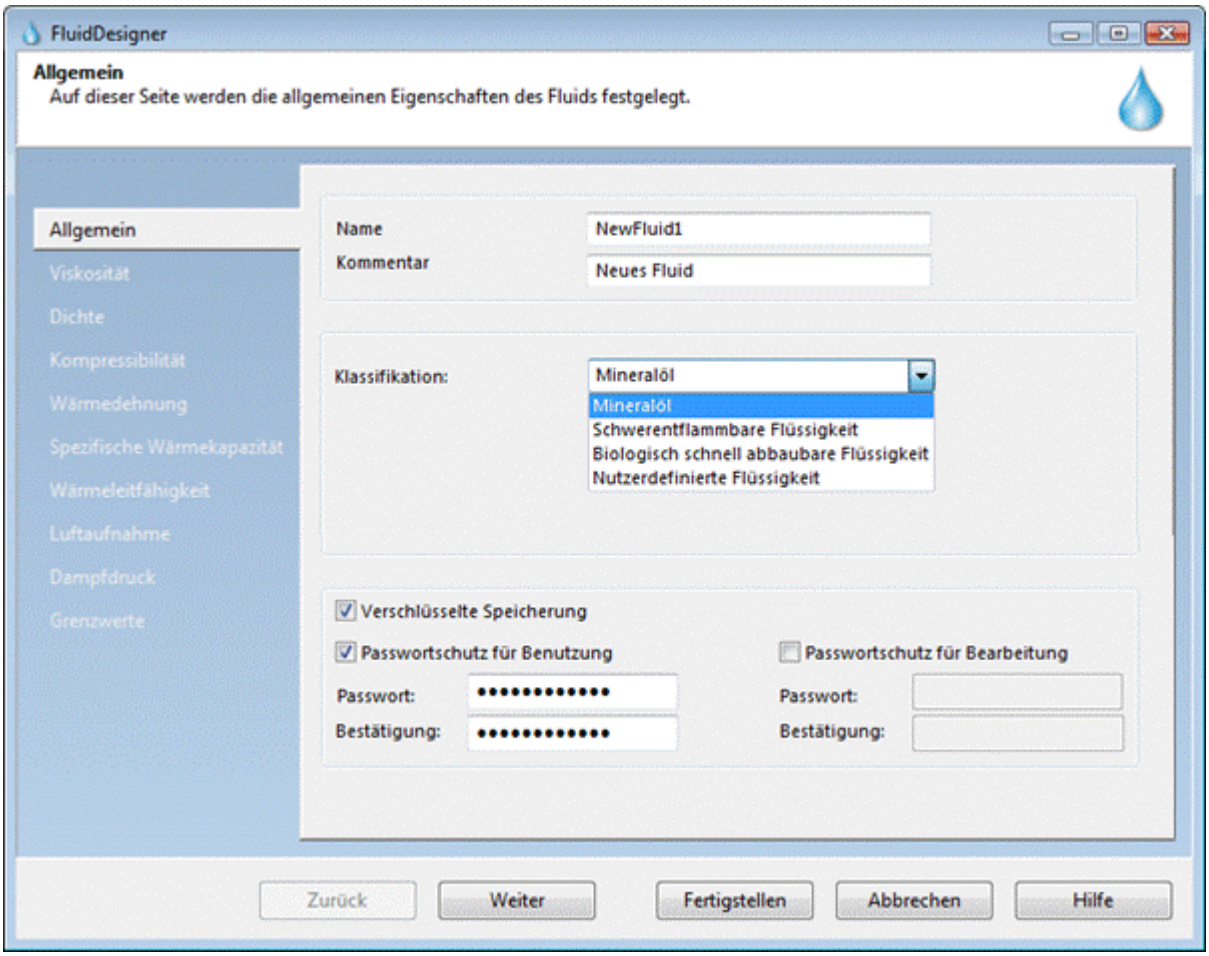



**Bild 8.32:** Allgemeine Einstellungen

Mit Hilfe der **Klassifikation** werden auf den nachfolgenden Dialogseiten bereits passende Standardwerte für die einzelnen Fluideigenschaften angeboten. Bei den Optionen „schwerentflammbare Flüssigkeit“ und „biologisch schnell abbaubare Flüssigkeit“ sollte zusätzlich noch eine Unterklassifikation vorgenommen werden.

Bei Aktivierung der **Sicherheitsoptionen** wird die den Fluidtyp beinhaltende Datei verschlüsselt. Das bedeutet, dass für das Auslesen der Fluideigenschaften SimulationX  und der FluidDesigner benötigt werden. Durch **Passwörter für Nutzung und Bearbeitung** kann vor Missbrauch der selbsterstellten Fluidtypen geschützt werden. Für weitere Informationen zum Thema Sicherheit beachten Sie bitte die ausführlichen Beschreibungen im Kapitel 8.2.2.11 des TypeDesigners. Der neu erstellte Fluidtyp wird standardmäßig im Verzeichnis "...*Eigene Dateien\SimulationX 3.5\My Fluids*" gespeichert. Dieses Verzeichnis kann der Anwender jedoch auch über den Dialog ***Extras/Optionen*** auf der Seite

***Verzeichnisse*** ändern. Alle in diesem Verzeichnis befindlichen Fluidtypen werden beim Neustart des Programms automatisch geladen. Nach Bearbeitung eines bestehenden Fluidtyps wird eine **Sicherheitskopie** der vorherigen Version gespeichert. Beim Ablegen der Sicherheitskopie wird ein fortlaufender Zähler hinter die Dateierweiterung „.mo“ geschrieben.

Auf den folgenden Dialogseiten des FluidDesigners werden die physikalischen Eigenschaften der Hydraulikflüssigkeit definiert. Dabei sollte beachtet werden:

- Für die Druckabhängigkeit sollte stets der absolute Druck verwendet werden. Dieser wird in der hydraulischen Verbindung aus dem relativen Druck (Variable *p* bzw. Anfangswert *p0*) und dem Atmosphärendruck (globaler Parameter `pAtm`) berechnet. Die Variable *p* im Fluidtyp bezieht sich allerdings stets auf den absoluten Druck!
- Mit Hilfe des FluidDesigners sollte nur das Verhalten der *reinen Flüssigkeit*, d. h. ohne Luftanteil und Kavitation, beschrieben werden. Der Einfluss von Luftanteil und Kavitation wird über zusätzliche Parameter wie Bunsenkoeffizient, Dampfdruck usw. beschrieben. Für das Verhalten der realen Flüssigkeit wird diese als ein Gemisch aus reiner Flüssigkeit, Gas (in der Regel Luft) und Dampf (verdampfte Flüssigkeit) betrachtet.
- Wenn die Fluideigenschaften mit Funktionen oder Kennlinien vorgegeben werden, dann beachten Sie bitte, dass einige Eigenschaften voneinander abhängen. Das ist für den Kompressionsmodul, die Dichte, die Wärmedehnung und die spezifische Wärmekapazität der Fall. Weiterhin beeinflussen Luftanteil und Kavitation die Kompressibilität und alle davon abhängigen Eigenschaften. Insbesondere bei Vorgabe ungenauer Funktionen oder Kennlinien für Dichte und Kompressibilität kann die Massebilanz nicht erfüllt werden.

#### 8.3.1.2 Viskosität

Auf dieser Seite wird die kinematische Viskosität der reinen Flüssigkeit (ohne Gasanteil und Kavitation) beschrieben.

Bei der Festlegung der Viskosität können Sie aus folgenden Vorgaben wählen:

- ISO-Viskositätsklasse nach DIN 51519
- Standard-Ansatzfunktion mit Koeffizienten
- Beliebige Ansatzfunktion
- Viskositäts-Temperatur/Viskositäts-Druck-Kennlinien (**Bild 8.33**).

Die Kennlinien können mittels des Schalters

angepasst bzw. neu erzeugt werden. Weiterführende Informationen zum Arbeiten mit Kennlinien finden Sie im **Kapitel 6.4.1**.

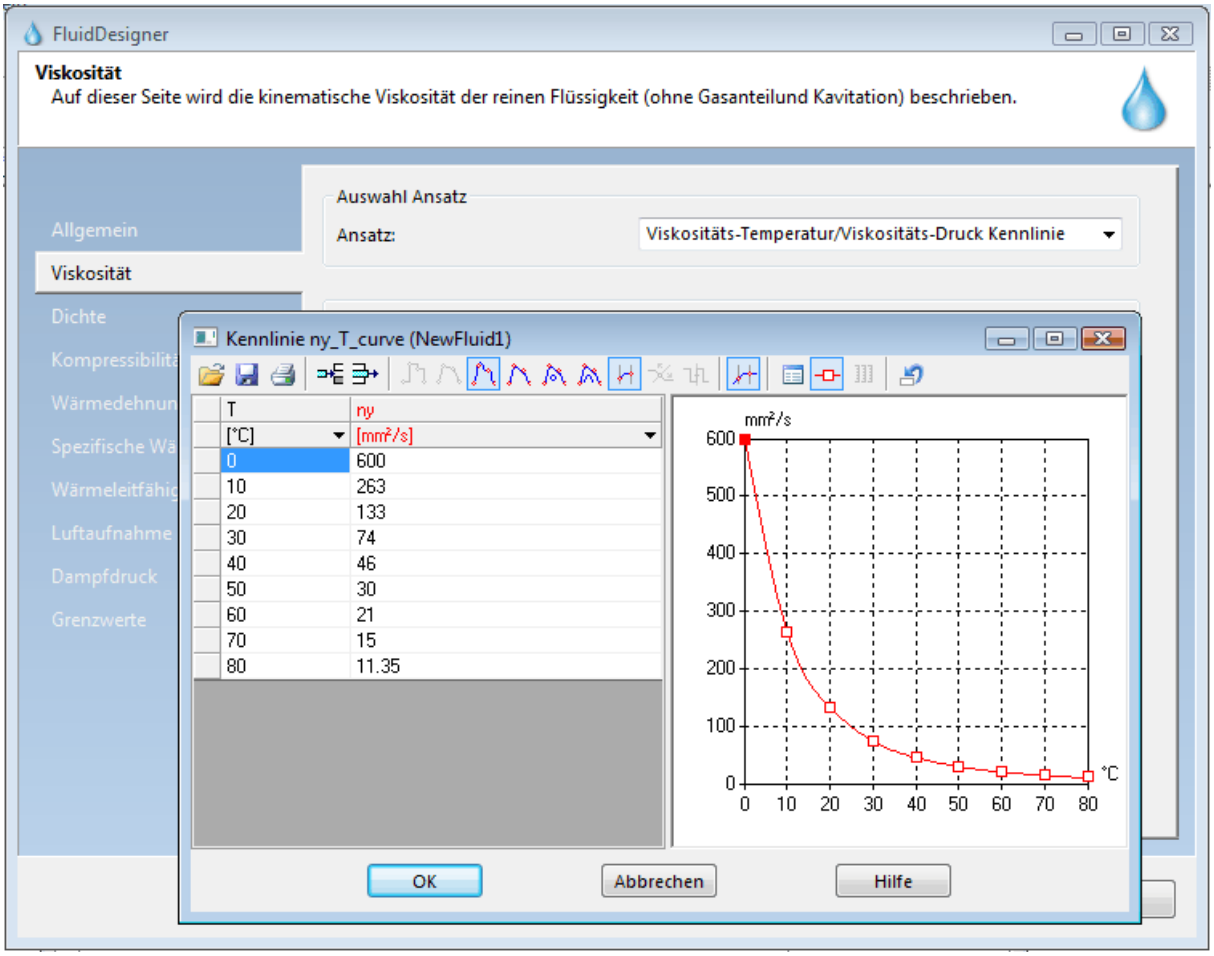



**Bild 8.33:** Seite *Viskosität* mit Viskositäts-Temperatur-Kennlinie

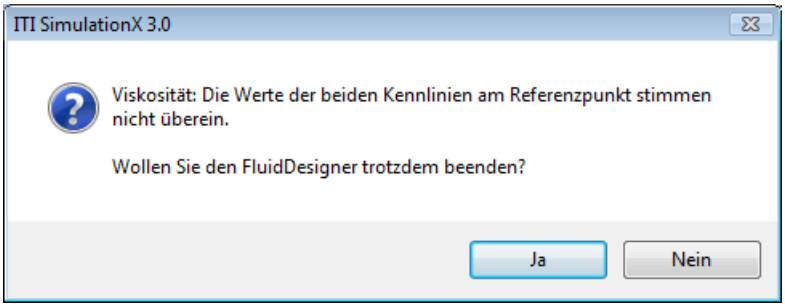
Die beiden Kennlinien $\nu_T(T)$ und $\nu_p(p)$ geben das Viskositäts-Temperatur-Verhalten (an der Stelle $p_{Ref}$ ) bzw. das Viskositäts-Druck-Verhalten (an der Stelle $T_{Ref}$ ) wieder. Aus diesen beiden Kennlinien wird intern ein 2D-Kennfeld $\nu(p,T)$ nach folgender Vorschrift aufgebaut:

$$\nu(p,T) = \frac{1}{\nu_T(T_{Ref})} \cdot \nu_T(T) \cdot \nu_p(p) \qquad \textbf{(8.1)}$$

Voraussetzung für diese Vorgehensweise ist die Bedingung, dass sich beide Kennlinien im Referenzpunkt $(p_{Ref}, T_{Ref})$ schneiden. Dies bedeutet, dass beide Kennlinien an diesem Punkt den gleichen Wert liefern müssen:

$$\nu_T(T_{Ref}) = \nu_p(p_{Ref}) \qquad \textbf{(8.2)}$$

Falls die Bedingung nicht erfüllt ist, erfolgt beim Verlassen des FluidDesigners eine Warnung.




### 8.3.1.3 Dichte

Diese Seite beschreibt die Dichte der reinen Flüssigkeit (ohne Gasanteil oder Kavitation). Es steht folgende Auswahl zur Verfügung:

- Standard-Ansatz
- Beliebiger Ansatz
- Dichte-Temperatur/Dichte-Druck-Kennlinie

Wenn die Fluideigenschaften mit Funktionen oder Kennlinien vorgegeben werden, dann beachten Sie bitte, dass einige Eigenschaften voneinander abhängen. Das ist z. B. für den Kompressionsmodul und die Dichte der Fall. Bei Vorgabe ungenauer Funktionen oder Kennlinien kann die Massebilanz nicht erfüllt werden.

Bei Vorgabe einer Funktion oder Kennlinie für den Kompressionsmodul, wird empfohlen den **Standard-Ansatz** für die Berechnung der Dichte beizubehalten. Die resultierende Dichte (die dann die Einflüsse von Gasanteil und Kavitation zeigt) kann mit den gegebenen Werten verglichen und eventuell angepasst werden.

### 8.3.1.4 Kompressibilität

Auf dieser Seite wird die Kompressibilität der reinen Flüssigkeit (ohne Gasanteil oder Kavitation) beschrieben. Folgende Auswahl ist verfügbar:

- Konstante Kompressibilität
- Standard-Ansatzfunktion mit Koeffizienten
- Beliebige Ansatzfunktion
- Kompressibilitäts-Druck/ Kompressibilitäts-Temperatur-Kennlinie

### 8.3.1.5 Wärmedehnung

Diese Seite beschreibt die Wärmedehnung der reinen Flüssigkeit (ohne Gasanteil oder Kavitation). Die folgende Auswahl ist verfügbar:

- Konstante Wärmedehnung
- Ansatzfunktion
- Wärmedehnungs-Druck/Wärmedehnungs-Temperatur-Kennlinie

### 8.3.1.6 Spezifische Wärmekapazität

Die spezifische Wärmekapazität der reinen Flüssigkeit (ohne Gasanteil oder Kavitation) wird auf dieser Seite mit der folgenden Auswahl festgelegt:

- Konstante Wärmekapazität

- Ansatzfunktion
- Wärmekapazitäts-Druck/Wärmekapazitäts-Temperatur-Kennlinie

#### 8.3.1.7 Wärmeleitfähigkeit

Definieren Sie auf dieser Seite die Wärmeleitfähigkeit der reinen Flüssigkeit (ohne Gasanteil oder Kavitation). Folgende Auswahl steht zur Verfügung:

- Konstante Wärmeleitfähigkeit
- Ansatzfunktion
- Wärmeleitfähigkeits-Druck/Wärmeleitfähigkeits-Temperatur-Kennlinie

#### 8.3.1.8 Luftaufnahme

Auf dieser Seite werden das Gasaufnahmevermögen der Flüssigkeit sowie die Eigenschaften des Gases charakterisiert. Sie können mit Voreinstellungen arbeiten oder beliebige Festlegungen treffen.

#### 8.3.1.9 Dampfdruck

Diese Seite beschreibt den Dampfdruck der Flüssigkeit. Die folgenden Ansätze stehen zur Verfügung:

- Standard-Ansatzfunktion
- Beliebige Ansatzfunktion
- Dampfdruck-Temperatur-Kennlinie

#### 8.3.1.10 Grenzwerte

Auf dieser Seite legen Sie die Grenzwerte für den Druck und die Temperatur fest. Sie können zwischen einer Voreinstellung oder beliebigen Werten wählen.

### 8.3.2 FluidDesigner Pneumatik – Gase

Der Aufruf des FluidDesigners erfolgt analog **8.3.1**, wobei Sie unter der Rubrik Gase ihre Auswahl treffen bzw. hinzufügen.

#### 8.3.2.1 Allgemein

Auf dieser Seite benennen Sie den neu zu erstellenden Gastyp. Des Weiteren können Sie einen Kommentar einfügen (siehe **Bild 8.34**).

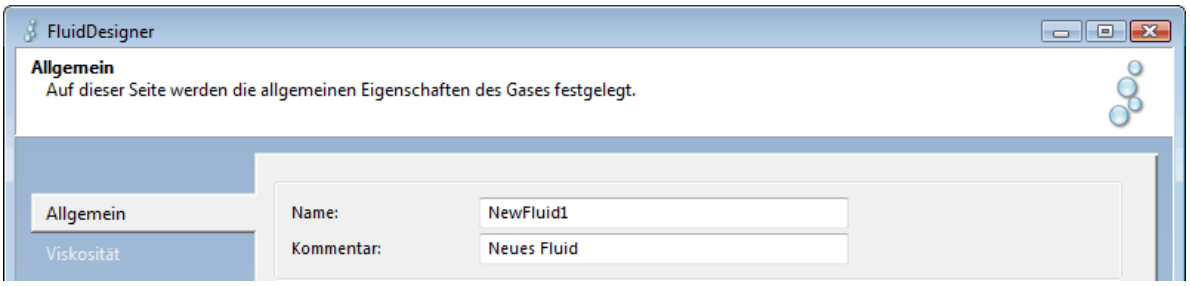



**Bild 8.34:** FluidDesigner für Gase

Der neu erstellte Gastyp wird standardmäßig im Verzeichnis "...*Eigene Dateien**Eigene Fluide**Pneumatics*" gespeichert. Dieses Verzeichnis kann der Anwender jedoch auch über den Dialog ***Extras/Optionen*** auf der Seite ***Verzeichnisse*** ändern. Alle in diesem Verzeichnis befindlichen Fluidtypen werden beim Neustart des Programms automatisch geladen.

#### 8.3.2.2 Viskosität

Auf dieser Seite wird die dynamische Viskosität des Gases beschrieben. Folgende Auswahl ist verfügbar:

- Konstante Viskosität
- Ansatzfunktion nach Sutherland
- Beliebige Ansatzfunktion
- Viskositäts-Druck/Viskositäts-Temperatur-Kennlinie.

#### 8.3.2.3 Kalorische Eigenschaften

Diese Seite beschreibt die kalorischen Eigenschaften des Gases. Dabei werden die spezifische Gaskonstante oder die Molare Masse, sowie verschiedene Ansätze für die spezifische Wärmekapazität berücksichtigt.

#### 8.3.2.4 Wärmeleitfähigkeit

Die Wärmeleitfähigkeit des Gases wird auf dieser Seite definiert. Sie können zwischen

- Konstanter Wärmeleitfähigkeit
- Beliebiger Ansatzfunktion und
- Kennlinie über T auswählen.

#### 8.3.2.5 Spezifische Enthalpie

Diese Seite beschreibt die spezifische Enthalpie des Gases. Zur Auswahl stehen

- Beliebige Ansatzfunktion
- Kennlinie über T

zur Verfügung.

#### 8.3.2.6 Realgas

Die Kennwerte für die Realgas-Modelle werden auf dieser Seite festgelegt.

#### 8.3.2.7 Grenzwerte

Auf dieser Seite legen Sie die Grenzwerte für den Druck und die Temperatur fest. Sie können zwischen einer Voreinstellung oder beliebigen Werten wählen.

### 8.3.3 FluidDesigner *Mischungen*

Sie können Mischungen aus allen verfügbaren Gasen erstellen und einsetzen. Die Benennung der Mischung und das Speichern erfolgt analog **8.3.2.1**.

Auf der Seite "Zusammensetzung" können Sie die einzelnen Komponenten prozentual definieren.

Weitere Erklärungen zu den Mischungen finden Sie im **Library Manual/ Pneumatics.**

# 8.4 Statechart Designer

## 8.4.1 Einführung in die Arbeit mit Statecharts

Steuerungen in Automatisierungseinrichtungen (= Steuereinrichtungen) stellen ein Bindeglied zwischen dem Bediener und der Anlage (z.B. Druckmaschine, Schaltautomatik im Auto oder Haustechnik) dar. Der Bediener löst in einer Leitstelle Kommandos aus, die von der Steuereinrichtung in Steuersignale für die Anlage umgesetzt werden. Die Anlage liefert ihrerseits Daten an die Steuereinrichtung, die dedizierte Abläufe kennzeichnen. Die Steuereinrichtung organisiert diese Abläufe in einer Zustandsmaschine, die zum einen die Bediener-Kommandos und zum anderen die Anlage-Daten (Sensordaten) auswertet (**Bild 8.35**). Außerdem liefert die Steuereinrichtung dem Bediener diverse Status-Informationen.

Während die Bediener-Kommandos in unregelmäßigen Abständen (sporadisch) ausgelöst werden, liefert die Anlage periodisch oder/und aperiodisch Daten an die Steuereinrichtung.
Die Bediener-Kommandos sind im Sinne der Informationstechnik Nachrichten mit diversen Argumenten. Die Anlage-Daten sind gewöhnlich Skalare. In der im Folgenden vorgestellten Zustandsmaschine werden diese Formen der Informationsübermittlung als *Signal-Ereignisse* und *Change-Ereignisse* differenziert behandelt. Zur Organisation von Zeitfolgen werden außerdem *Time-Ereignisse* benutzt, die von der Zustandsmaschine selbst erzeugt werden.

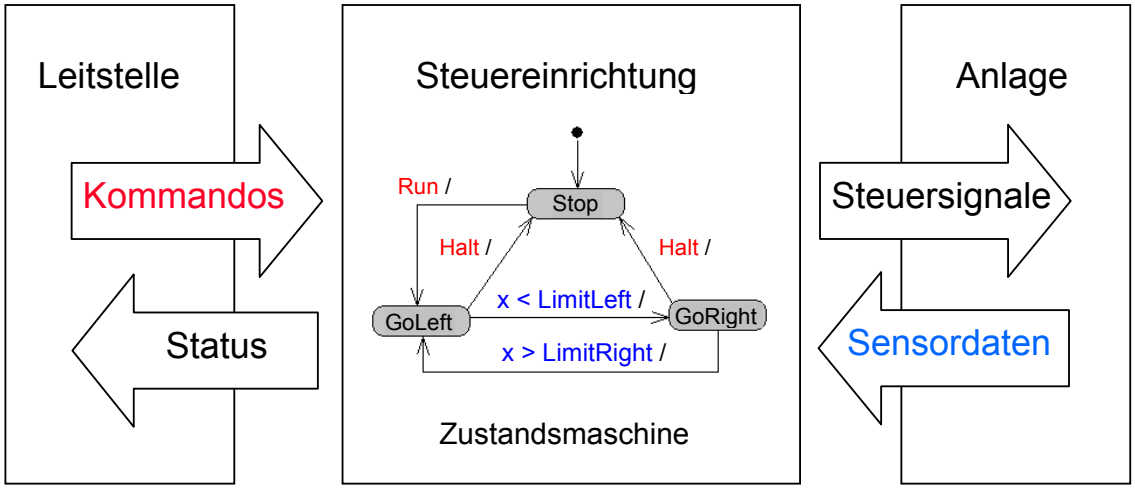



**Bild 8.35:** Stellung der Zustandsmaschine in der Automatisierungstechnik

## 8.4.2 Grafische Notation von Zustandsmaschinen

Die grafische Darstellung der Zustandsmaschine und ihre semantische Interpretation folgen den Definitionen in der **UML 2.1.2 Superstructure Specification [6]**.

Die Basiskomponenten von Zustandsdiagrammen sind Zustände (engl. States) und Transitionen (engl. Transitions) zwischen den Zuständen. Die Folge der Transitionen startet von einem Initial-Zustand (engl. Initial State) mit einer Initial-Transition zum ausgewiesenen Zielzustand. **Bild 8.36** zeigt die Basiskomponenten in ihren Beziehungen.

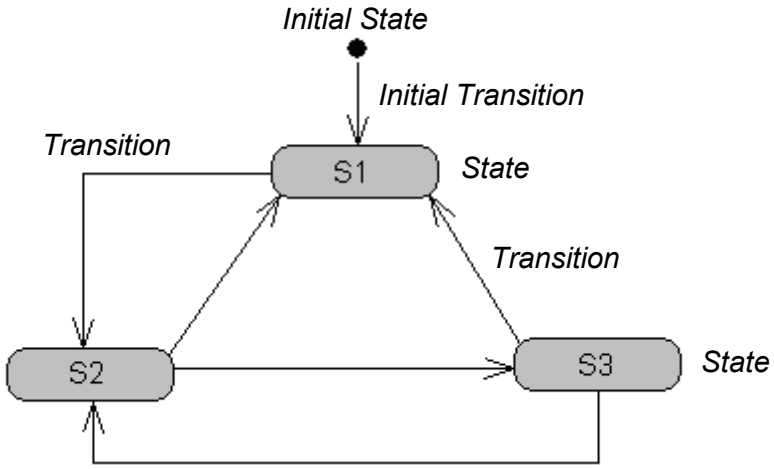



**Bild 8.36:** Elemente der grafischen Darstellung einer Zustandsmaschine

### 8.4.2.1 Zustände / States

Ein Zustand repräsentiert eine Situation, in welcher eine invariante Bedingung gilt. Diese Invariante kann sowohl statisch (z.B. Warten auf ein bestimmtes Ereignis) als auch dynamisch (z.B. Ausführen einer Aktivität) sein.

Folgende Typen von Zuständen werden unterschieden:

- *Simple State,*
- *Composite State,*
- *Submachine State.*

*Simple States* sind im Sinne einer Hierarchie nicht untergliedert, d.h. sie enthalten keine weiteren Zustände. *Composite States* enthalten eine oder mehrere Regionen, die jeweils eine Menge von Zuständen und Transitionen umfassen. Die Regionen repräsentieren parallele Steueroperationen. *Submachine States* inkludieren komplette Sub-Zustandsmaschinen in die aktuelle Zustandsmaschine. *Submachine States* sind semantisch äquivalent zu *Composite States*.

*Submachine States* werden hier nicht betrachtet. Die Untergliederung von *Composite States* wird auf eine einzelne Region beschränkt.

Zustände können aktiv oder inaktiv sein. Ist ein Zustand aktiv, so sind alle *Composite States*, die ihn direkt oder indirekt enthalten, ebenfalls aktiv.

Ein Zustand wird grafisch durch horizontale Linien in mehrere Abschnitte geteilt:

- *Name Compartment* mit dem Namen des Zustandes
- *Internal Activities Compartment* mit internen Aktivitäten
- *Internal Transitions Compartment* mit internen Transitionen.

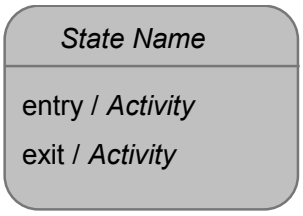



**Bild 8.37:** Zustand mit Namen und internen Aktivitäten

Die internen Aktivitäten werden ausgeführt, wenn der Zustand aktiv ist. Die Schlüsselwörter “entry“ und “exit“ spezifizieren dazu die näheren Umstände.

Die *Entry-Activity* wird einmalig ausgeführt, nachdem der Zustand aktiviert bzw. betreten wurde. Wenn der Zustand verlassen wird, erfolgt die Ausführung der *Exit-Activity*.

### 8.4.2.2 Transitionen / Transitions

Eine Transition beschreibt einen Zustandsübergang, der an folgende Bedingung geknüpft ist:

- der Quellzustand der Transition ist aktiviert,
- ein *Trigger* löst den Zustandsübergang aus und
- eine als *Guard* definierte logische Bedingung ist wahr.

An den Zustandsübergang kann wahlfrei eine Aktivität (engl. Activity) gekoppelt werden, die beim Durchlaufen der Transition ausgeführt wird. Diese Aktivität ist „zeitlos“ und „ununterbrechbar“. Das Durchlaufen einer Transition nimmt keine Zeit in Anspruch. Besitzt eine Transition keinen expliziten Trigger, so wird ihr implizit ein *Completion-Trigger* zugordnet.

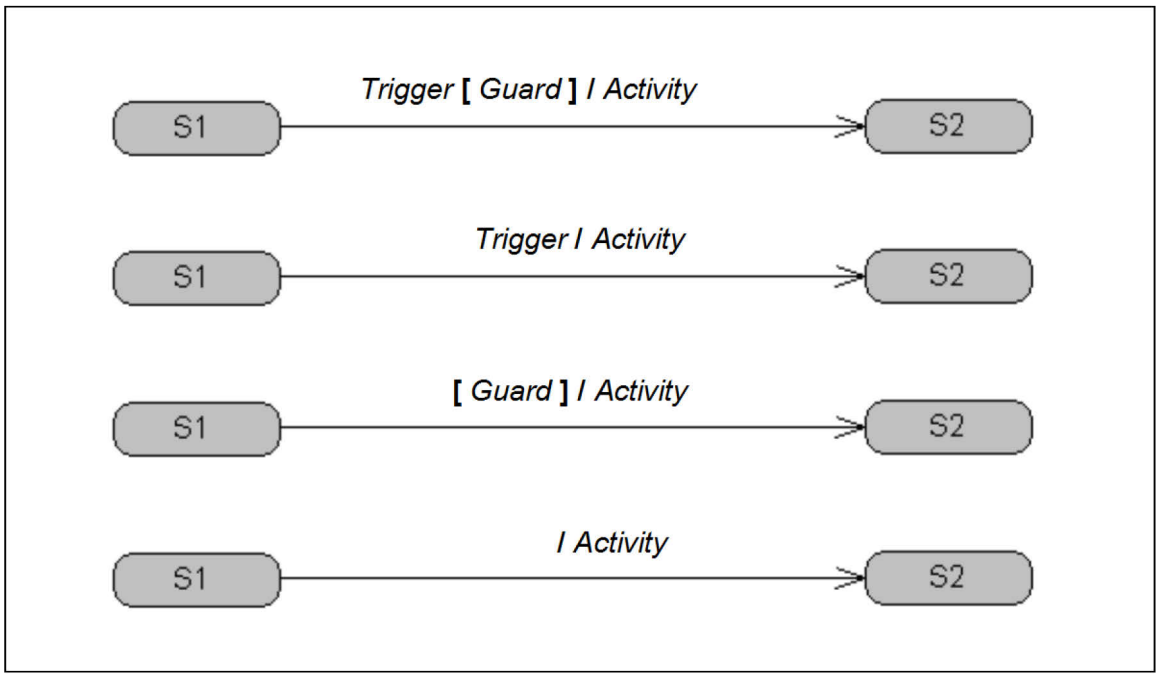



**Bild 8.38:** Varianten der Zustandsübergänge

Im Folgenden werden betrachtet:

- *Signal-Trigger*
- *Change-Trigger*
- *Time-Trigger*
- *Completion-Trigger.*

*Signal-Trigger* lösen bei Empfang erwarteter Signale aus.

*Change-Trigger* lösen aus, wenn korrespondierende Boole'sche-Relationen von Systemvariablen mit TRUE evaluiert werden.

*Time-Trigger* lösen nach dem Ablauf einer vorgegebenen Dauer aus.

*Completion-Trigger* sind implizite Trigger, die auslösen, wenn States ihre *Activities* abgeschlossen haben.

### 8.4.2.3 Pseudozustände / Pseudostates

Pseudozustände sind instationäre Zustände der Dauer Null. Sie werden genutzt, um den Initial-Zustand und komplexe Zustandsübergänge zu definieren.  zeigt die im Kontext von industriellen Steuerungen verwendeten Pseudozustände.

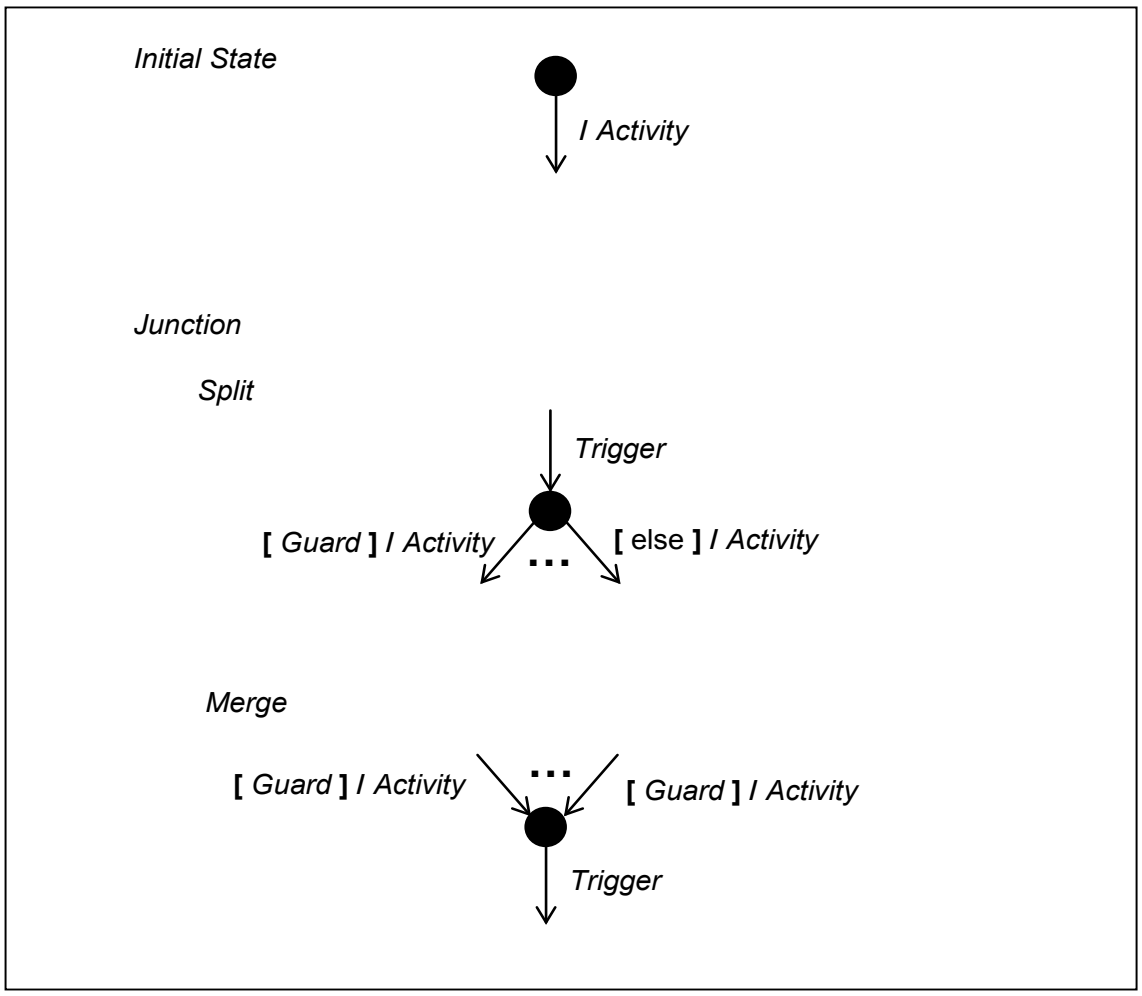



**Bild 8.39:** Pseudozustände

Der *Initial Pseudostate* besitzt zwei Verwendungen:

- Er ist der Einstiegspunkt des Zustandsautomaten.
  Von ihm geht einzig die Initial-Transition zum ersten Zustand ab. Die Initial-Transition besitzt keinen *Trigger* oder *Guard*, kann jedoch mit einer *Activity* bewertet werden.
- Er definiert den *Default Entry* in einen *Composite State*.
  Siehe dazu Abschnitt **8.4.2.8**.

*Junction Pseudostates* sind Verbindungspunkte, die eine einlaufende Transition

- in mehrere auslaufende Transitionen aufteilen (engl. split)

oder mehrere einlaufende Transitionen in eine auslaufende Transition vereinen (engl. merge).

### 8.4.2.4 Trigger

**Signal-Trigger**

Ein *Signal-Trigger* löst eine Transition aus, wenn ein erwartetes Signal-Ereignis eintritt. Das Signal-Ereignis ist ein asynchrones Ereignis wie z.B. der Empfang einer Nachricht. Asynchron heißt, dasjenige Subsystem, welches das Signal-Ereignis bewirkt hat, arbeitet nach dem Senden des Signals sofort weiter, ohne auf die Reaktion des Empfängers zu warten.

Mit dem Signal-Ereignis ist die Zuweisung von Argumenten, die optional mit dem Signal übertragen werden, an korrespondierende Attribute bzw. Variable des Zustandsautomaten verbunden.

*Signal-Trigger* werden mit dem Namen des auslösenden Ereignisses und der Aufzählung der Argumente notiert:

*signal-trigger ::= event-name [* **(** *attribute-list* **)** *]*

*attribut-list ::= attribut-name* **:** *type-name [* **,** *attribut-name* **:** *type-name ]**

Die Zuweisung der Argumente an die Attribute bzw. Variable des Zustands-automaten erfolgt nur, wenn die korrespondierende Transition tatsächlich feuert (siehe **8.4.2.5**).

**Change-Trigger**

Ein *Change-Trigger* löst eine Transition in Folge eines Change-Ereignisses aus. Das Change-Ereignis wird generiert, wenn ein korrespondierender Boole'scher Ausdruck mit TRUE evaluiert wird. Der Boole'sche Ausdruck schließt optional Input-Variable, interne Variable, Parameter und Konstanten ein.

Notation:

*change-trigger* ::= *boolean-expression*
oder
*change-trigger* ::= **when** *boolean-expression*

Im Kontext von industriellen Steuerungen gelten folgende zusätzliche Festlegungen:

- Der Boole'sche Ausdruck wird ständig evaluiert.
- Das Change-Ereignis wird ausgelöst, wenn der Boole'sche Ausdruck zum ersten Mal TRUE liefert, und existiert, solange für den Boole'schen Ausdruck TRUE gilt.

**Time-Trigger**

Ein *Time-Trigger* spezifiziert einen Zeitpunkt, zu dem ein Time-Ereignis generiert wird, und löst auf dieses Ereignis hin die Transition aus.

Notation:

*time-trigger* ::= **after (** *time-expression* **)**

Der Zeitausdruck für das Time-Ereignis ist relativ zum Zeitpunkt der Aktivierung des Quellzustandes der Transition. Das Time-Ereignis wird nach Ablauf der berechneten Dauer nur generiert, wenn der Quellzustand der Transition noch aktiv ist.

**Completion-Trigger**

Einer Transition ohne Trigger wird implizit ein *Completion-Trigger* zugewiesen, der in Folge eines Completion-Ereignisses auslöst.

Das Completion-Ereignis wird generiert, wenn die *Entry-Activities* des Quellzustandes der Transition abgeschlossen worden sind. Besitzt der Quellzustand keine *Entry-Activities* wird das Completion-Ereignis sofort nach Aktivierung des Quellzustandes generiert.

Fehlen die Trigger bei mehreren abgehenden Transitionen eines Zustandes, dann sind diese Transitionen durch sich gegenseitig ausschließende *Guards* zu separieren.

### 8.4.2.5 Guards

Ein *Guard* ist eine zusätzliche Boole'sche Bedingung für das Feuern einer Transition.

Der *Guard* wird evaluiert, wenn das Trigger-Ereignis der Transition eintritt. Falls der *Guard* zu diesem Zeitpunkt FALSE liefert, feuert die Transition nicht und das Trigger-Ereignis wird gelöscht. Bei *Change-Triggern* gilt die Besonderheit, dass die Trigger-Ereignisse solange existieren, wie die korrespondierenden Bedingungen TRUE sind.
Notation:

*guard* ::= **[** *boolean-expression* **]**

Die Boole'sche Bedingung erfasst Attribute bzw. Variable des Zustandsautomaten und gegebenenfalls Argumente des Trigger-Ereignisses. Die Argumente des Trigger-Ereignisses werden nur auf Attribute des Zustandsautomaten abgebildet, wenn die Transition tatsächlich feuert.

### 8.4.2.6 Prioritäten

Falls mehrere von einem Zustand abgehende Transitionen gleichzeitig gefeuert werden können, ist eine Entscheidung notwendig, nur eine von diesen Transitionen zu feuern. Für diese Entscheidung werden den Transitionen Prioritäten in Form von Nummern 1, 2, 3, … zugeordnet. Sind mehrere dieser Prioritätsnummern identisch, erfolgt zwischen den korrespondierenden Transitionen eine zufällige Auswahl.

Die Prioritätsnummer wird an der Transition auf der Seite des Quellzustandes notiert.

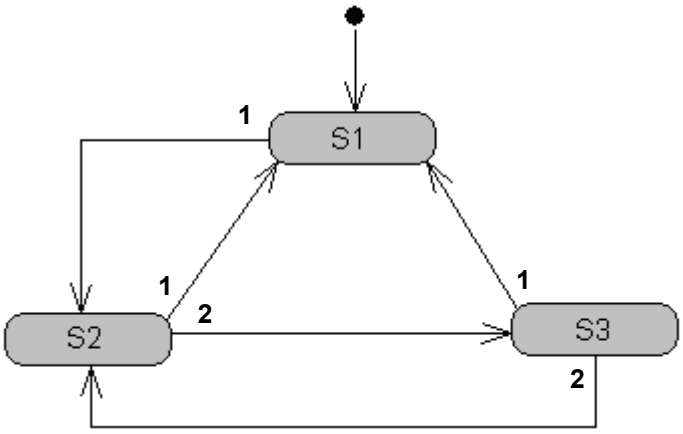



**Bild 8.40:** Priorität von Transitionen

Die Vergabe von Prioritäten für das Feuern von Transitionen ist eine Ergänzung des UML-Standards. Dieser beschränkt sich auf eine "zufällige" Auswahl gemäß der Reihenfolge der grafischen Notation der Transitionen.

### 8.4.2.7 Self-Transitionen und interne Transitionen

*Self*-Transitionen sind Transitionen mit identischem Quell- und Zielzustand. Beim Feuern der Transition werden in der angegebenen Reihenfolge ausgeführt: *Exit-Activity* des Zustandes, *Transition-Activity*, *Entry-Activity* des Zustandes. Besitzt der Quellzustand der *Self-Transition* zusätzlich eine abgehende Transition mit *Time-Trigger*, bewirkt das Feuern der *Self-Transition* infolge des Aus- und Wiedereintritts in den Zustand den Restart des Zeitgebers.

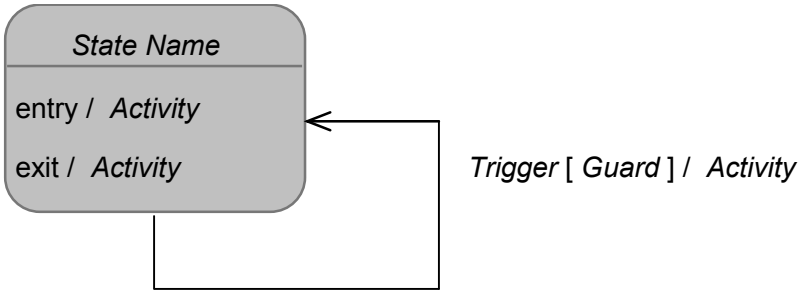



**Bild 8.41:** Self-Transition

Interne Transitionen werden innerhalb eines Zustandes mit *Trigger*, *Guard* und *Activity* notiert. *Guards* sind optional. Interne Transitionen werden ausgeführt, wenn der Zustand aktiv ist und die Übergangsbedingungen erfüllt sind. Im Gegensatz zu den *Self*-Transitionen verursacht das Feuern der internen Transitionen keine *Entry*- und *Exit-Activities* des Zustandes. Die Transitionen können mehrfach das gleiche Trigger-Ereignis benutzen, der *Guard* spezifiziert dann die Auswahl der zu feuernden Transition. Falls das Trigger-Ereignis Parameter enthält, können diese in der *Transition-Activity* ausgewertet werden.

In Bezug auf die externen Transitionen, inklusive *Self*-Transitionen, sind die internen Transitionen niederprior. Wenn eine externe Transition und eine interne Transition gleichzeitig feuerbar sind, dann erhält die externe Transition den Vorrang. Außerdem gilt: Die Übergangsbedingungen der internen Transitionen werden in der Reihenfolge ihrer Notation von oben nach unten geprüft.

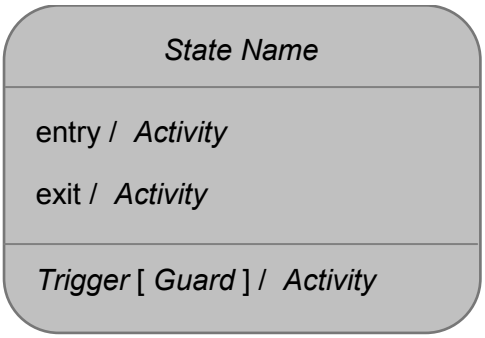



**Bild 8.42:** Interne Transition

### 8.4.2.8 Zusammengesetzte Zustände / Composite States

Zusammengesetzte Zustände (engl. Composite States) zeichnen sich gegenüber den einfachen Zuständen durch einen zusätzlichen Dekompositionsabschnitt aus.

Der zusammengesetzte Zustand kann damit folgende Abschnitte enthalten (**Bild 8.43**):

- *Name Compartment* mit dem Name des Zustandes
- *Internal Activities Compartment* mit internen Aktivitäten
- *Internal Transitions Compartment* mit internen Transitionen
- *Decomposition Compartment* mit Subzuständen.

Der Dekompositionsabschnitt kann eine oder mehrere orthogonale Regionen umfassen, die jeweils eine Menge von Zuständen, Pseudozuständen und Transitionen enthalten.

Im hier definierten Subset wird der Dekompositionsabschnitt auf eine einzige Region beschränkt. Der zusammengesetzte Zustand erhält damit das Attribut “nicht-orthogonal“.

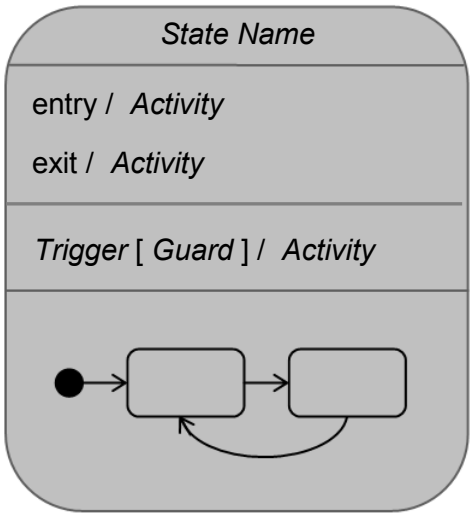



**Bild 8.43:** Zusammengesetzter Zustand

Im Dekompostionsabschnitt sind alle Arten der bisher eingeführten Zustände möglich. Ein Subzustand kann wiederum ein zusammengesetzter Zustand sein.

Interne Transitionen des zusammengesetzten Zustandes gelten auch für alle Subzustände. Wird solch eine Transition getriggert, so wird sie innerhalb des zu diesem Zeitpunkt aktiven Subzustands ausgeführt, ohne diesen zu verlassen. Die Transitionen eines hierarchisch tiefer gelegenen Subzustandes haben Priorität vor den Transitionen des umfassenden Zustands (lower-first priority).

Beim Eintritt in den nicht-orthogonalen *Composite State* werden folgende Fälle unterschieden **(Bild 8.44**):

- *Default Entry:* Die einlaufende Transition endet auf dem äußerem Rand des *Composite State*. Die Initial-Transition, die vom Initial-Zustand ausgeht, aktiviert den Zielzustand. An die Initial-Transition kann eine *Activity* gekoppelt werden, sie hat jedoch keine Bewertung durch einen *Trigger* oder *Guard*. Die *Entry-Activity* des *Composite State* wird vor der *Transition-Activity* der Initial-Transition ausgeführt.
- *Explicit Entry:* Die einlaufende Transition führt zu einem Subzustand des *Composite State*. Dieser Subzustand wird aktiviert, seine *Entry-Activity* wird nach der *Entry-Activity* des *Composite State* ausgeführt.

- *Shallow History Entry:* Die einlaufende Transition führt zu einem *Shallow History Pseudostate* (flache Historie). Es wird der zuletzt aktive Subzustand des *Composite State* reaktiviert. Für den Fall, dass der *Composite State* das erste Mal aktiviert wird, folgt der Eintritt in den *Default History State.* Der *Default History State* ist ein Subzustand, der das Ziel der einzigen Transition ist, die vom *History Pseudostate* ausläuft. Die *Entry-Activity* des *Composite State* wird vor *der Entry-Activity* des reaktivierten Substate ausgeführt.

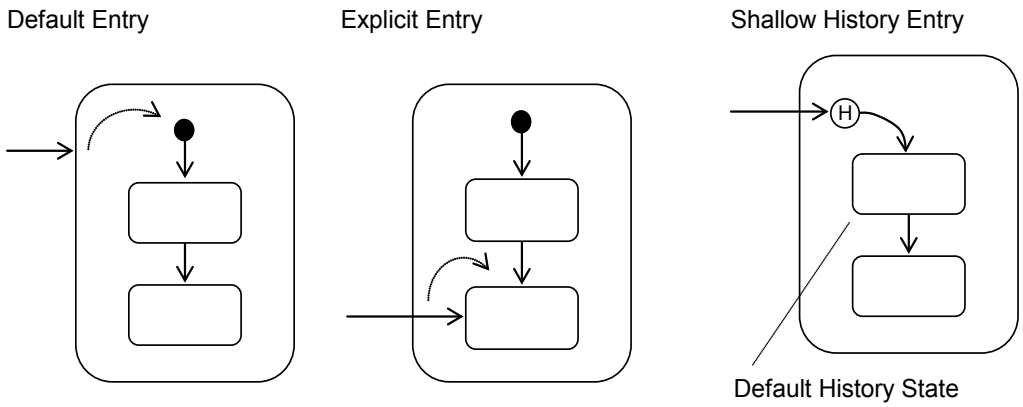



**Bild 8.44:** Eintrittsvarianten in *Composite States*

Der Austritt aus dem nicht-orthogonalen *Composite State* erfolgt durch eine High-Level-Transition, die vom äußeren Rand des *Composite State* ausgeht, oder eine Low-Level-Transition, die von einem Subzustand zu einem Zustand außerhalb des *Composite State* führt (**Bild 8.45**). Der Austritt aus dem *Composite State* erfolgt rekursiv beginnend mit dem innersten aktiven Subzustand. Dabei werden die *Exit-Activities* der Zustände in aufsteigender Hierarchie ausgeführt.

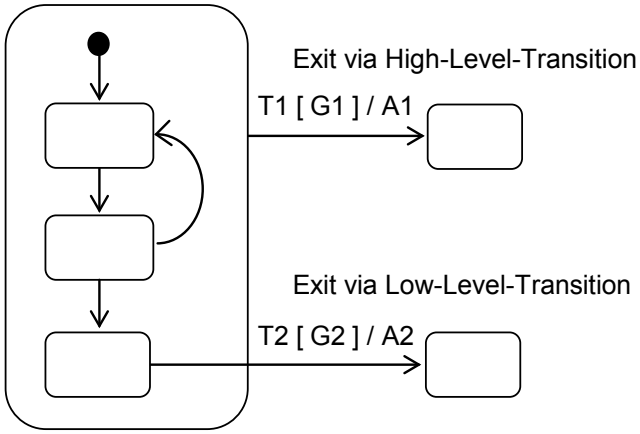



**Bild 8.45:** Austrittsvarianten aus *Composite States*

Nach den UML-Festlegungen haben die *Low-Level-Transitionen* eine höhere Priorität als *High-Level-Transitionen*.

In einer aktiven Konfiguration (**Bild 8.46**) besitzen demnach die *Low-Level-Transitionen t1, t2, t3* und *t4* Priorität gegenüber der *High-Level-Transition t0*.

Könnten eine Transition $tj \in (t1, \ldots t4)$ und *t0* zum identischen Zeitpunkt per *Signal-Trigger T0, T1, ... T4* schalten, dann schaltet *tj*. Der Trigger *T0* verliert seine Wirkung.

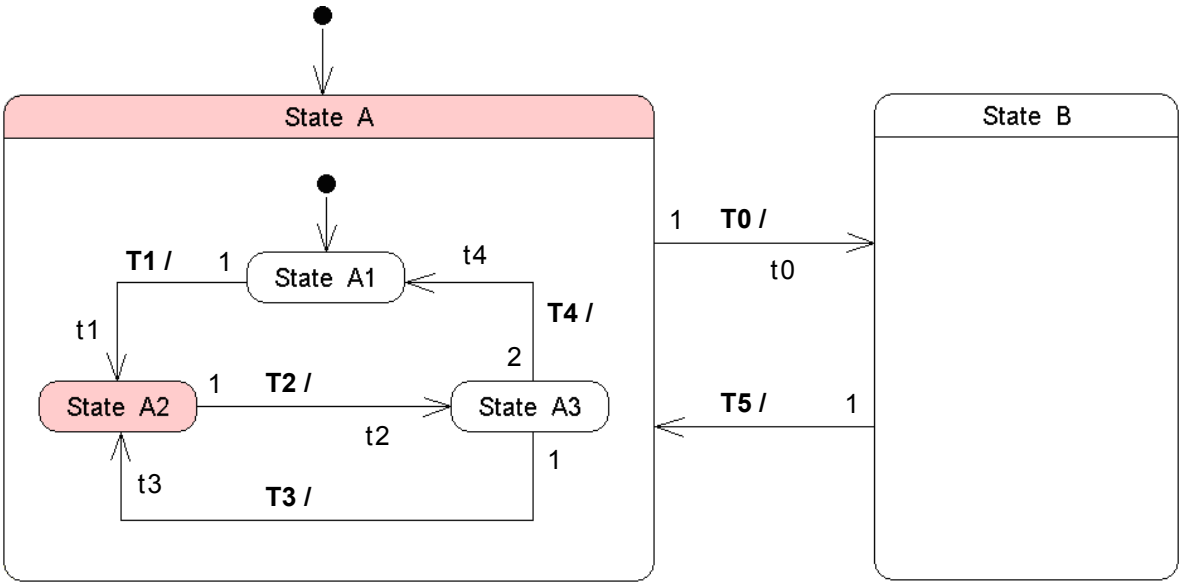



**Bild 8.46:** Aktive Konfiguration *(State A, State A2)* mit *Signal-Trigger T0* der *High-Level-Transition t0*

Um über eine *High-Level-Transition* eine Notfunktion einleiten zu können, wird zusätzlich vereinbart:
Die Vergabe der Prioritätsnummer 0 an eine *High-Level-Transition* hebt diese auf das Niveau einer *Highest-Priority-Transition* für den *Composite State* und alle seiner Subzustände.

Die *Highest-Priority-Transition* (**Bild 8.47**) schaltet sofort und deaktiviert sowohl den aktiven Subzustand als auch den *Composite State* selbst, wenn der korrespondierende Trigger auslöst.

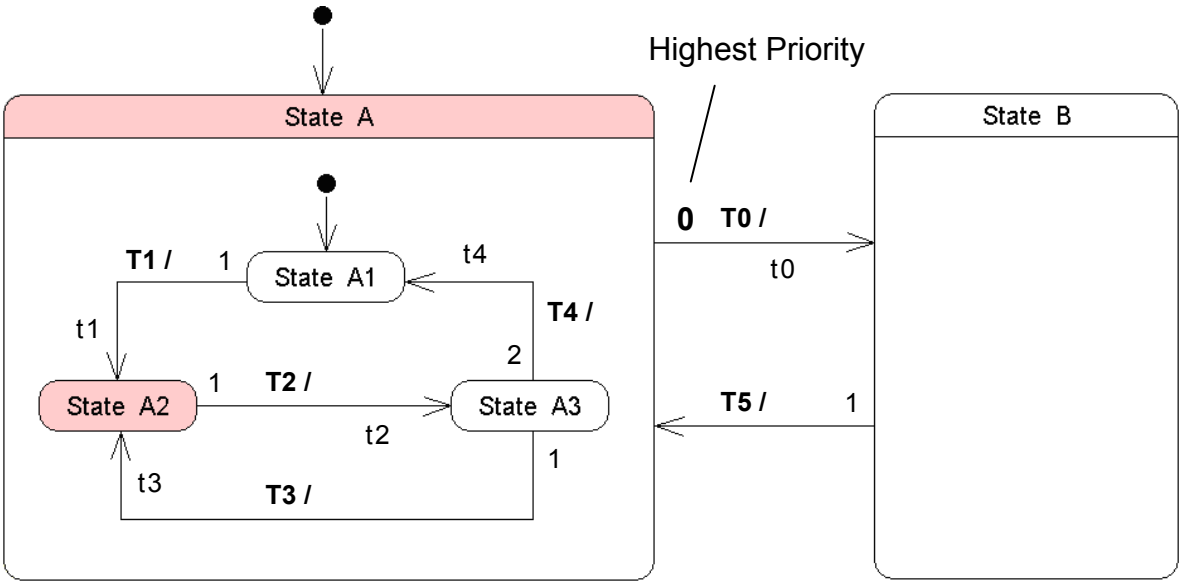



**Bild 8.47:** Festlegung der *High-Level-Transition t0* als *Highest-Priority-Transition*

Einschränkungen zu UML

- Keine Self-Transitionen
- Kein Entry Point Pseudostate, kein Exit Point Pseudostate
- Kein Termination Pseudostate
- Kein Final State
- Kein Deep History Entry

## 8.4.3 Statechart Designer in SimulationX

Der Statechart Designer ist in den TypeDesigner eingebunden und stellt eine spezielle Form der Verhaltensbeschreibung dar. Er kann zur Modellierung von zeit- und zustandsdiskreten Kontrollalgorithmen in SimulationX genutzt werden. Die Modelle werden grafisch auf der Basis von UML-Zustandsdiagrammen aufgebaut und mittels des integrierten Statechart-Compilers automatisch in Modelica-Code übersetzt. Weiterhin können Zustände und Transitionen in SimulationX dynamisch visualisiert werden.

### 8.4.3.1 Aufruf und Oberfläche

Der Statechart Designer wird auf der Seite „Verhalten“ des TypeDesigners über den Auswahlbox-Eintrag „Statechart Diagramm“ gestartet.

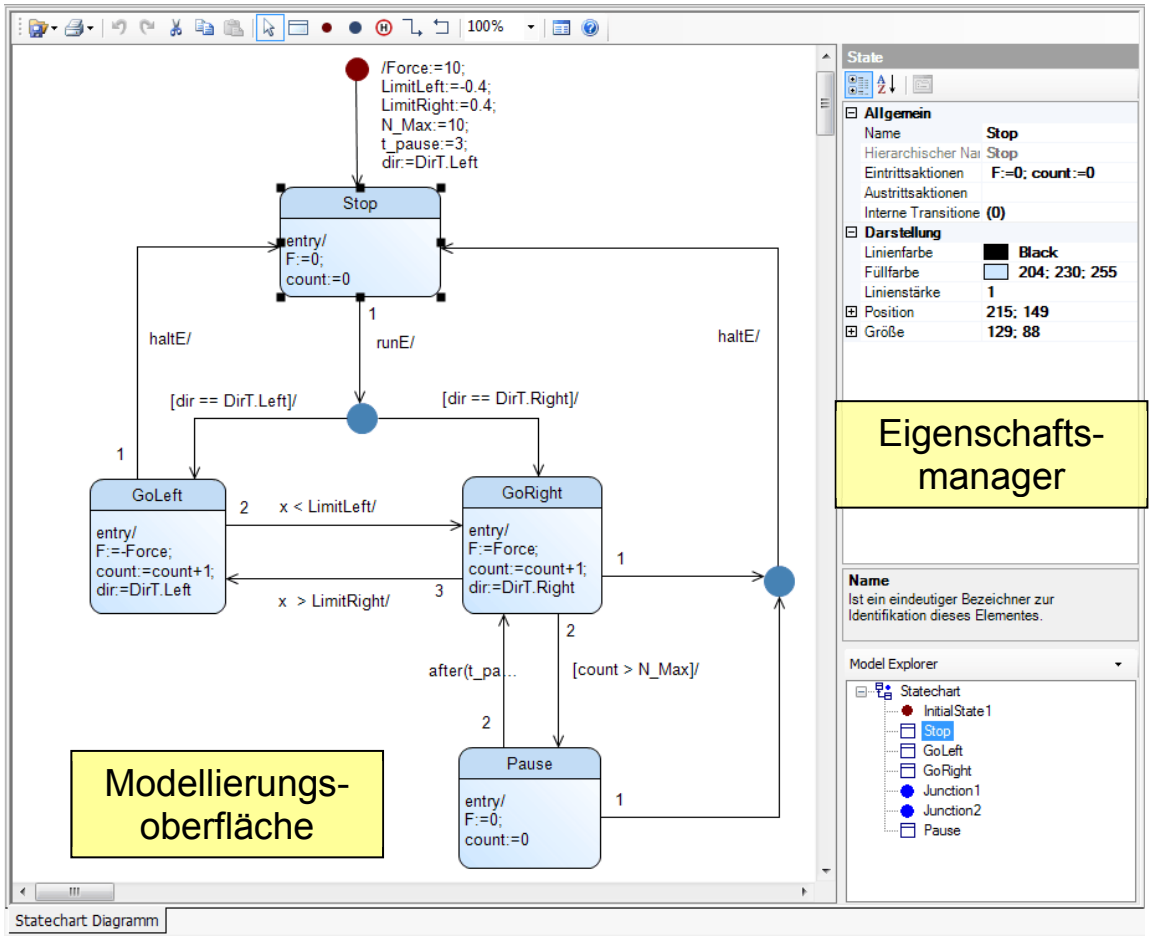


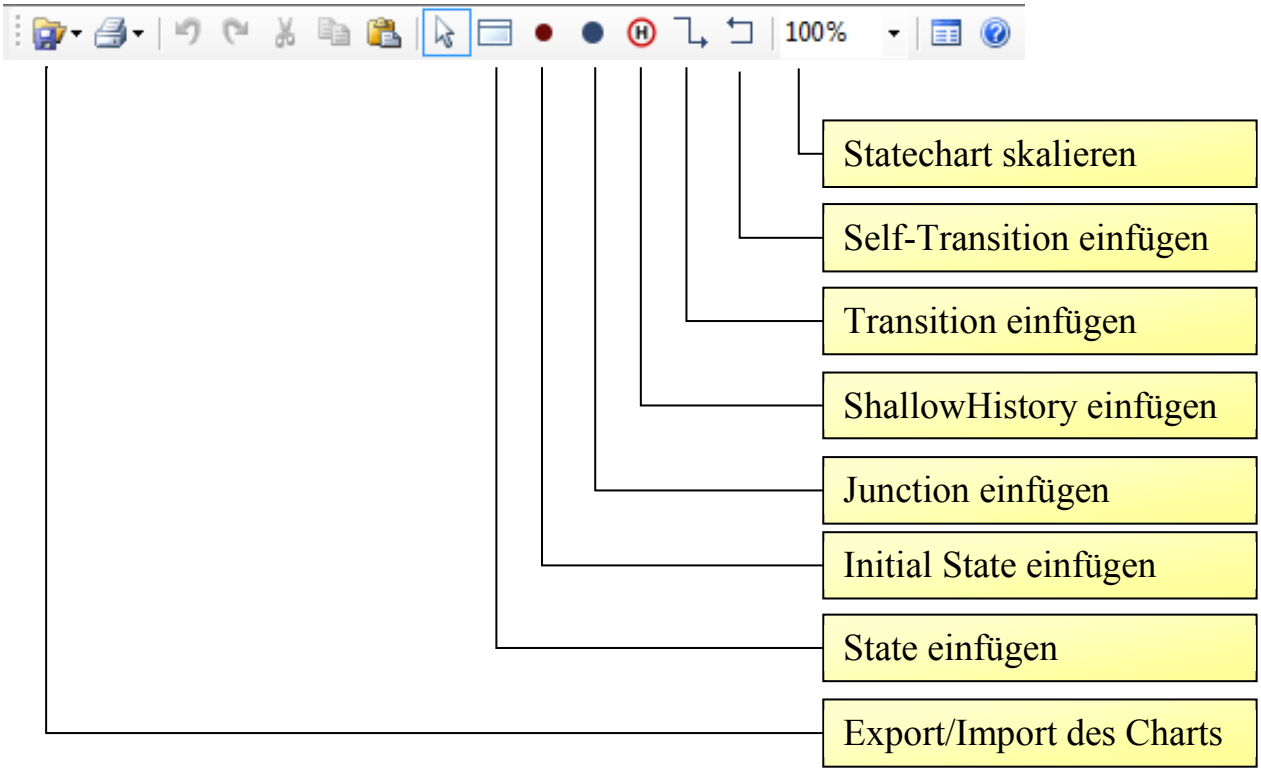

**Bild 8.48:** Statechart-Arbeitsblatt

Für die Unterstützung des Modellierungsprozesses ist die Oberfläche des Statechart Designers in 3 Bereiche unterteilt. Die Modellierungsfläche ist der zentrale Bereich, auf dem die Zustandsdiagramme erstellt werden.

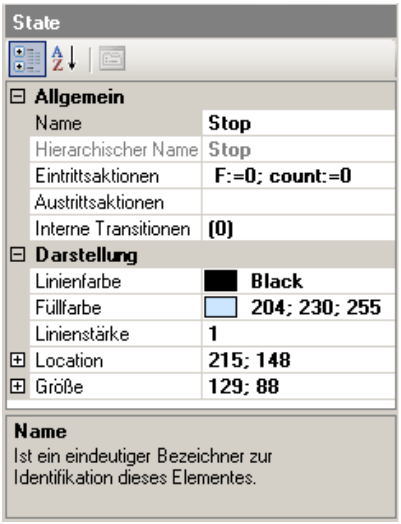
Daneben gibt es noch den Eigenschafts-Manager, der die entsprechenden Eigenschaften eines ausgewählten Elementes darstellt. Darunter befindet sich der Arbeitsbereich, der den Modell-Explorer bzw. die Signal-Event-Definition Tabelle anzeigt. Im Modell-Explorer wird der Aufbau des Modells hierarchisch in einer Baumstruktur visualisiert.

### 8.4.3.2 Modellierung

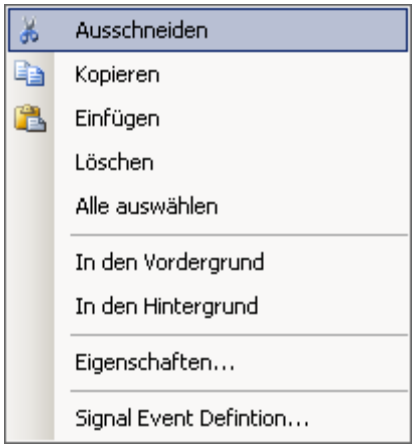
Über die Symbolleiste lassen sich die verschiedenen Notationselemente eines Statechart auswählen. Ist ein Element (z.B. Zustand) selektiert, kann es mit Hilfe der Maus auf der Modellierungsfläche gezeichnet werden.




Über den Eigenschafts-Manager können das ausgewählte Element und seine Darstellung (z.B. Größe, Füllfarbe) dann detaillierter spezifiziert werden. Die Art des ausgewählten Elements bestimmt, welche Eigenschaften dargestellt werden. Modelica-Code zum Spezifizieren von z.B. Eintrittsaktionen kann direkt eingegeben werden, oder bei längeren Codeabschnitten auch in das Eingabefenster, das sich über die Schaltfläche öffnen lässt. Diese Schaltfläche wird sichtbar, wenn Sie die entsprechende Zeile anklicken.



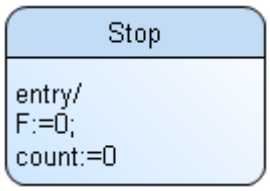
Der eingegebene Modelica-Code wird automatisch auf seine Korrektheit geprüft, gegebenenfalls wird eine entsprechende Meldung ausgegeben.




Alle Notationselemente können auch über ein Kontextmenü bearbeitet werden.

Während des Modellierens werden automatisch bestimmte Design-Richtlinien überprüft, um die UML- Kompatibilität des Statecharts sicherzustellen. Im Folgenden wird das Handling der verschiedenen unterstützen UML- Notations-elemente beschrieben.

### *Einen Zustand (State) hinzufügen*

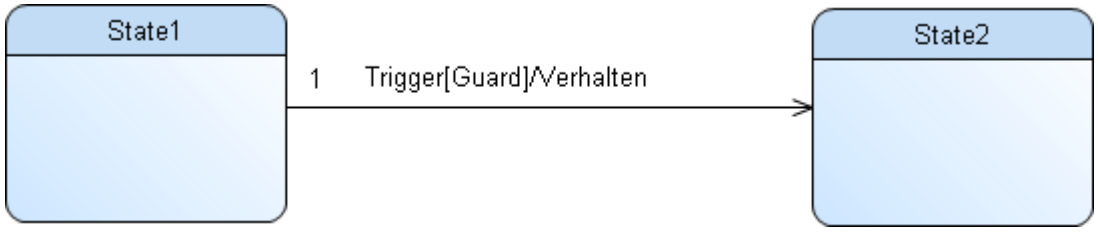



Durch das Klicken in die Modellierungsfläche des Statechart Designers kann ein State mit einer vorgegebenen Standardgröße erzeugt werden. Alternativ lässt sich ein State auch über das Ziehen der Maus, bei gedrückter linker Maustaste, er-stellen.

### *Eine Transition erstellen*

Durch das Klicken in den Ausgangszustand und Ziehen der Maus, bei gedrückter linker Maustaste, in den Zielzustand kann eine Transition erzeugt werden.

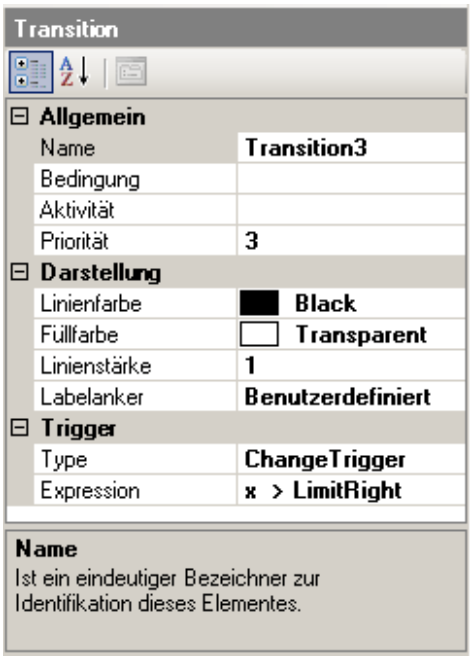
Eine Transition wird durch eine durchgezogene gerichtete und üblicherweise be-schriftete Kante abgebildet.



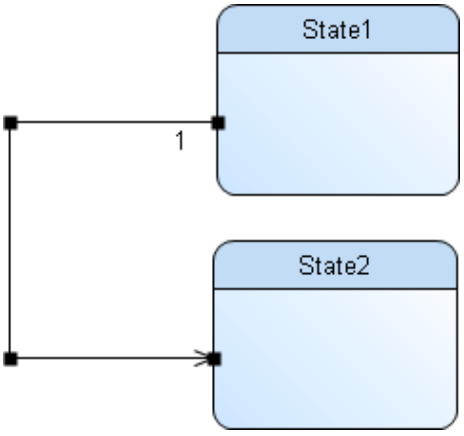
Die Beschriftung (Label) einer Transition kann die folgenden Elemente enthalten:

- Trigger
- Bedingung (Guard) ; siehe **8.4.2.5**
- Verhalten (Aktivität) ; siehe **8.4.2.4**
- Priorität ; siehe **8.4.2.6**

Alle beschriebenen Elemente einer Transition lassen sich im Eigenschafts-Manager spezifizieren. Die Beschriftung (Label) einer Transition kann beliebig, mit Hilfe der Maus, verschoben und in der Größe verändert werden. Zusätzlich lässt sich die Verankerung der Beschriftung an der Transition noch durch die Einstellung *Labelanker* im Eigenschafts-Manager festlegen.

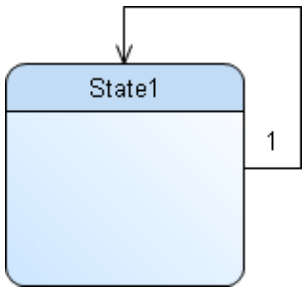



Klickt man mit der Maus auf einen beliebigen Punkt der Transition, wird an dieser Stelle ein neuer Markierungspunkt eingefügt, mit dem das Aussehen der Transition geändert werden kann. Über das Kontextmenü ist es möglich, einen so erzeugten Markierungspunkt wieder zu entfernen.



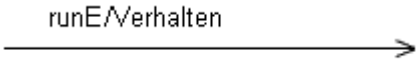
### *Eine Self-Transition erstellen*

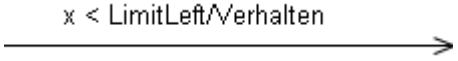
Durch das Ziehen der gedrückten Maus (linke Maustaste) aus dem Zustand heraus, wird automatisch eine Self-Transition erzeugt.

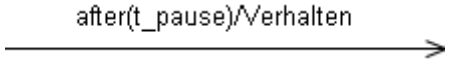



### *Trigger*

Im StateChart Designer werden folgende Trigger explizit unterstützt:

- SignalTrigger: 

- ChangeTrigger: 

- TimeTrigger: 


Der Trigger einer Transition lässt sich im Eigenschafts-Manager spezifizieren, wenn die entsprechende Transition selektiert ist. Über die Type-Auswahlbox lässt sich die Art des Triggers festlegen. Die Bedeutung des Expression-Ausdrucks ist abhängig von der Triggerart. Bei der Auswahl eines SignalTriggers wird automatisch, falls noch nicht vorhanden, ein Default-SignalEvent erzeugt (siehe Abschnitt ***Definieren von Signal Events***).

### *Pseudozustände (Pseudostates) erstellen*

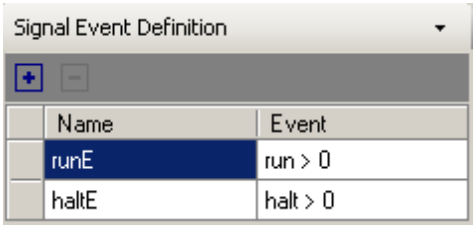
Pseudozustände werden durch ein Klicken auf die Modellierungsfläche erzeugt und sind in ihrer Größe nicht veränderbar. Im Folgenden werden einige Hinweise zu den Pseudozuständen gegeben:

- Die vom Startpunkt ausgehende Transition hat weder einen Trigger noch eine Bedingung. Jeder Startzustand darf höchstens eine ausgehende Transition besitzen.
- Pro Compound Transition gibt es nur **einen** Trigger.
- Mit Historienzuständen ist es möglich, in einem zusammengesetzten Zustand den Zustand, der bei Betreten als Erster aktiv werden soll, dynamisch zu bestimmen.

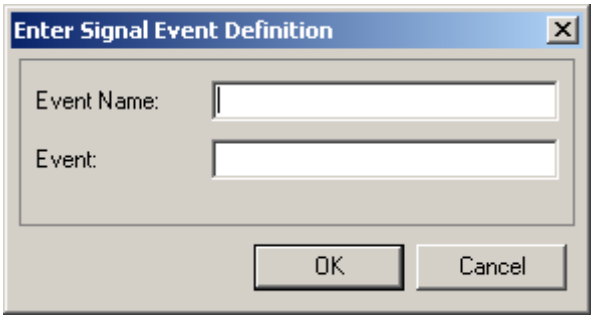
- Als Unterzustände eines zusammengesetzten Zustands sind alle Arten von Zuständen möglich.

### *Definieren von Signal Events*

Für die Verwendung eines SignalTriggers ist es notwendig, ein entsprechendes Signal Event vorher zu definieren. Im Arbeitsbereich ist es möglich, Signal Event Definitionen hinzuzufügen bzw. zu entfernen.




Beim Aktivieren des Plus-Knopfes öffnet sich ein Dialog zum Definieren eines Signal Events. Es ist aber auch möglich, die Eingaben direkt in der Tabelle vorzunehmen. Alle Eingaben werden automatisch auf ihre Gültigkeit hin überprüft.



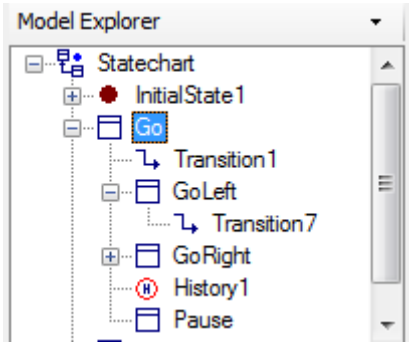

Event Name: Name des Events, der im Label der Transition angezeigt wird (runE)
Event: Boole'scher Ausdruck (z.B. run > 0)

Über das Auslösen des Minus-Knopfes lassen sich markierte Signal Events wieder löschen.
Weitere Informationen zum Modellieren von Zustandsautomaten finden Sie in den entsprechenden **Tutorial 11** und **Tutorial 12**.

### Model Explorer

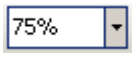
Der Modell-Explorer im Statechart Designer zeigt alle im Modell verwendeten Transitionen und Zustände (inklusive der Pseudozustände), sowie deren hierarchischen Aufbau und erleichtert so das Navigieren in komplexen Modellen. Wird ein Zustand im Modell-Explorer markiert, wird das Modell gegebenenfalls so verschoben, dass der entsprechende Zustand auf der Modellierungsfläche sichtbar wird. Wird ein Zustand auf der Modellierungsfläche markiert, so wird dieser auch

im Modell-Explorer selektiert. Über das Kontextmenu können einzelne Transitionen und Zustände gelöscht werden.




### Zoom

Um die Übersichtlichkeit komplexer Modelle zu erhöhen, lässt sich über die Zoomfunktion der angezeigte Bereich der Modellierungsfläche vergrößern bzw. verkleinern.




### Export und Import von Modellen

Im Statechart Designer erstellte Modelle lassen sich über die Exportfunktion (Symbolleiste ) auf einem Datenträger im Binärformat speichern. Dabei werden sowohl die logischen als auch die visuellen Informationen des Statecharts gespeichert.
Die gespeicherten Modelle lassen sich im StateChart Designer wieder importieren. Dabei werden alle logischen und visuellen Informationen wieder hergestellt.

### Drucken von Modellen

Im Statechart Designer erstellte Modelle lassen sich über die Druckfunktion (Symbolleiste ) ausdrucken bzw. in einer Druckvorschau anzeigen. Dabei wird die in den Optionen festgelegte Art der Skalierung des Modells für den Druck verwendet.

### Generieren von Modelica-Code

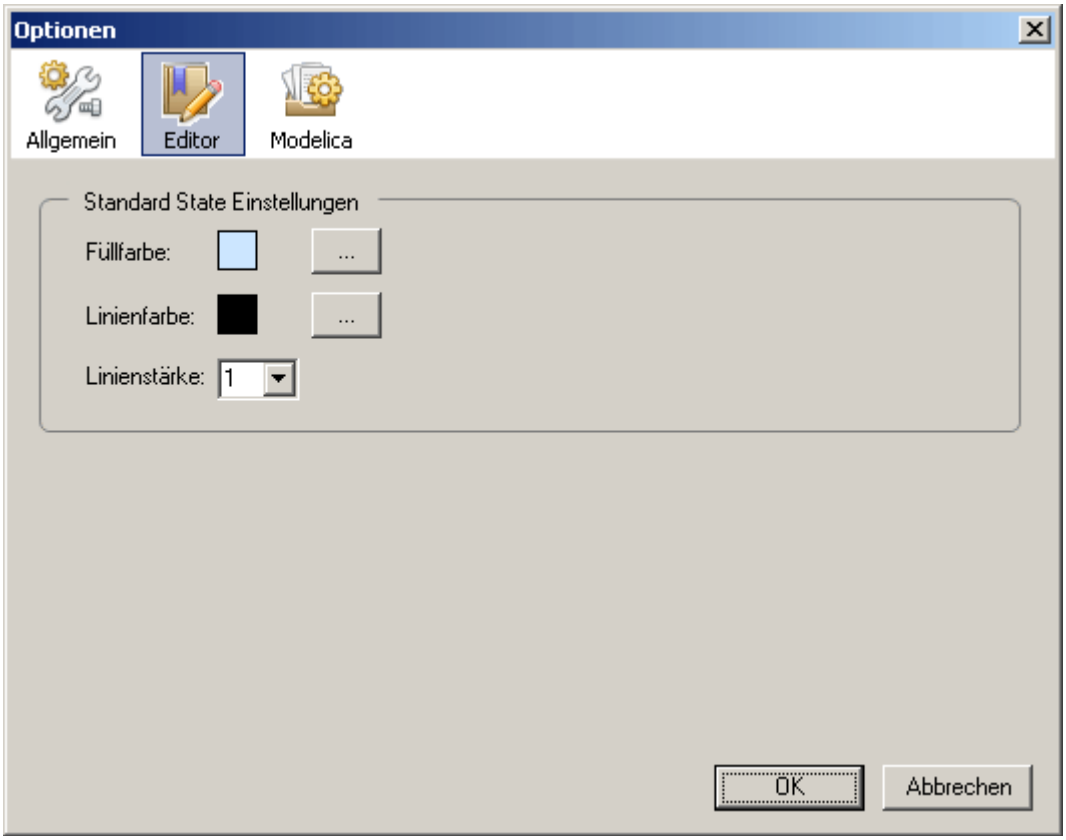
Beim Verlassen der Seite „Verhalten“ im TypeDesigner, z.B. durch das Betätigen des Buttons

, wird das automatisierte Generieren des Modelica-Quelltextes gestartet. Dabei werden die logischen Informationen des erstellten Modells ausgewertet und in Modelica Quelltext umgewandelt, der dann in den umgebenden Typen integriert wird. Zusätzlich werden die logischen und visuellen Informationen als Annotations im Typ gespeichert, so dass das Statechart beim erneuten Öffnen des SimulationX- Modells wieder angezeigt werden kann.

### Anpassen des Statechart Designers

Über den Optionsdialog (Symbolleiste ) lässt sich der StateChart Designer individuell anpassen. Die Optionen sind in die folgenden Kategorien unterteilt:

- Allgemein (Druckeinstellungen)
- Editor (Standard-Einstellungen für Zustände)
- Modelica (Einstellungen für Coverage)




## 8.4.3.3 Animierte Darstellung des Statechart in SimulationX

SimulationX bietet die Möglichkeit die Zustandsübergänge des Statecharts während der Simulation anzuzeigen. Die Visualisierung zur Simulationszeit wird im SimulationX- Modell wie eine Compound-Struktur angezeigt. Dazu klicken Sie auf das entsprechende Element im Modell und nutzen die Funktion - Open Compound.

Für die Visualisierung werden Modelica Grafikprimitive mit *DynamicSelect*-Anweisungen in der Diagrammebene verwendet. Die Visualisierung ist damit voll kompatibel zur Modelica-Sprachspezifikation.

Die Visualisierung zeigt folgende Informationen:

- ein vereinfachtes Zustandsdiagramm
- den gerade aktiven Zustand (Zustand ist rot)
- den letzten aktiven Zustand (Zustand mit roter Umrandung)

- die letzte geschaltete Transition (Transition ist rot)

Weiterhin ist es möglich, die Animation über die in SimulationX integrierte Aufnahmefunktion aufzunehmen und im Anschluss komfortabel wieder abzuspielen.

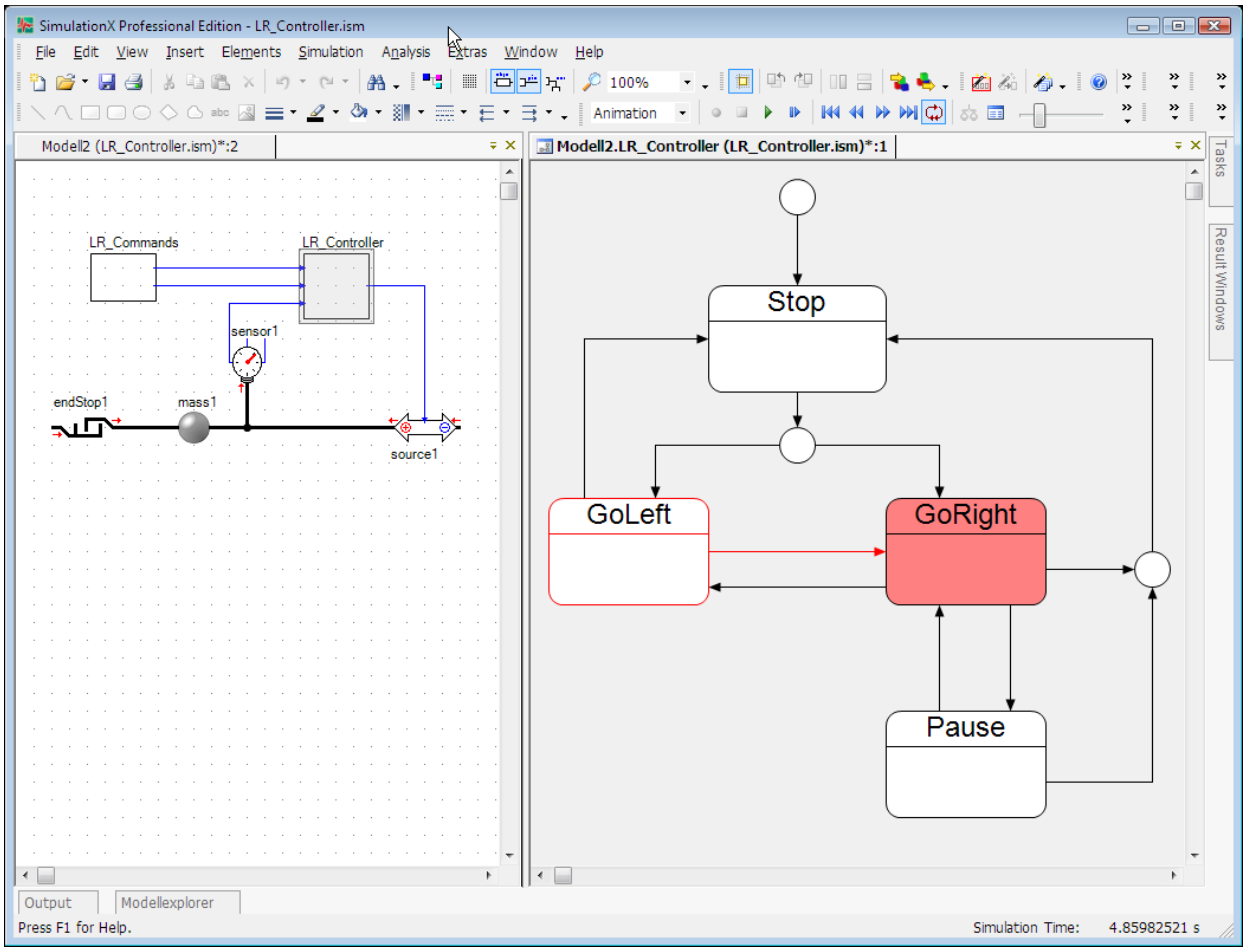



**Bild 8.49:** Animationsansicht des Statecharts

### 8.4.4 Semantik der Notationselemente in Modelica-Code

Die Semantik des ausgewählten UML-Statechart-Subsets wird am Beispiel eines Linearantriebs (**Bild 8.50**) mit Operator-Befehlen und Endanschlägen in Modelica-Code **[4]** erklärt.

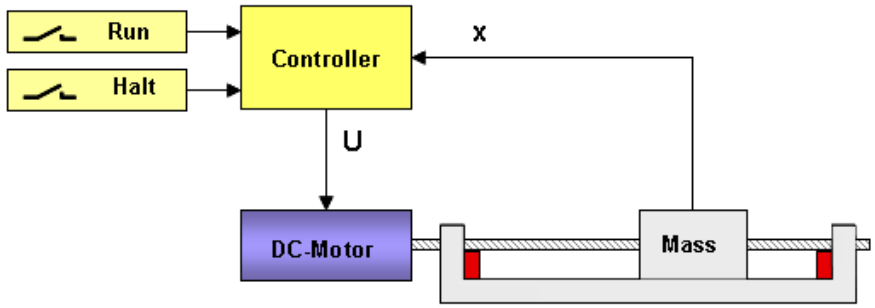



**Bild 8.50:** Komponenten des Linearantriebs

Das Steuerprogramm ist wie folgt spezifiziert:

- Starte nachdem der Run-Befehl gegeben wurde und fahre nach links
- Fahre 10 mal zwischen der linken und rechten Endposition hin und her
- Lege danach eine Pause von 3 s ein, setze anschließend die Bewegung fort
- Stoppe sofort, wenn ein Halt-Befehl gegeben wird
- Setze beim erneuten Run-Befehl mit der Aktion fort, die zum Zeitpunkt des Halt-Befehls ausgeführt wurde.

Die Operator-Befehle Run und Halt sowie die Position x der Masse sind die Inputs für den Controller. Der Controller liefert als Output die Motor-Versorgungs-spannung.

#### 8.4.4.1 Zustände

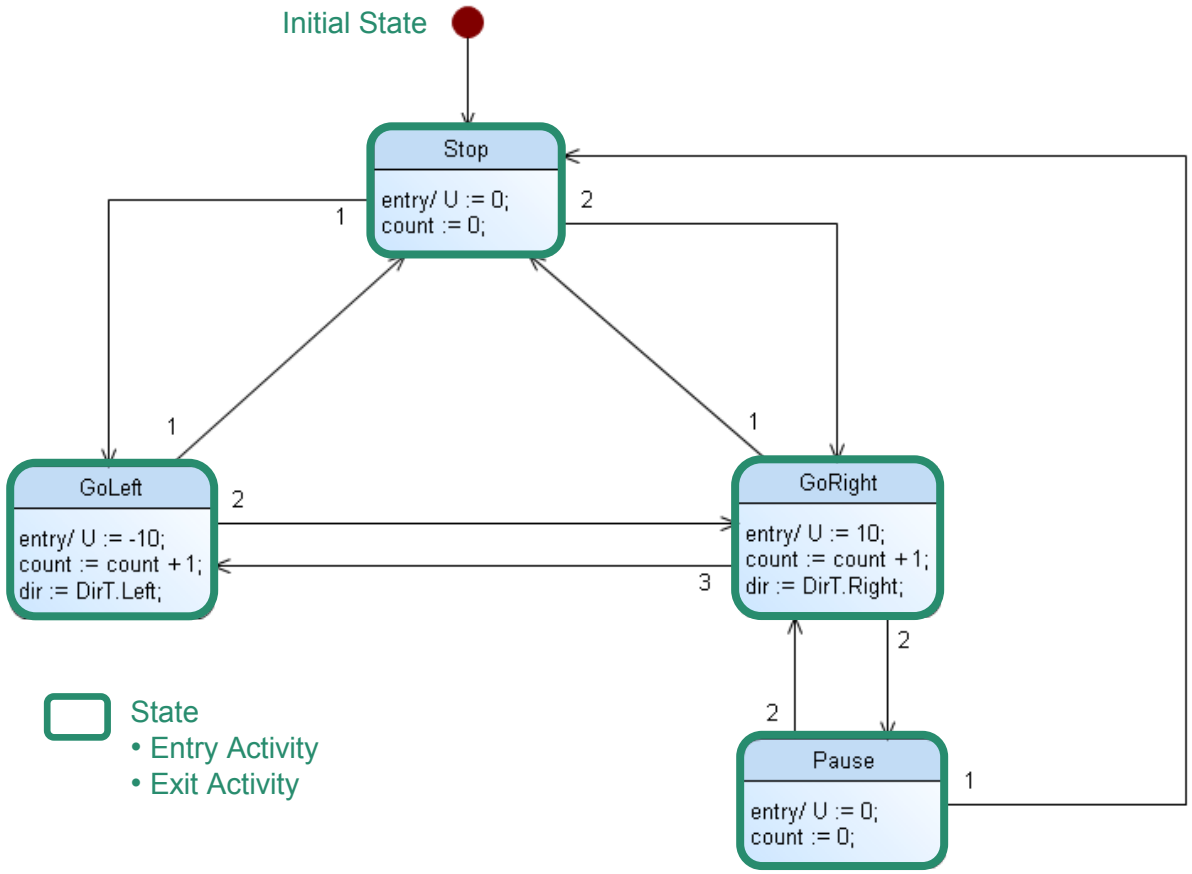
Der Zustandsautomat des Controllers wird durch einen zeitlosen Initial-Zustand und verschiedene zeitlich anhaltende Zustände beschrieben (**Bild 8.51**).

Die Namen der Zustände zeigen die geforderten Aktivitäten des Antriebs an: *Stop*, *GoLeft*, *GoRight* und *Pause*. Die Wertzuweisungen an die entsprechenden Steuersignale werden nach dem Betreten der Zustände als Entry-Activities ausgeführt. Im Allgemeinen sind diese Activities einfache Modelica-Algorithmen.

Die Modelica-Codierung schließt die Entry-Acivities in *when-clauses* ein, die den Eintritt in den jeweiligen Zustand detektieren. Nach dem Betreten des Zustandes *Stop* werden die Versorgungsspannung des Motors *U* und die Zählvariable *count* auf Null gesetzt. In *GoLeft* und *GoRight* wird *U* auf den für die Bewegung ver-einbarten Wert gesetzt und die Zählvariable wird inkrementiert. Beides geschieht einmalig, nachdem der jeweilige Zustand aktiviert wurde.
Außerdem wird in *GoLeft* und *GoRight* eine Enumeration-Variable *dir* gesetzt.

***dir*** speichert die aktuelle Bewegungsrichtung.

Im Gegensatz zu den Entry-Activities werden Exit-Activities ausgeführt, wenn die jeweiligen Zustände verlassen werden.

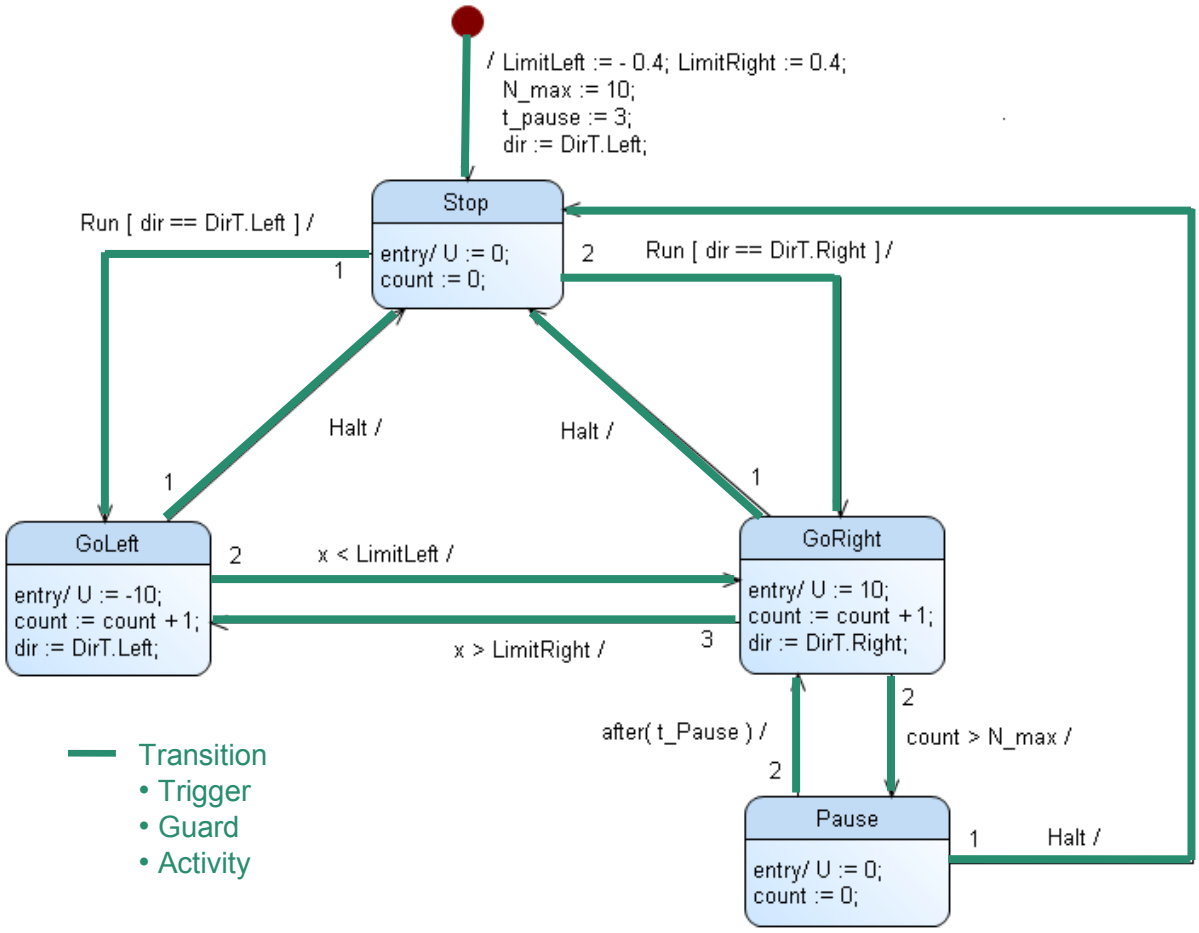



```
when ( state == StateT.Stop ) then
   U := 0;
   count := 0;
elsewhen ( state == StateT.GoLeft ) then
   // ...
elsewhen ( state == StateT.GoRight ) then
   // ...
elsewhen ( state == StateT.Pause ) then
   U := 0;
   count := 0;
   t_PauseFinished := time + t_Pause;
end when;
```

**Bild 8.51:** Zustände und Entry-Activities des Zustandsautomaten

### 8.4.4.2 Transitionen

Die Transitionen von Zustand zu Zustand werden optional mit einem Trigger, einem Guard und einer Aktivität bewertet (**Bild 8.52**). Eine Ausnahme bildet die Initial-Transition, die keinen Trigger oder Guard erlaubt. Die Modelica-Codierung fasst alle Übergänge in einer *if-clause* zusammen, in welcher der aktive Zustand detektiert und die Zustandsvariable auf den Folgezustand gesetzt wird. Zur Bestimmung des Folgezustandes werden die zur jeweiligen Transition gehörenden Trigger und Guard ausgewertet.

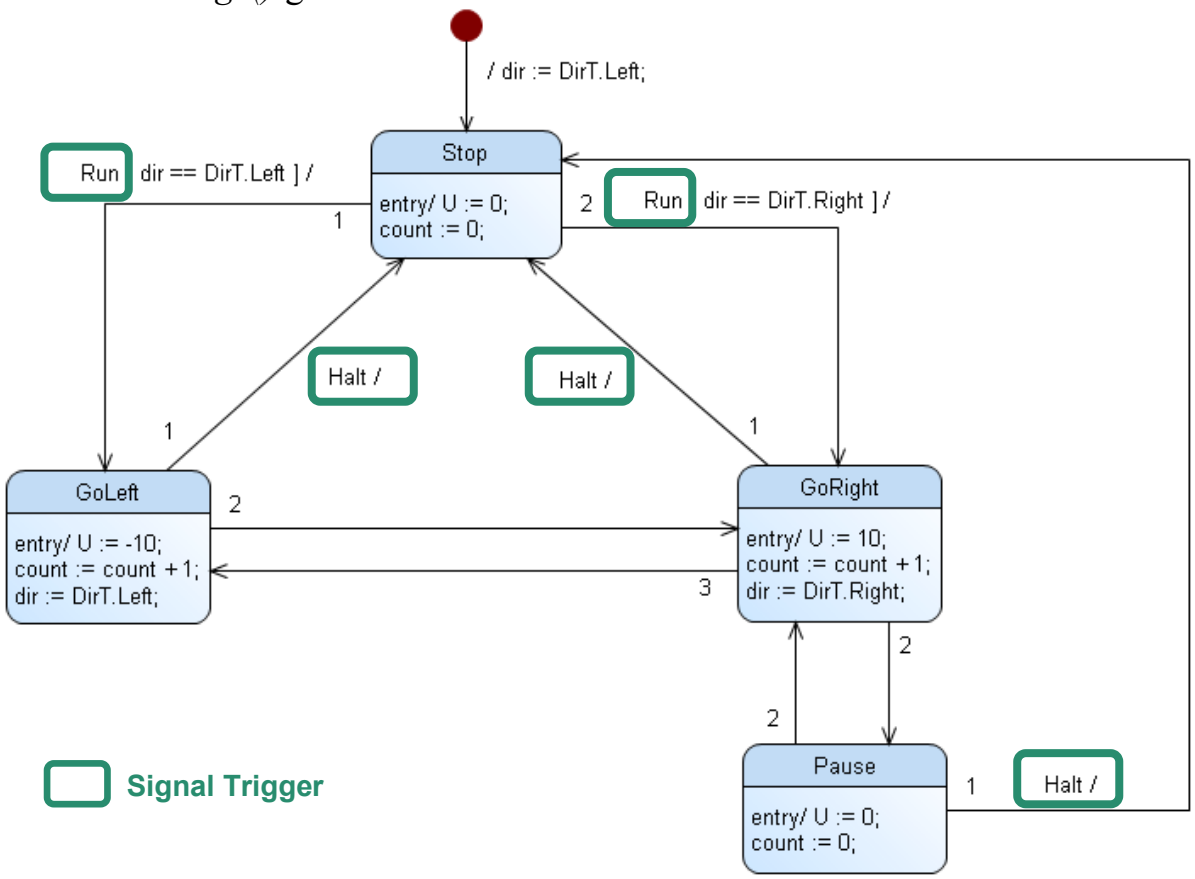



```
if ( pre( state ) == StateT.Stop ) then
   // ... state := ...;
elseif ( pre( state ) == StateT.GoLeft ) then
   // ... state := ...;
elseif ( pre( state ) == StateT.GoRight ) then
   // ... state := ...;
elseif ( pre( state ) == StateT.Pause ) then
   // ... state := ...;
end if;
```

**Bild 8.52:** Transitionen des Zustandsautomaten

### 8.4.4.3 Signal-Trigger

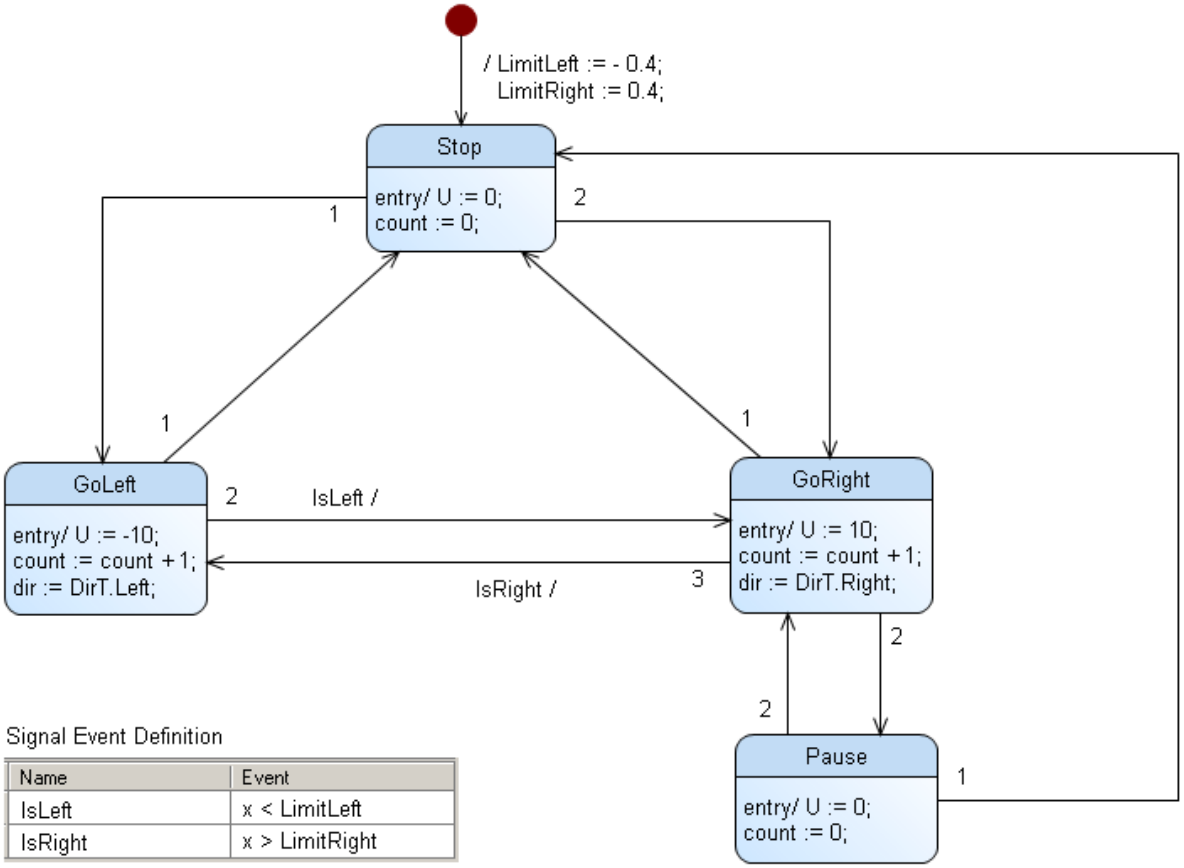
Signal-Trigger lösen Zustandsübergänge aus, wenn erwartete Nachrichten eintreffen. Diese Nachrichten können eine Menge von Argumenten umfassen oder sind einfache Befehle wie *Run* und *Halt* (**Bild 8.53**). Der Empfang einer Nachricht wird durch das Umschalten (Toggeln) einer Boole'schen Variablen signalisiert. Handelt es sich um einen Befehl, sind Befehl und Toggel-Variable identisch. Um den Empfang der Nachricht bzw. Befehls zu erkennen, wird die Modelica-Funktion *change()* genutzt.




```
if ( pre( state ) == StateT.Stop ) then
   if ( change( Run ) ) then
      // ...
   end if;
elseif ( pre( state ) == StateT.GoLeft ) then
   if ( change( Halt ) ) then
      // ...
   end if;
// ...
end if;
```

**Bild 8.53:** Signal-Trigger des Zustandsautomaten

Die Nachrichten oder Befehle können Inputs der Zustandsmaschine sein oder innerhalb der Zustandsmaschine erzeugt werden. Im zweiten Fall (**Bild 8.54**) werden beispielsweise die Boole'schen Variablen *IsLeft* und *IsRight* vereinbart. Die Position *x* der Masse wird hinsichtlich des Überschreitens der Grenzwerte *LimitLeft* und *LimitRight* überwacht. Jedes Überschreiten dieser Grenzwerte wird durch das Toggeln von *IsLeft* und *IsRight* signalisiert.

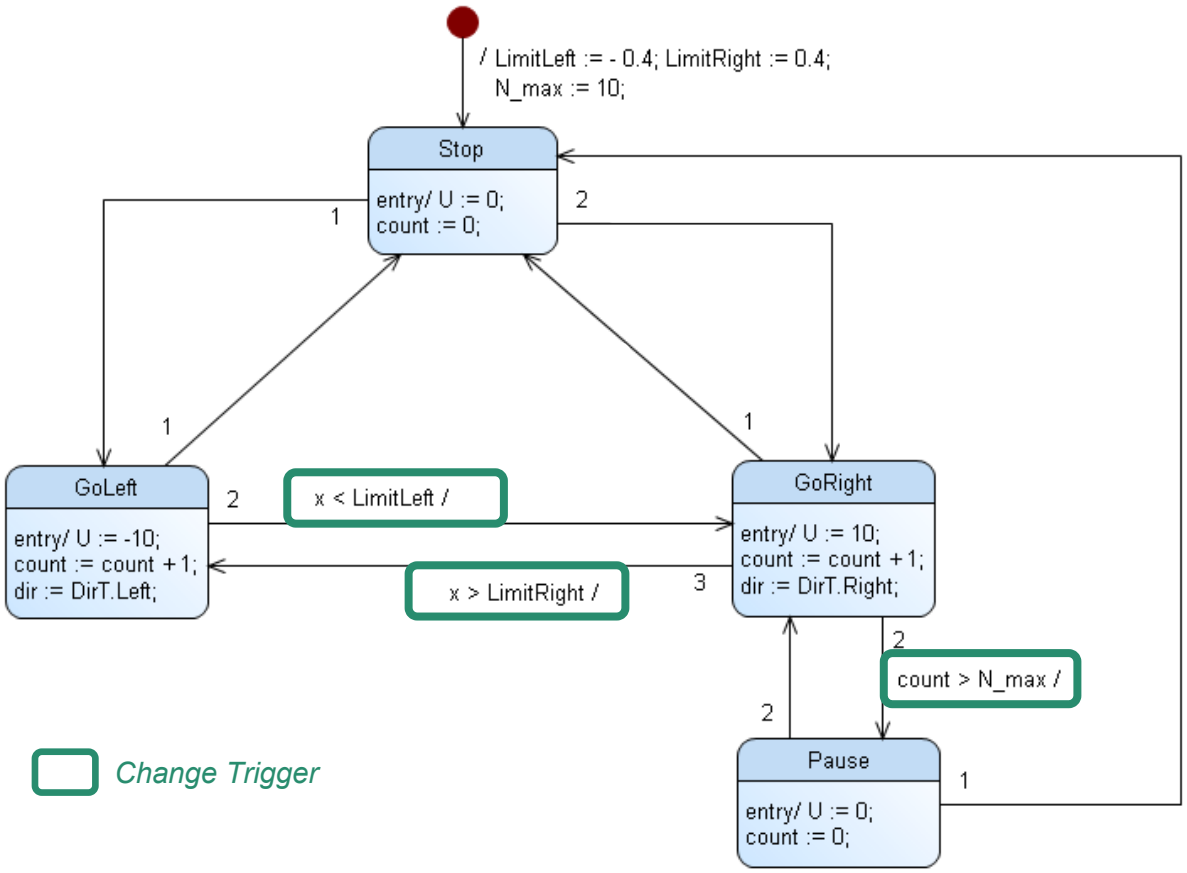



```
Boolean  IsLeft, IsRight;
Real     x, LimitLeft:=-0.4, LimitRight:=0.4;

when ( x < LimitLeft ) then
   IsLeft := not IsLeft;
end when;

when ( x > LimitRight ) then
   IsRight := not IsRight;
end when;
```

**Bild 8.54:** Generierung von Signal-Ereignissen

### 8.4.4.4 Change-Trigger

Change-Trigger lösen Zustandsübergänge aus, wenn Systemvariable vorgegebene Bedingungen erfüllen. Diese Bedingungen werden als Boole'sche Ausdrücke notiert. Der Change-Trigger wird auf "On" geschaltet, wenn der Boole'sche Ausdruck das erste Mal mit TRUE evaluiert wird. Er bleibt "On", so lange wie die Auswertung des Boole'schen Ausdrucks TRUE liefert. Im Fall des Linearantriebs sind die Überschreitungen des linken und rechten Grenzwertes als Change-Trigger notiert (**Bild 8.55**). Die Modelica-Repräsentation eines Change-Triggers ist eine *if-clause* mit dem im Zustandsdiagramm angegebenen Boole'schen Ausdruck.

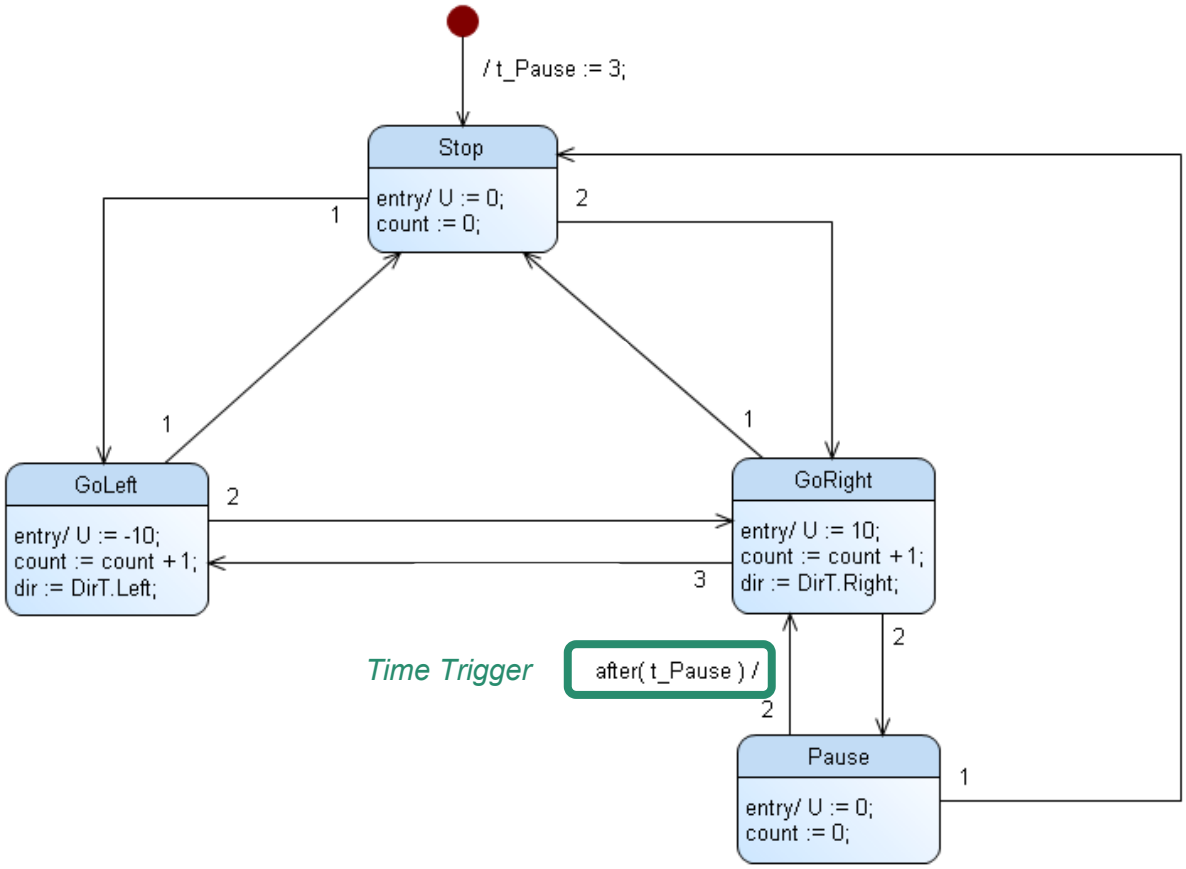



```
if ( pre( state ) == StateT.Stop ) then
     // ...
elseif ( pre( state ) == StateT.GoLeft ) then
   if ( change( Halt ) ) then
      state := StateT.Stop;
   elseif ( x < LimitLeft ) then
      state := StateT.GoRight;
   end if;
// ...
end if;
```

**Bild 8.55:** Change-Trigger des Zustandsautomaten

### 8.4.4.5 Time-Trigger

Time-Trigger lösen Zustandsübergänge aus, wenn vorgegebene Zeiten abgelaufen sind (**Bild 8.56**). Die in *after-clauses* notierten Zeitvorgaben sind relativ zum Zeitpunkt der Aktivierung des Quellzustandes der Transition.




```
when ( state == StateT.Stop ) then
   // ...
elsewhen ( state == StateT.Pause ) then
   U := 0; count := 0;
   t_PauseFinished := time + t_Pause;  // assign time limit
end when;
```

```
when ( time >= t_PauseFinished ) then
   timeout := not timeout;          // toggle timeout signal
end when;
```

```
if ( state == StateT.Stop ) then
   //...
elseif ( state == StateT.Pause ) then
   if ( change( Halt ) ) then
      // ...
   elseif ( change( timeout ) ) then // evaluate timeout signal
      state := StateT.GoRight;
   end if;
end if;
```

**Bild 8.56:** Time-Trigger des Zustandsautomaten

In der Modelica-Repräsentation (**Bild 8.56**) werden die Time-Trigger wie folgt organisiert:

(1) Eine erste *when-clause* detektiert die Aktivierung des Quellzustandes der zeitbewerteten Transition und weist einer zugeordneten Variablen das Zeitlimit zu.

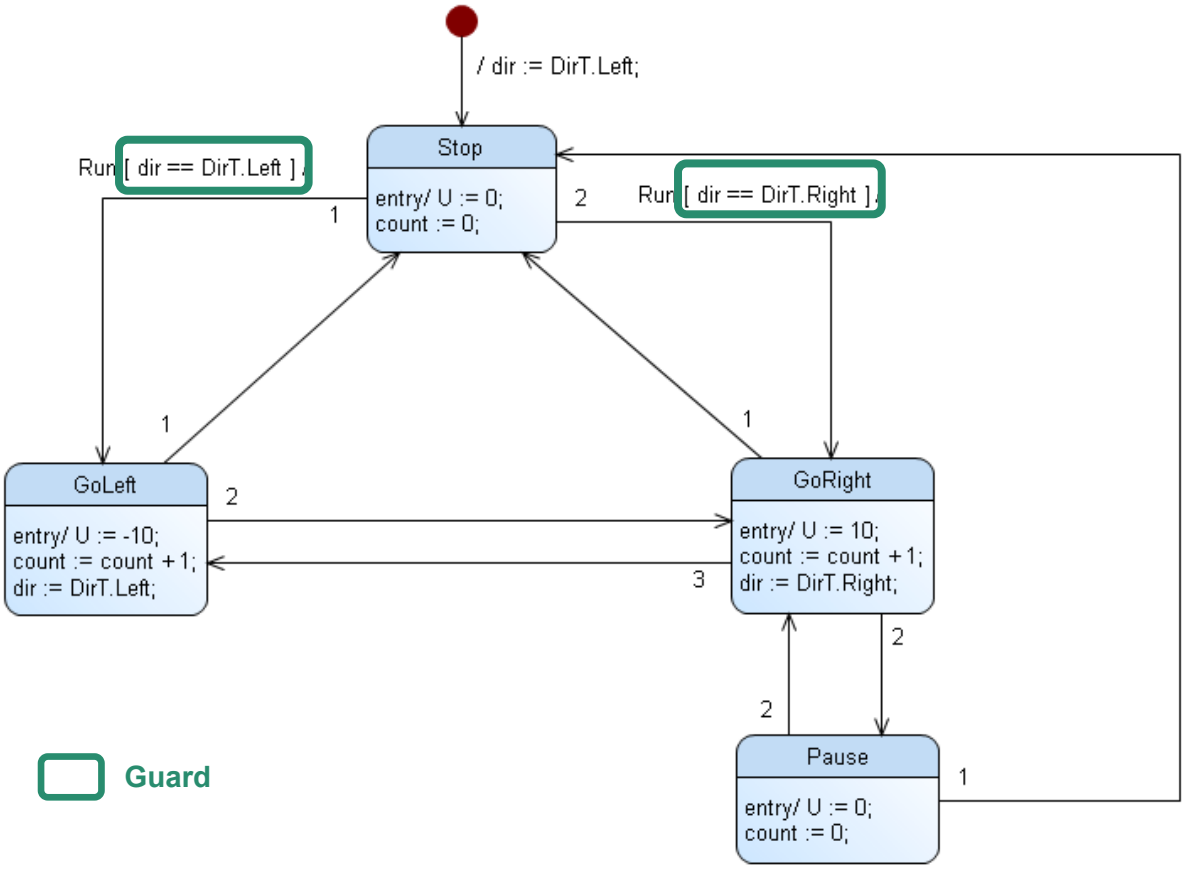
(2) Eine zweite *when-clause* prüft, ob die Simulationszeit das Zeitlimit überschreitet. Wenn ja, wird ein Timeout-Signal getoggelt.

(3) Eine *if-change-clause* erkennt das Toggeln des Timeout-Signals und setzt die Zustandsvariable auf den Zielzustand der zeitbewerteten Transition.

#### 8.4.4.6 Guards

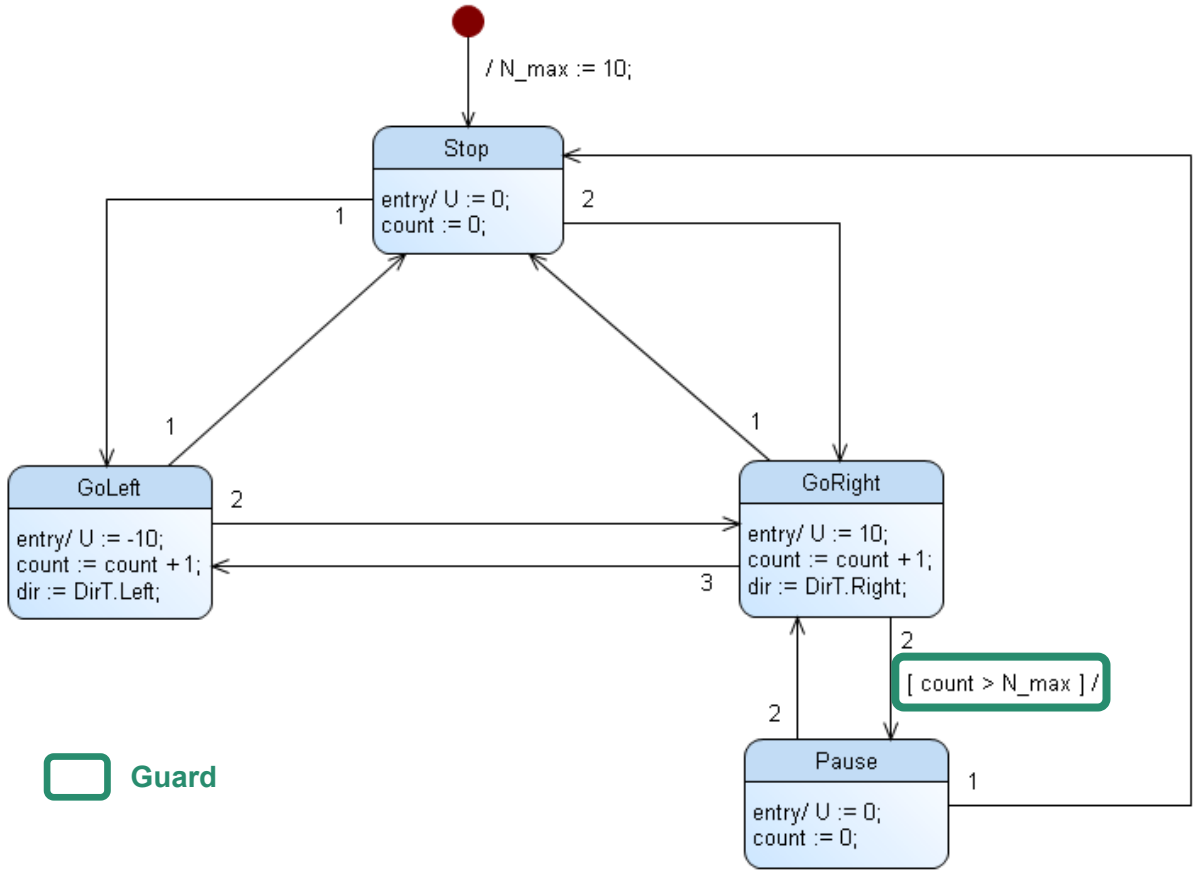
Guards sind zusätzliche Boole'sche Bedingungen für das Feuern einer Transition. Sie werden ausgewertet, wenn die assoziierten Trigger auslösen. Im Fall des Linearantriebes werden Guards genutzt, um den *Run*-Befehl zu differenzieren (**Bild 8.57**). Gemäß Anforderung ist nach einem Restart die zuvor unterbrochene Aktivität fortzusetzen. Dazu wird eine Variable *dir* eingeführt, die bei der Aktivierung der jeweiligen Bewegung gesetzt und in den Guards ausgewertet wird. Im Modelica-Code werden Trigger und Guard UND-verknüpft.

Die Transitionsbedingungen von *(GoRight, Pause)* kann als Change-Trigger (**Bild 8.55**) oder Guard (**Bild 8.58**) formuliert werden.




```
if ( pre( state ) == StateT.Stop ) then
   if ( change( Run ) and ( dir == DirT.Left ) ) then
      state := StateT.GoLeft;
   elseif ( change( Run ) and ( dir == DirT.Right ) ) then
      state := StateT.GoRight;
   end if;
elseif ( pre( state ) == StateT.GoLeft ) then
// ...
end if;
```

**Bild 8.57:** Guards als zusätzliche Schaltbedingungen




```
Boolean Complete;
Integer N_Max;

when( ) then
   // ...
elsewhen ( state == StateT.GoRight ) then
   U:= 10; count := count + 1; dir := DirT.Right;
   Complete := not Complete;
end when;
```

```
if ( pre( state ) == StateT.Stop ) then
  // ...
elseif ( pre( state ) == StateT.GoRight ) then
   if ( change( Halt ) ) then
      state := StateT.Stop;
   elseif ( change( Complete ) and ( count > N_Max ) ) then
      state := StateT.Pause;
   elseif ( x > LimitRight ) then
      state := StateT.GoLeft;
   end if;
// ...
end if;
```

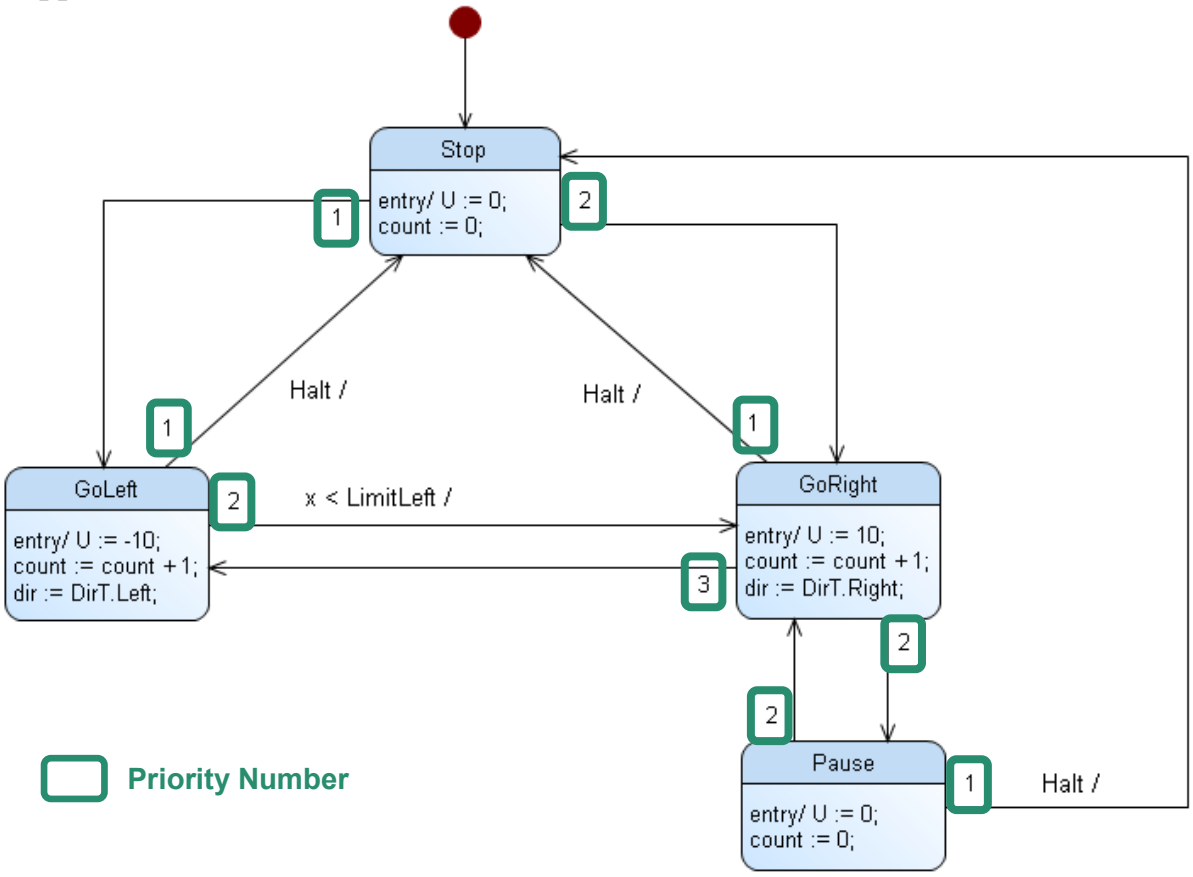
**Bild 8.58:** Verknüpfung von Guard und Completion-Trigger

Wird im Zustandsautomaten der Guard allein als Schaltbedingung aufgeführt, so wird dieser in der Modelica-Repräsentation mit einem Completion-Trigger

verknüpft. Eine *when-clause* detektiert den Eintritt in den Quellzustand der Transition *(GoRight, Pause)* und löst als letzte Operation das Completion-Ereignis aus, indem die Boole'sche Variable *Complete* getoggelt wird. Die *if-clause* für den Zustandsübergang enthält die UND-Verknüpfung des Completion-Triggers *change (Complete )* und des Guard.

### 8.4.4.7 Prioritäten

Die an den Transitionen notierten Prioritätsnummern bestimmt die Reihenfolge beim Test der Transitionen auf Feuerbarkeit. Im Fall des Linearantriebes erhält der *Halt*-Befehl die höchste Priorität, um den Antrieb in jeder Situation sicher zu stoppen.

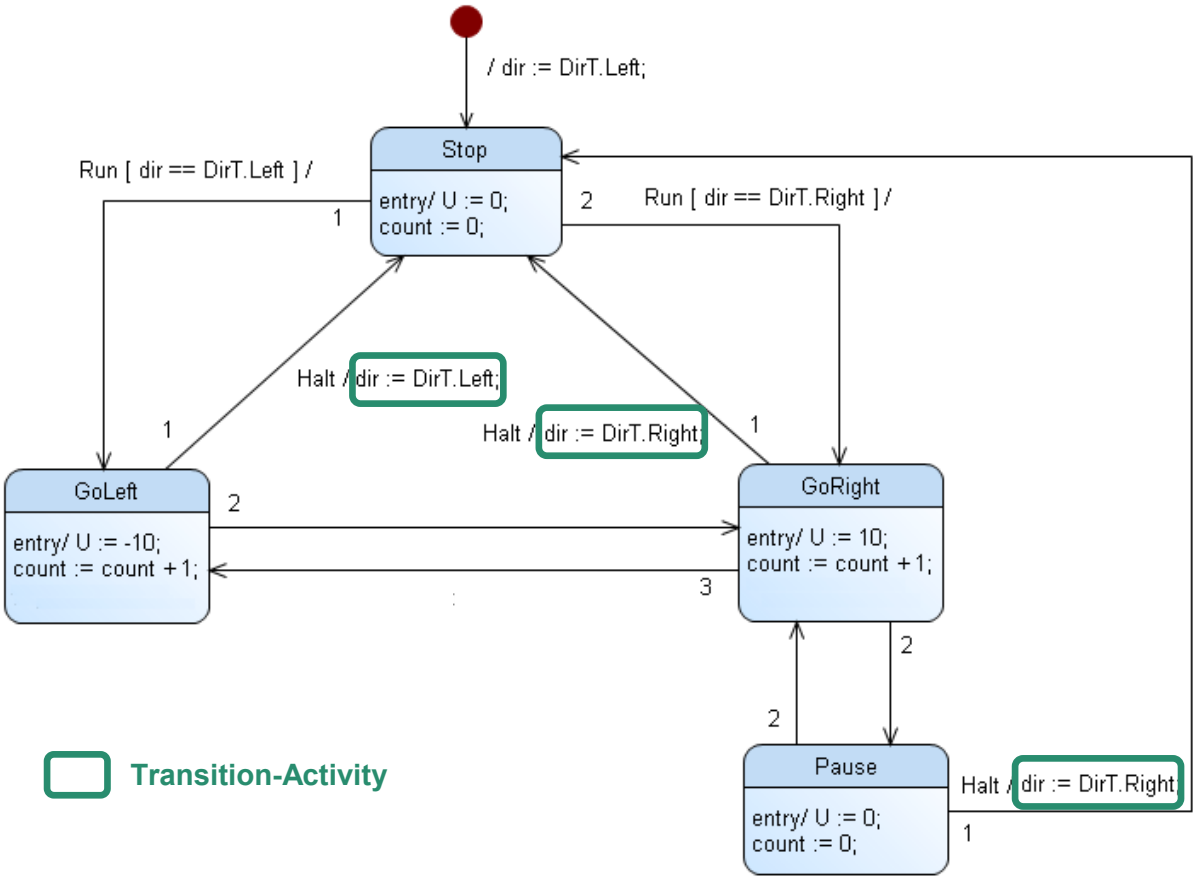



```
if ( pre( state ) == StateT.Stop ) then
     // ...
elseif ( pre( state ) == StateT.GoLeft ) then
   if ( change( Halt ) ) then       // Priority 1
      state := StateT.Stop;
   elseif ( x < LimitLeft ) then    // Priority 2
      state := StateT.GoRight;
   end if;
// ...
end if;
```

**Bild 8.59:** Prioriäten beim Test der Transitionsbedingungen

### 8.4.4.8 Transition-Activities

Die Speicherung der Bewegungsrichtung kann alternativ zu Entry-Activities in den Transition-Activities erfolgen, indem der Variablen *dir* die Enumeration der gestoppten Bewegung *GoLeft* oder *GoRight* zugewiesen wird. Diese Wertzuweisungen, die hier für einfache Modelica-Algorithmen stehen, sind in die *if-clause* zur Bestimmung der Zuständsübergänge aufgenommen.




```
if ( pre( state ) == StateT.Stop ) then
     // ...
elseif ( pre( state ) == StateT.GoLeft ) then
   if ( change( Halt ) ) then
      state := StateT.Stop;
      dir := DirT.Left;          // Transition-Activity
   elseif ( x < LimitLeft ) then
      state := StateT.GoRight;
   end if;
// ...
end if;
```

**Bild 8.60:** Transitionen mit Aktivitäten

### 8.4.4.9 Zusammengesetzte Zustände

Zusammengesetzte Zustände werden in Verbindung mit High-Level-Transitionen, Initial States, History States und Junctions für den Entwurf von hierarchischen Zustandsautomaten genutzt.

Die Modelica-Repräsentation des Zustandsautomaten definiert dazu:

- Individuelle Zustandsvariablen für das *Main* (oberste Hierarchieebene) und jeden Composite State
- Individuelle Entry-Activity-Blöcke für das *Main* und jeden Composite State
- Individuelle Transition-Blöcke für das *Main* und jeden Composite State
- *entry*-Variable für das Ziel beim Eintritt in den Composite State
- *onto*-Variable für das Ziel beim Verlassen des Composite State
- *intial*-Variable zur Festlegung des zu aktivierenden Zustands beim Default Entry
- *history*-Variable zur Speicherung des letzten aktiven Zustands für den History Entry
- Completion-Trigger zur Kennzeichnung der Deaktivierung des Composite State.

Im hierarchischen Zustandsdiagramm für den Linearantrieb (**Bild 8.61**) wird beim Übergang *(Stop, Go)* eine Transition verwendet, die über eine Junction geteilt wird und in *Go* explizit den Subzustand *GoLeft* oder *GoRight* akiviert.

Im Modelica-Block *Transitions of Main* (**Bild 8.66**) wird die Auflösung dieser Transition in einer *if-elseif-clause* definiert, wobei die *entry*-Variable *entryGoState* das Ziel *GoLeft* oder *GoRight* zugewiesen bekommt. Der Zielzustand auf *Main*-Niveau ist *Go*.

Die High-Level-Transition *(Go, Stop)* wird aus Sicherheitsgründen mit der Prioritätsnummer Null bewertet und damit zur Highest-Priority-Transition für den Composite State *Go* erhoben.

Im Modelica-Block *Transitions of Go* (**Bild 8.66**) wird demzufolge die Transitionsbedingung von *(Go, Stop)*, das Signal-Ereignis *Halt*, in jedem Sub-zustand als erste geprüft. Bei positivem Ergebnis werden die *onto*-Variable *ontoState* auf den Zielzustand *Stop* gesetzt und der Zustandsvariablen von *Go* der Zustand *InActive* zugewiesen

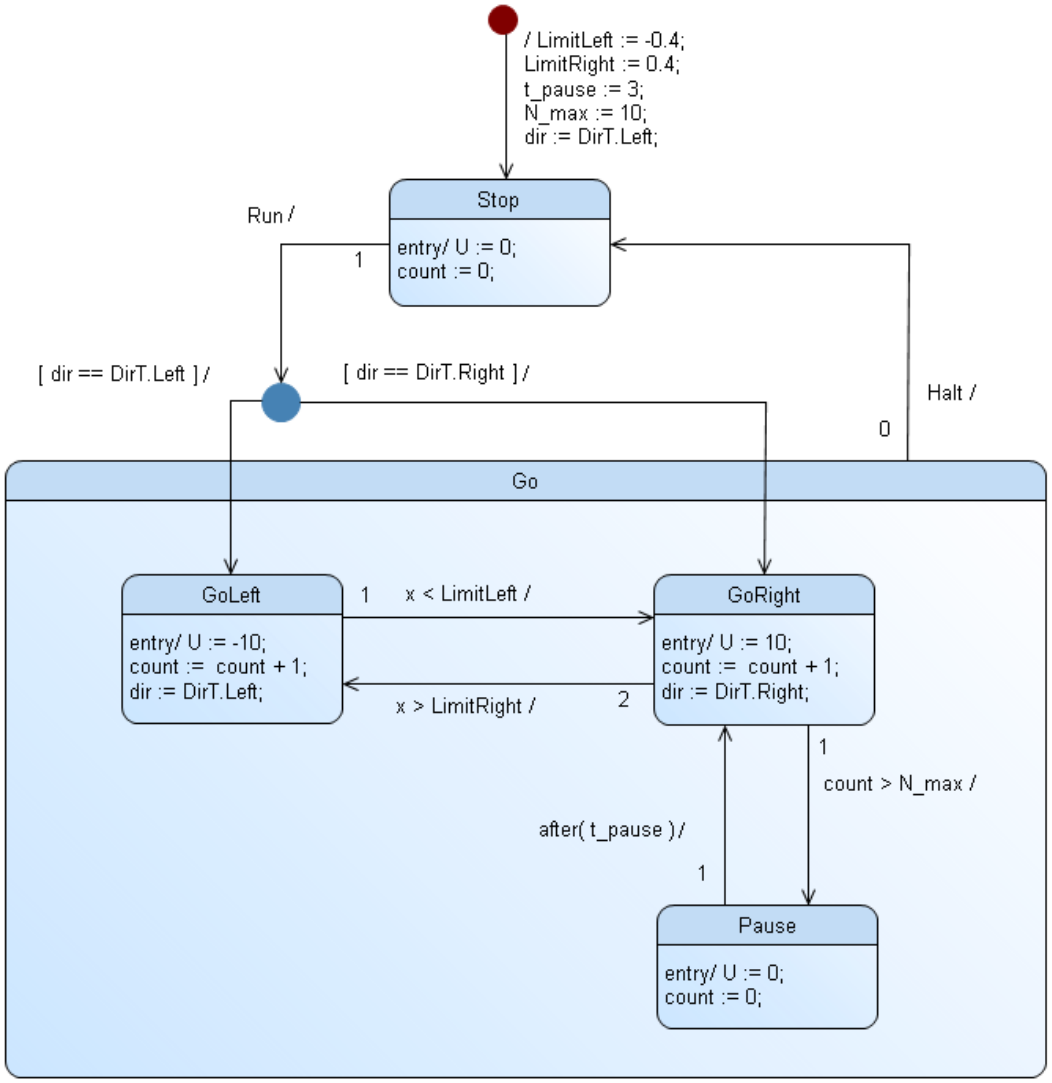



**Bild 8.61:** Hierarchisches Zustandsdiagramm mit Explicit Entry

Im Entry-Activity-Block von *Go* (**Bild 8.65**) wird der Zustand *InActive* detektiert und daraufhin die Boole'sche Variable *completeGo* getoggelt. Dieses Signal-Ereignis wird im Modelica-Block *Transitions of Main* erkannt und löst via Variable *ontoState* den Übergang auf den Zielzustand aus.

Der oben erläuterte Mechanismus wiederholt sich für jede Hierarchie-Stufe, d.h. für jeden weiteren Composite State innerhalb eines Composite State. Varianten entstehen durch die unterschiedlichen Eintrittsmöglichkeiten: Explicite Entry (**Bild 8.61**), History Entry (**Bild 8.67**) und Default Entry (**Bild 8.70**).

```
model Controller

input SignalBlocks.InputPin Run[*];
input SignalBlocks.InputPin Halt[*];
input SignalBlocks.InputPin x[*];
output SignalBlocks.OutputPin U[*];

type StateT   = enumeration( Stop, Go );
type GoStateT = enumeration( InActive, GoLeft, GoRight, Pause
);
type DirT     = enumeration( Left, Right );

StateT   state, ontoState;
GoStateT goState, entryGoState, historyGoState, initialGoState;
Boolean  completeGo;
Boolean  timeout;
Real     t_pause_finished;
```

**Bild 8.62:** Modelica-Codierung der Antriebssteuerung: Deklarationen

```
// Initialization
when initial() then
  state            := StateT.Stop;
  ontoState        := StateT.Stop;
  goState          := GoStateT.InActive;
  entryGoState     := GoStateT.InActive;
  initialGoState   := GoStateT.InActive;
  historyGoState   := GoStateT.InActive;
  completeGo       := false;
  timeout          := false;
  t_pause_finished := 0;
  Limit_Left       := -0.4;
  Limit_Right      := 0.4;
   t_pause          := 3;
   N_max            := 10;
   dir              := DirT.Left;
end;
```

**Bild 8.63:** Modelica-Codierung der Antriebssteuerung: Initialisierung

```
// Event Generation
when ( time >= t_pause_finished ) then
  timeout := not timeout;
end when;
```

**Bild 8.64:** Modelica-Codierung der Antriebssteuerung: Ereignisgenerierung

```
// Entry Activities of Main
when ( state == StateT.Stop ) then
  U := 0;
  count := 0;
elsewhen ( state == StateT.Go ) then
  goState := entryGoState;
end when;

// Entry Activities of Go
when ( goState == GoStateT.GoLeft ) then
  U := -10;
  count := count + 1;
  dir := DirT.Left;
elsewhen ( goState == GoStateT.GoRight ) then
  U := +10;
  count := count + 1;
  dir := DirT.Right;
elsewhen ( goState == GoStateT.Pause ) then
  U := 0;
  count := 0;
  t_pause_finished := time + t_pause;
elsewhen ( goState == GoStateT.InActive ) then
  completeGo := not completeGo;
end when;
```

**Bild 8.65:** Modelica-Codierung der Antriebssteuerung: Entry-Activities

```
// Transitions of Main
if ( pre( state ) == StateT.Stop ) then
  if ( change( Run ) and ( dir == DirT.Left ) ) then
     state := StateT.Go;
      entryGoState := GoStateT.GoLeft;
   elseif ( change( Run ) and ( dir == DirT.Right ) ) then
     state := StateT.Go;
      entryGoState := GoStateT.GoRight;
  end if;
elseif ( pre( state ) == StateT.Go ) then
  if ( change( completeGo ) ) then
     state := ontoState;
  end if;
end if;

// Transitions of Go
if ( pre( goState ) == GoStateT.GoLeft ) then
  if ( change( Halt ) ) then
     ontoState := StateT.Stop;
     goState   := GoStateT.InActive;
  elseif ( x < LimitLeft ) then
      goState := GoStateT.GoRight;
  end if;
```

```
elseif ( pre( goState ) == GoStateT.GoRight ) then
  if ( change( Halt ) ) then
     ontoState := StateT.Stop;
     goState   := GoStateT.InActive;
  elseif ( count > N_Max ) then
     goState := GoStateT.Pause;
  elseif ( x > LimitRight ) then
      goState := GoStateT.GoLeft;
   end if;
elseif ( pre( goState ) == GoStateT.Pause ) then
   elseif ( change( Halt ) ) then
      t_pause_finished := time;
      ontoState := StateT.Stop;
      goState   := GoStateT.InActive;
   elseif ( change( timeout ) ) then
      goState := GoStateT.GoRight;
   end if;
end if;

// Annotations
   // ...

end Controller;
```

**Bild 8.66:** Modelica-Codierung der Antriebssteuerung: Transitionen

Der Wiedereintritt in *Go* über einen History State (**Bild 8.67**) wird mit einer *history*-Variablen *historyGoState organisiert*, die die Identifikation des zuletzt aktiven Subzustands speichert. Die Zuweisung erfolgt in den Entry-Activities der Subzustände von *Go* (**Bild 8.68**).

Zur Aktivierung von *Go* wird in den *Transitions of Main* der Wert der *history*-Variablen *historyGoState* auf die *entry*-Variable von *Go* übertragen (**Bild 8.69**).

Für die erstmalige Aktivierung von *Go* wird *historyGoState* beim Start der Zustandsmaschine mit *GoLeft* intialisiert.

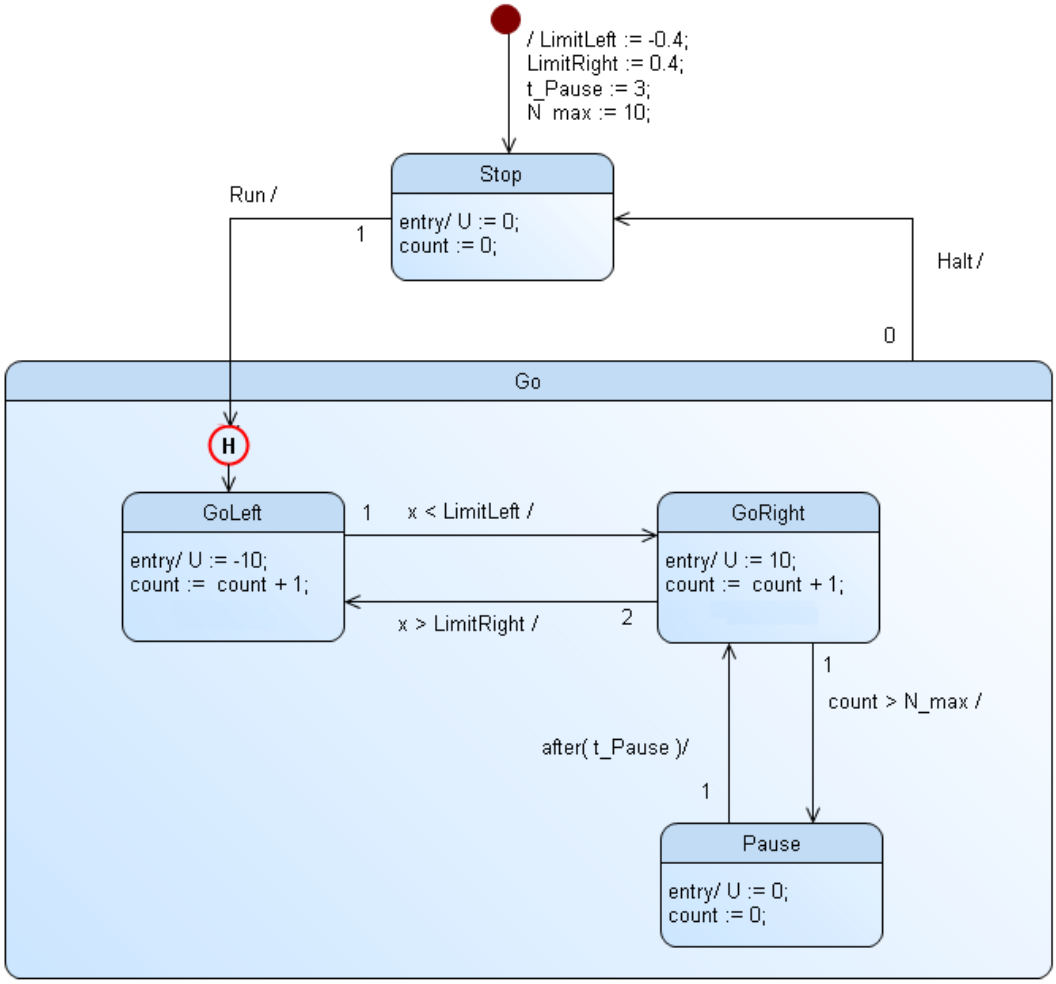



**Bild 8.67:** Hierarchisches Zustandsdiagramm mit History Entry

```
// Entry Activities of Main
when ( state == StateT.Stop ) then
   U := 0;
   count := 0;
elsewhen ( state == StateT.Go ) then
   goState := entryGoState;
end when;

// Entry Activities of Go
when ( goState == GoStateT.GoLeft ) then
   U := -10;
   count := count + 1;
   historyGoState := GoStateT.GoLeft;

elsewhen ( goState == GoStateT.GoRight ) then
   U := +10;
   count := count + 1;
   historyGoState := GoStateT.GoRight;
```

```
elsewhen ( goState == GoStateT.Pause ) then
   U := 0;
   count := 0;
   t_pause_finished := time + t_pause;
   historyGoState := GoStateT.Pause;

elsewhen ( goState == GoStateT.InActive ) then
   completeGo := not completeGo;
end when;
```

**Bild 8.68:** Modelica-Codierung der Historie: Setzen der *history*-Variablen von *Go*

```
// Transitions of Main
if ( pre( state ) == StateT.Stop ) then
   if ( change( Run ) ) then
      state := StateT.Go;
      entryGoState := historyGostate;
   end if;
elseif ( pre( state ) == StateT.Go ) then
   if ( change( completeGo ) ) then
      state := ontoState;
   end if;
end if;
...
// Transitions of Go
if ( pre( goState ) == GoStateT.GoLeft ) then
   // wie in Bild 3.12-5
end if;
```

**Bild 8.69:** Modelica-Codierung der Historie: Setzen der *entry*-Variablen von *Go*

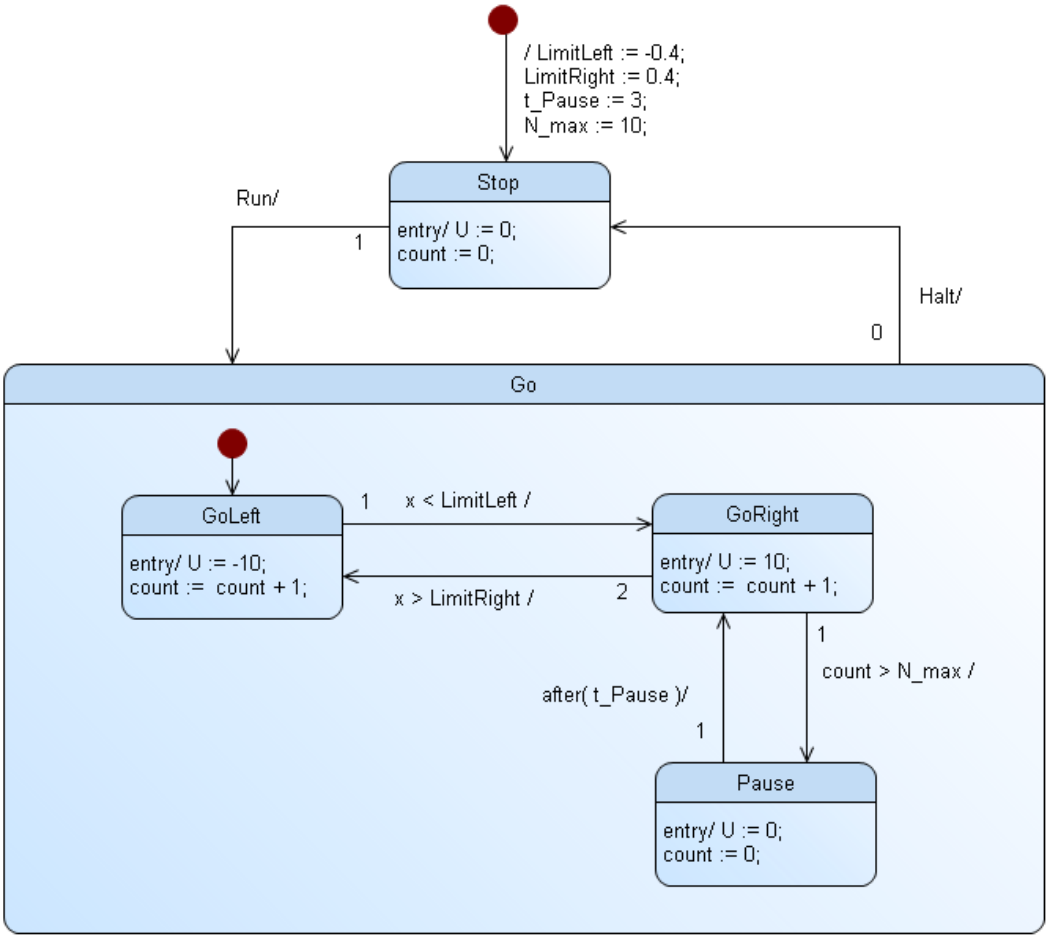
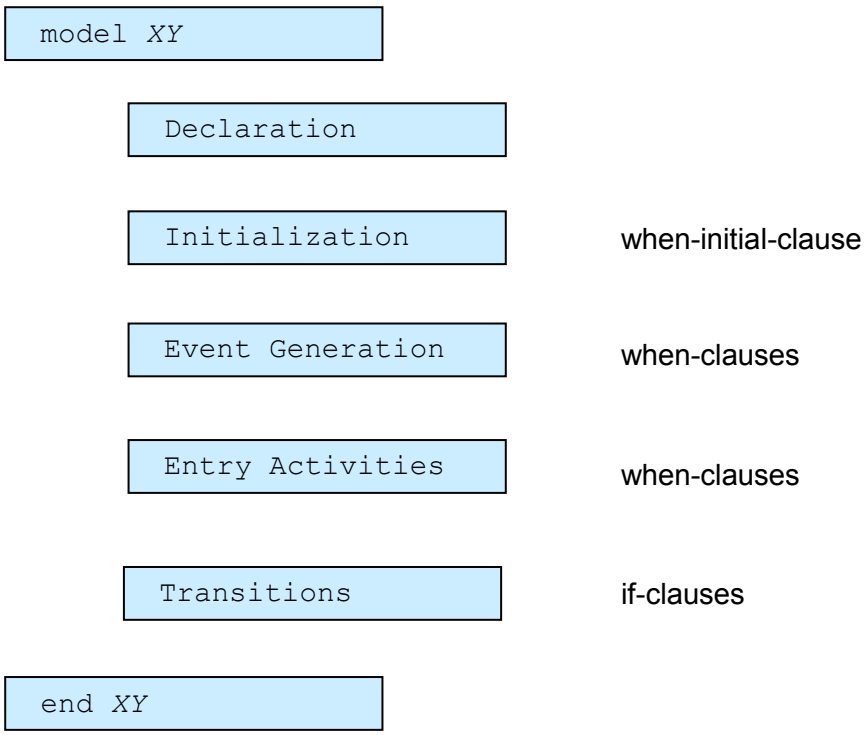



**Bild 8.70:** Hierarchisches Zustandsdiagramm mit Default Entry

Der Default Entry wird mit einem Initial Pseudostate definiert, dessen auslaufende Transition auf den Zustand zeigt, der beim Eintritt in den Composite State zu aktivieren ist. Dieses Ziel, hier *GoLeft*, wird einer *initial*-Variablen beim Start der Zustandsmaschine zugewiesen. Zur Aktivierung von *Go* wird in den *Transitions of Main* der Wert der *initial*-Variablen auf die *entry*-Variable von *Go* übertragen (analog zur History-Organisation nach **Bild 8.69**). Mit dem hier gezeigten Default Entry beginnt der Linearantrieb die Bewegung in die linke Richtung und setzt diese nach *Halt* stets in die linke Richtung fort.

### 8.4.4.10 Gesamtstruktur der Modelica-Codierung

Die in **Abschnitt 8.4.4** gezeigten Modelica-Code-Segmente werden wie folgt zur Codierung von Zustandsmaschinen zusammengefasst:




Es umfassen:

| | |
|---|---|
| Declaration | Typvereinbarungen,<br>EA-Signale, interne Signale,<br>Zustandsvariablen,<br>inital-, entry-, onto- und history-Variablen von Hierarchien,<br>Zeitvariablen,<br>Nutzervariablen,<br>Parameter |
| Initialization | Anfangswertzuweisungen an Variablen,<br>Anweisungen der Initial-Transitionen |
| Event Generation | Generierung von Signal-Ereignissen entsprechend Signal Event Definition Table,<br>Generierung von Timeout-Ereignissen |
| Entry Activities | Anweisungen entsprechend der Entry-Activities der Zustände,<br>Anweisung der Grenzwerte an Zeitvariablen und Generierung von Completion-Ereignissen für jede Hierarchieebene |
| Transitions | Anweisung des Folgezustands,<br>Anweisungen entsprechend der Exit-Activities der Zustände,<br>Anweisungen entsprechend der Transition-Activities und Rücksetzen von Zeitvariablen für jede Hierarchieebene. |

SIMULATION X®

Powered by ITI

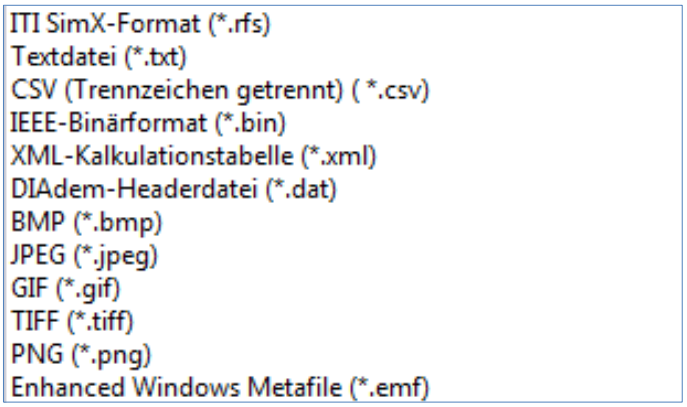
Interfaces

# Interfaces

# 9 Schnittstellen

## 9.1 Datenformate

Zum *Austausch von Daten mittels Datenträger* mit anderen Windows-Anwendungen wird in SimulationX eine Reihe von Formaten bereitgestellt.




Durch die Bereitstellung von verschiedensten Optionen für Text- und Binärformate ist eine große Vielfalt an Formaten generierbar bzw. lesbar.
Beachten Sie bitte, dass nicht jedes Format an jeder Stelle im Programm, wo Daten gelesen bzw. geschrieben werden, unterstützt wird.
Einzelne Elemente speichern direkt in einem MS-Office Format (z. B. Kennfelder im Excel-Format *.xml).

### 9.1.1 Textformat

Dieses universelle Datenformat wird vorrangig zum Datenaustausch mit Office-Anwendungen eingesetzt. Optional können Spalten- und Dezimaltrennzeichen festgelegt werden (**Bild 9.1**).

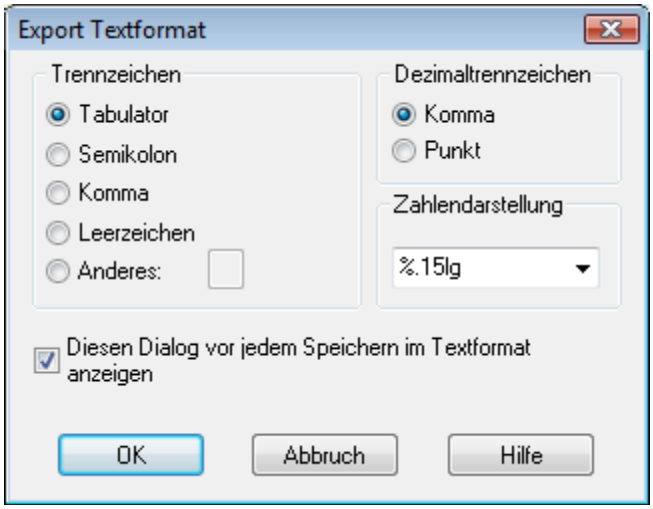



**Bild 9.1:** Optionale Einstellungen für Textdateien

#### 9.1.1.1 Datei speichern

Sollen Daten (z.B. Ergebnisverläufe, Kennlinien) im Textformat gespeichert werden, ist im entsprechenden Dateidialog ( ) als erstes der Dateityp *Textdatei (*.txt)* mittels Combobox auszuwählen. Danach ist die Schaltfläche *Optionen* freigegeben. Klicken Sie auf diese Schaltfläche wird der Einstellungsdialog für das Textformat geöffnet (**Bild 9.1**). In diesem Dialog werden die trennzeichenbestimmenden Einstellungen vorgenommen. Wählen Sie das gewünschte Trennzeichen mittels Radioknopf aus. Unter Trennzeichen wird hier das Spaltentrennzeichen verstanden.
Gesondert wird das Dezimaltrennzeichen zur Ausgabe von Zahlen gewählt. Diese Auswahl ist unabhängig von der aktuellen Ländereinstellung ihres Rechners.
Bei markierter Checkbox „***Diesen Dialog vor jedem ...***“ werden die globalen Einstellungen vor jedem Speichern abgefragt, ohne dass die Schaltfläche ***Optionen*** geklickt wurde. Ist die Checkbox nicht markiert, gelten die Einstellungen für jede im Textformat zu speichernde Datei bis zu einer neuen globalen Einstellung (Klick auf ***Optionen*** im Dateidialog).
Als weiteres Textformat steht das Format *CSV (Trennzeichen getrennt) (*.csv)* zur Verfügung. Wenn Sie mittels Combobox dieses Format als *Dateityp* ausgewählt haben, können keine optionalen Einstellungen vorgenommen werden. Als Spaltentrennzeichen wird ein Komma (,) und als Dezimaltrennzeichen ein Punkt (.) ausgegeben.

Um **alle** protokollierten Ergebnisse eines Modells in eine Textdatei zu exportieren, verwenden Sie unter ***Extras/Export*** den Befehl ***Ergebnisse…***

#### 9.1.1.2 Datei laden

Ein Assistent unterstützt den Anwender beim Laden von externen Daten. In Abhängigkeit vom Element, welches Daten lädt, kann die Seitenanzahl des Assistenten schwanken, d.h. die dort fehlenden Seiten haben für das betreffende Element keine Bedeutung. Im Folgenden wird der Import von Textdaten am Beispiel des Elements „Kennlinie“ erläutert.

.. Wählen Sie im Dateidialog den Dateityp Textdatei (*.txt, *.csv). Alle Dateien dieser Typen werden angezeigt. Wählen Sie die zu importierende Datei aus und klicken Sie Öffnen. Der Import-Assistent wird aktiviert.

- ***Import - Textformat Seite 1***

  Die erste Seite (**Bild 9.2**) gestattet die Auswahl der Trennzeichen der vorliegenden Datei. Siehe dazu auch die Einstellungen, welche unter Datei speichern (**9.1.1.1**) vorgenommen wurden. Im Feld „***Import beginnen mit***“ wird die erste zu importierende Datenzeile eingestellt (evtl. Kopfzeilen können so überlesen werden).
  Der untere Bereich der ersten Seite zeigt die Daten, wie sie in der Datei

enthalten sind. Klicken Sie auf die eine oder andere Option, wird die Anzeige aktualisiert. Sie müssen also das konkrete Format der zu importierenden Daten kennen und können vorab prüfen ob die Formateinstellungen das gewünschte Ergebnis liefern werden.

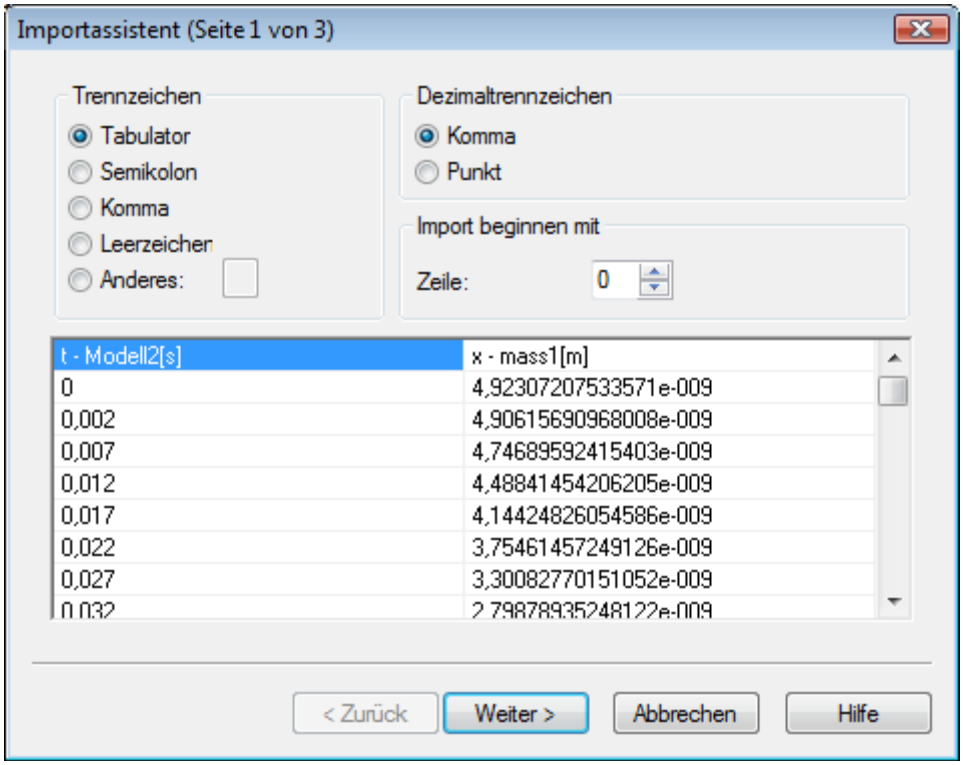



**Bild 9.2:** Import-Assitent, Textformat Seite 1

- ***Import - Textformat Seite 2***

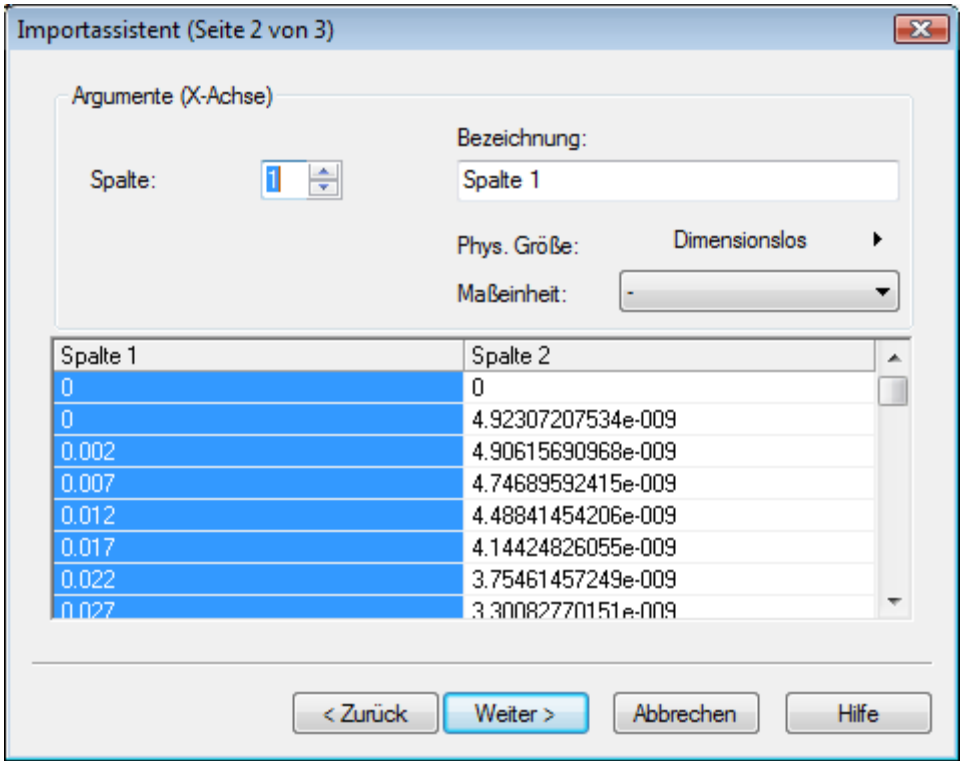



**Bild 9.3:** Import-Assistent, Textformat Seite 2 (Beispiel)

Auf der 2. Seite des Importassistenten wird die Spalte ausgewählt, die die Argumente (X-Werte) der Stützstellen liefern soll. Die Auswahl erfolgt durch Eingabe der Spaltennummer im Feld *Spalte* bzw. durch Klicken auf die Tasten ⬍ neben dem Eingabefeld. Die ausgewählte Spalte wird farblich hervorgehoben und die Werte werden entsprechend der Trennzeicheneinstellung konvertiert und angezeigt. Gegebenenfalls sind Korrekturen auf Seite 1 vorzunehmen.

- ***Import - Textformat Seite 3***

  Wählen Sie auf dieser Seite die Spalte aus, welche die Y-Werte enthält. Die Auswahl erfolgt durch Eingabe der Spaltennummer im Feld *Spalte* bzw. durch Klicken auf die Tasten ⬍ neben dem Eingabefeld. Die ausgewählte Spalte wird farblich hervorgehoben und die Werte werden entsprechend der Trennzeicheneinstellung konvertiert und angezeigt (analog der Seite 2).
  Bei einigen Elementen (z. B. Hysterese) kann sich der Vorgang auf Seite 3 mehrmals wiederholen.

## 9.1.2 IEEE-Binärformat

Dieses kompakte Binärformat wird hauptsächlich zum Datenaustausch mit signalverarbeitenden Programmen (z. B.: Messwerterfassung und -verarbeitung) genutzt.
Die Daten werden nach dem IEEE Standard gespeichert. Eine Reihe von Optionen (**Bild 9.4)** gestattet die Anpassung an alle definierten Binärformate.
Grundsätzlich werden für jeden Punkt ein X-Wert und ein Y-Wert gespeichert (jeder Ergebnisverlauf hat zwei Kanäle).

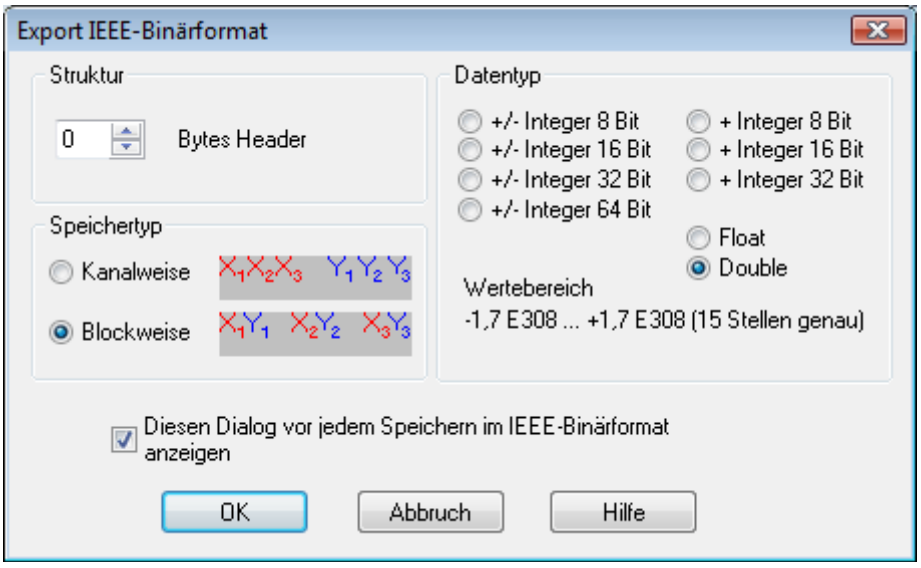



**Bild 9.4:** Einstellungen für Binärformat

### 9.1.2.1 Datei speichern

Sollen Daten (z.B. Ergebnisverläufe) im Binärformat gespeichert werden, ist im entsprechenden Dateidialog ( ) als erstes der Dateityp IEEE-Binärformat (*.bin) mittels Combobox auszuwählen. Danach ist die Schaltfläche Optionen freigege-

ben. Klicken Sie auf diese Schaltfläche wird der Einstellungsdialog für das Binärformat geöffnet **(Bild 9.4)**. In diesem Dialog sind alle formatbestimmenden Eingaben vorzunehmen. In der Abteilung Struktur kann den Daten eine bestimmte Anzahl Bytes vorangestellt werden. Das ist besonders für jene Programme interessant, welche beim Lesen von Binärdaten bestimmte Kopfinformationen erwarten.
Unter Datentyp wird festgelegt, für welchen Wertebereich und mit welcher Genauigkeit die Daten gespeichert werden.
Schließlich legt man mit dem Speichertyp fest, ob die Daten kanalweise (kanalorientiert, zeilenorientiert) oder blockweise (blockorientiert, spaltenorientiert) gespeichert werden sollen.
Die Checkbox *„**Diesen Dialog vor jedem ...**“* wirkt, wie unter **9.1.1.1** beschrieben.

### 9.1.2.2 Datei laden

Ein Assistent unterstützt den Anwender beim Laden von externen Daten. In Abhängigkeit vom Element, welches Daten lädt, kann die Seitenanzahl des Assistenten schwanken, d.h. die dort fehlenden Seiten haben für das betreffende Element keine Bedeutung. Im Folgenden wird der Binärdatenimport am Beispiel des Elements *Kennlinie* erläutert.

Wählen Sie im Dateidialog den Dateityp *IEEE-Binärformat (*.bin)*. Alle Dateien dieses Typs werden angezeigt. Wählen Sie die zu importierende Datei aus und klicken Sie ***Öffnen***. Der Import-Assistent wird aktiviert.

- ***Import - IEEE Binärformat Seite 1***

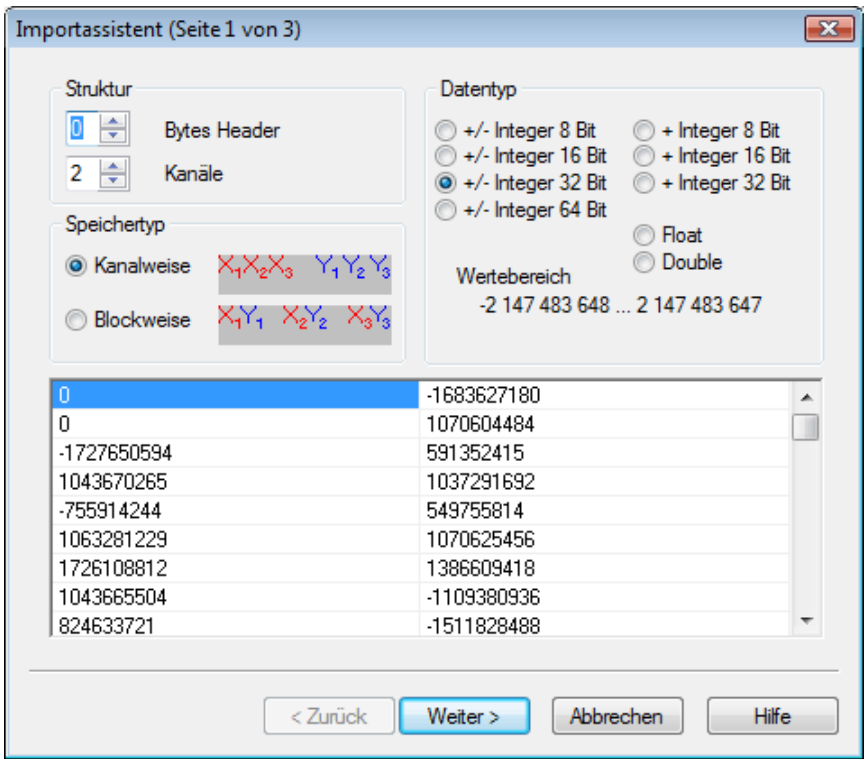



**Bild 9.5:** Import-Assistent, Binärformat Seite 1

Die erste Seite (**Bild 9.5**) gestattet die Auswahl des Formats der vorliegenden Datei. Siehe dazu auch die Einstellungen, welche unter Datei speichern (weiter oben) vorgenommen wurden. Der untere Bereich der ersten Seite zeigt die Daten, wie sie entsprechend der vorgenommenen Einstellungen importiert werden würden. Klicken Sie auf die eine oder andere Option, wird die Anzeige aktualisiert. Sie müssen also das konkrete Format der zu importierenden Daten kennen und können vorab prüfen ob die Formateinstellungen das gewünschte Ergebnis liefern werden.

- ***Import - IEEE Binärformat Seit 2***

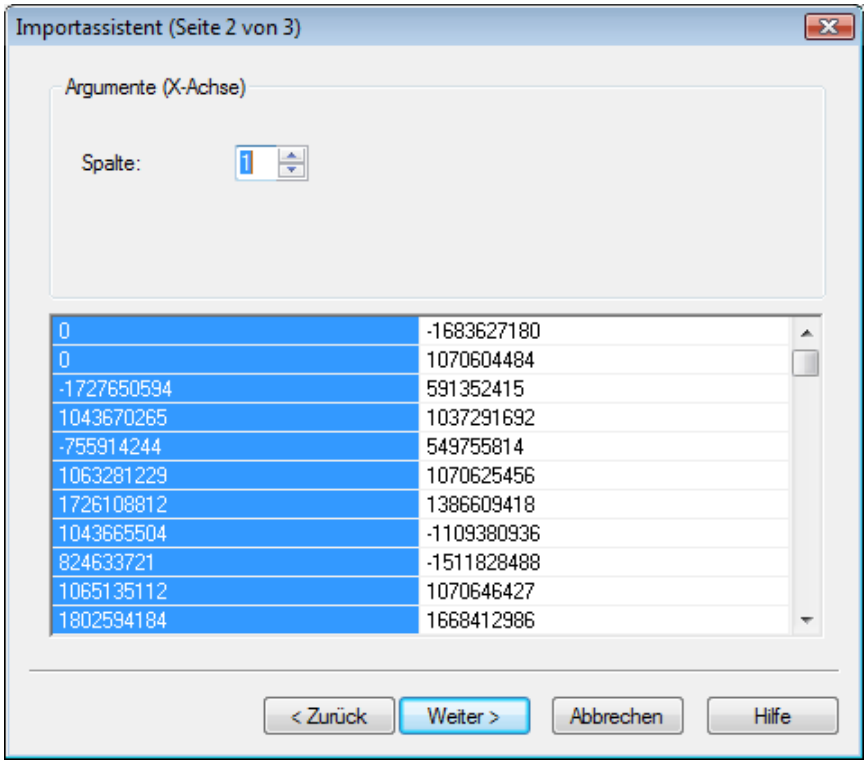



**Bild 9.6:** Import-Assistent, Binärformat Seite 2 (Beispiel)

Die Auswahl erfolgt durch Eingabe der Spaltennummer im Feld *Spalte* bzw. durch Klicken auf die Tasten ⬍ neben dem Eingabefeld. Die ausgewählte Spalte wird farblich hervorgehoben, die Werte werden entsprechend der Formateinstellung (Seite 1) angezeigt (**Bild 9.6**).

- ***Import - IEEE Binärformat Seite 3***
  Wählen Sie auf dieser Seite (**Bild 9.7**) die Spalte aus, welche die Y-Werte enthält. Die Auswahl erfolgt durch Eingabe der Spaltennummer im Feld *Spalte* bzw. durch Klicken auf die Tasten ⬍ neben dem Eingabefeld. Die ausgewählte Spalte wird farblich hervorgehoben, die Werte werden entsprechend der Formateinstellung (Seite 1) angezeigt.

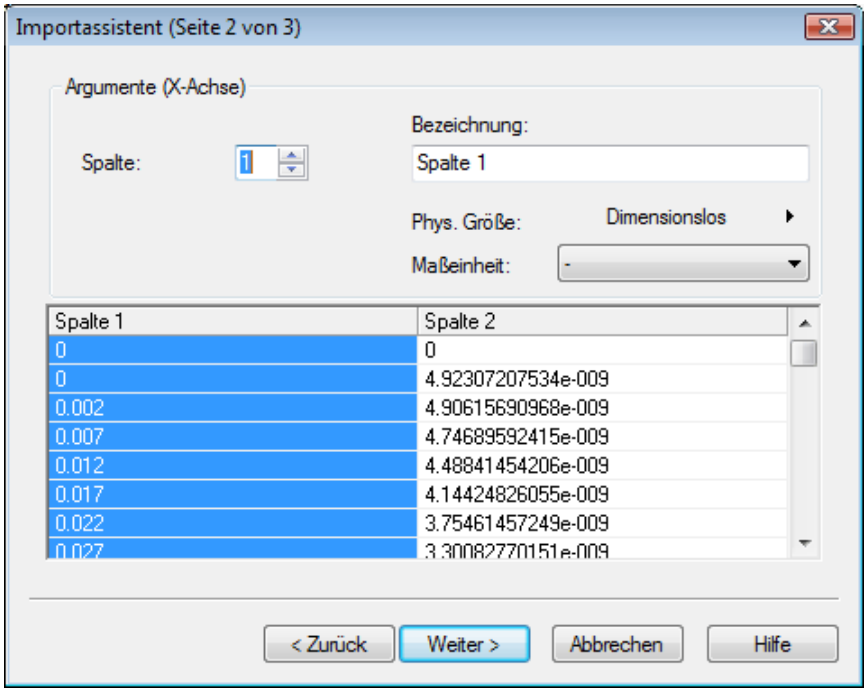



**Bild 9.7:** Import-Assistent, Binärformat Seite 3

### 9.1.3 ITI-Binärformat (*.rfs) und ITI-ASCII-Format (*.rfu)

Die ITI-Hausformate unterstützten den Datenaustausch zwischen ITI-Produkten. Im einfachsten Fall ist das z.B. der *Transport* eines Ergebnisverlaufs von einem Ergebnisfenster (auf Rechner 1) in ein anderes Ergebnisfenster (auf Rechner 2). Der Vorteil dieses Formates besteht darin, dass nicht erst eine Anzahl von Optionen zu spezifizieren sind. Nachteilig wirkt sich der eingeschränkte Kreis von *Tauschpartnern* aus.
Es handelt sich um ein kompaktes Datenformat zum schnellen Austausch von Ergebnisverläufen. Es sind keine speziellen Einstellungen erforderlich.

### 9.1.4 XML-Format

Es handelt sich um ein universelles Datenformat für den Austausch von Informationen zu anderen Anwendungen, z. B. Internet-Applikationen oder Textverarbeitung. Ein Import in diesem Format wird nicht unterstützt.

### 9.1.5 DIAdem-Headerformat

Dieses Format gestattet das Speichern von Ergebnisverläufen im DIAdem-Format der GfS GmbH Aachen. Dabei entsteht eine *Name*.dat Datei mit allen verlaufsbeschreibenden Parametern und eine *Name*.bin Datei, welche die eigentlichen Funktionswerte enthält. Eine genaue Beschreibung dieses Formats finden Sie in Ihrem DIAdem-Handbuch.

## 9.2 COM Interface

SimulationX verfügt über eine COM-Schnittstelle, die es ermöglicht, typische, immer wiederkehrende Arbeitsabläufe zu automatisieren.

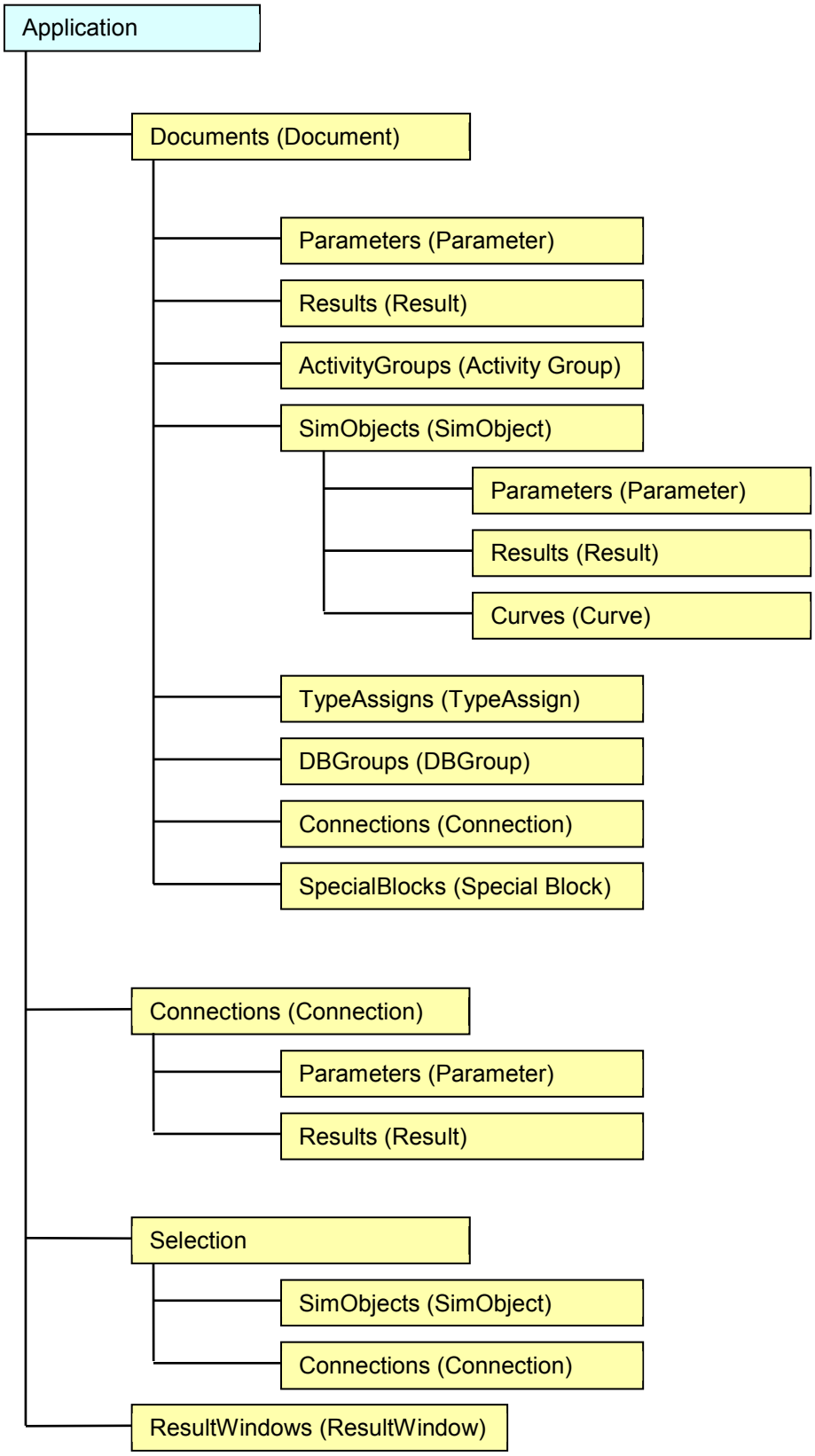



**Bild 9.8:** Objektstruktur

Für die Nutzung dieses Interfaces stehen Ihnen die verschiedensten Möglichkeiten offen:

- integrierter Script-Editor mit der Möglichkeit zum Aufzeichnen von
- Makros
- Visual Basic for Applications (VBA) als Bestandteil der MS-Office-Anwendungen
- Windows Script (VBScript, JScript)
- Visual Basic Anwendungen
- C/C++ Applikationen, ...

Die im Weiteren aufgeführten Beispiele sind in VBScript implementiert.
Über Grundsätze der Bereitstellung und Nutzen von COM-Schnittstellen informieren Sie sich bitte in der einschlägigen Fachliteratur, z. B unter.
http://msdn.microsoft.com/de-de/library/ms950396.aspx.

## 9.2.1 Klassen

### 9.2.1.1 Application

Die Klasse **Application** repräsentiert den Anwendungsrahmen von SimulationX. Sie verwaltet Simulationsmodelle in der Form von Documents (Modellen).
Sie starten das Programm, indem Sie ein Objekt der Klasse **Application** erzeugen.

```
Set sim = CreateObject("iti.simx3")
```

Die Eigenschaft „Visible“ legt die Sichtbarkeit von SimulationsX fest. Initial bleibt das Hauptfenster unsichtbar.

```
sim.Visible = true
```

### 9.2.1.2 Documents

Die Auflistungsklasse **Documents** verwaltet alle geladenen Simulationsmodelle als Objekte vom Typ **Document**. Ein neues, leeres Modell können Sie mit folgender Anweisung erstellen:

```
Set doc1 = sim.Documents.Add
```

Ein bereits vorhandenes Modell öffnen Sie mit der Methode **Open**.

```
'Öffnene einer Datei
Set doc2 = sim.Documents.Open ("c:\user\model.isx")

'Öffnen eines Typs
Set doc2 = sim.Documents.Open ("Modelica.Blocks.Examples.PID_Controller")
```

Als Rückgabewert beider Methoden erhalten sie ein Objekt der Klasse **Document**.

Über die Klasse **Document** lassen sich Simulationsrechnungen steuern. Mit den folgenden VBScript-Anweisungen starten Sie eine transiente Simulation und warten, bis die Rechnung abgeschlossen ist. Die Eigenschaft **SolutionState** informiert Sie dabei über den aktuellen Zustand der Rechnung.

```
const simStopped = 16
doc.Reset
doc.Start
While doc.SolutionState <> simStopped
Wend
MsgBox "Ready..."
```

Die Methode **Start** verfügt über einen optionalen Parameter ***SimCalculationMode***, der die Auswahl des Simulationsverfahrens (transient oder stationär) erlaubt. Mit der folgenden Anweisungsfolge starten Sie damit eine stationäre Simulation. Beachten Sie dabei die Unterschiede beim Warten auf die Beendigung der Rechnung.

```
const simReady = 2
const SimCalculationModeSteadyState =

doc.Reset
doc.Start SimCalculationModeSteadyState
While doc.SolutionState <> simReady
Wend
MsgBox "Ready..."
```

Die Eigenschaft **Document.Solver** erlaubt die Umschaltung des Lösungsverfahrens. Folgende Werte werden in Abhängigkeit der jeweiligen Lizenz unterstützt:

- simSolverBDF = 0
- simSolverMEBDF = 1
- simSolverCVODE = 2
- simSolverFixedStep = 3

Der Export von C-Code kann über die Funktion **Document.ExportSourceCode** vollständig per Script gesteuert werden. Wenn beim Aufruf des Code-Exports über die COM-Schnittstelle keine Eingänge, Ausgänge bzw. Parameter übergeben werden, werden die im Modell gespeicherten Informationen verwendet.

#### 9.2.1.3 SimObjects

Die Auflistungsklasse **SimObjects** verwaltet die Elemente innerhalb eines Modells oder innerhalb der aktuellen Auswahl (Klasse **Selection**). Ein neues Element vom Typ "Translatorische Masse" mit dem Namen "m2" erzeugen Sie durch Aufruf der Methode **Add**.

```
doc1.SimObjects.Add "Mechanics.Translation.Mass", "m1"
```

Optional können Sie als weitere Parameter die X- und Y-Position des neuene Elements (in Pixeln) innerhalb der Strukturansicht angeben.

```
doc1.SimObjects.Add "Mechanics.Translation.Mass", "m1", 90,
90
```

Rückgabewert der Methode SimObjects.Add ist ein Objekt der Klasse **SimObject**. Der Zugriff auf ein bestimmtes Element erfolgt in aller Regel über die Angabe seines Namens.

```
Set objMass = doc1.SimObjects("m1")
```

Mit einer For-Each-Anweisungsfolge erhalten Sie Zugriff auf alle Elemente eines Modells oder der aktuellen Auswahl (siehe **4.1.1.8**.)

#### 9.2.1.4 Connections

Die Auflistungsklasse **Connections** verwaltet die Verbindungen innerhalb eines Dokuments. Eine neue Verbindung mit dem Namen "Connection1" zwischen den Anschlusspunkten zweier Elemente kann mit folgendem Befehl gezogen werden.

```
doc.Connections.Add "m1.ctr2", "Spring1.ctr1",
"Connection1"
```

#### 9.2.1.5 Parameters

Die Auflistungsklasse **Parameters** ist in verschiedenen Klassen des ITI SimulationX-Objektmodells verfügbar. Sie gestattet den Zugriff auf Parameter von Modellen, Elementen und Verbindungen. Auf ein und denselben Parameter kann auf unterschiedliche Art und Weise zugegriffen werden.

```
Set mass1 = doc.Parameters("m1.m")
Set mass2 = doc.SimObjects("m1").Parameters("m")
```

Beide Anweisungen liefern das gleiche Ergebnis. Der Wert eines Parameters wird durch die Eigenschaft **Value** repräsentiert, wobei eine Zuweisung von Zahlen und Funktionsausdrücken zulässig ist.

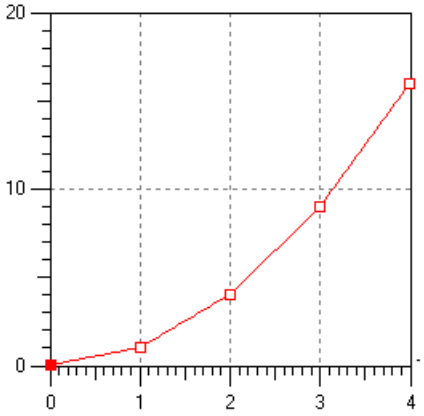
```
doc.Parameters("m1.m").Value = 1.0
doc.Parameters("Source1.F").Value = "9.81*m1.m"
```

Das Startattribut von Parametern lässt sich über die Eigenschaft **Parameter.Start** schreiben bzw. lesen.

### 9.2.1.6 Curves

Die Auflistungsklasse **Curves** gestattet den Zugriff auf Kennlinienobjekte innerhalb von Elementen eines Simulationsmodells. Mit den folgenden Anweisungen können Sie die Werte einer Kennlinie setzen.

```
Dim x(4)
Dim y(4)
For i=LBound(x) To UBound(x)
    x(i) = i
    y(i) = i*i
Next
Set curve = doc.SimObjects("Curve1").Curves("curve")
curve.XSequence.CurveValues = x
curve.YSequences(1).CurveValues = y
```




Der Zugriff auf die Werte erfolgt standardmäßig in SI-Einheiten. Die Verwendung der jeweils eingestellten Maßeinheit kann mit einem optionalen Parameter erzwungen werden.

```
' Vorgabe der Maßeinheit
curve.YSequences(1).Quantity =
"Mechanics.Translation.Displace"
curve.YSequences(1).Unit = "mm"
' in SI unit
curve.YSequences(1).CurveValues(TRUE) = y
' in current unit
curve.YSequences(1).CurveValues(FALSE) = y
```

### 9.2.1.7 Results

Die Auflistungsklasse **Results** ist in verschiedenen Klassen des SimulationX-Objektmodells verfügbar. Sie gestattet den Zugriff auf Ergebnisgrößen von Modellen, Elementen und Verbindungen. Auf ein und dieselbe Ergebnisgröße kann auf unterschiedliche Art und Weise zugegriffen werden.

```
Set x1 = doc.Results("Connection1.x")
Set x2 = doc.Connections("Connection1").Results("x")
```

Beide Anweisungen liefern das gleiche Ergebnis. Das Protokoll einer Ergebnisgröße kann durch Setzen der Eigenschaft **Protocol** eingeschaltet bzw. ausgeschaltet werden. Zur Auswertung der Simulationsrechnung stellt die Klasse **Result** die Eigenschaften **MinValue**, **MaxValue** (Minimum und Maximum), **MeanValue** (Mittelwert), **RMSValue** (Effektivwert) sowie **LastValue** (letzter Wert) bereit. Folgende Anweisung zeigt Minimum und Maximum der Ergebnisgröße x1 in einer Messagebox an.

```
MsgBox "Min: " & x1.MinValue & ", Max: " & x1.MaxValue
```

Der Zugriff auf alle Werte einer Ergebnisgröße wird durch die Eigenschaft **ProtValues** realisiert.

```
For Each val In x1.ProtValues
     MsgBox "Wert: " & val
Next
```

Der Zugriff auf die Werte erfolgt standardmäßig in SI-Einheiten. Die Verwendung der jeweils eingestellten Maßeinheit kann mit einem optionalen Parameter erzwungen werden.

```
' in SI unit
For Each val In x1.ProtValues(vbTrue)
     MsgBox "Wert: " & val
Next
' in current unit
For Each val In x1.ProtValues(vbFalse)
     MsgBox "Wert: " & val
Next
```

Der Zugriff auf die Ergebnisse einer stationären Simulation erfolgt in ähnlicher Art und Weise über die Eigenschaft **SteadyStateProtValues**. Der Zugriff auf die Ergebnisse erfordert die Angabe weiterer Parameter:

| SimSteadyStateResult | steadyStateResult | Art der Ergebnisgröße |
|---|---|---|
| Int | order | Ordnung |
| bool | bSI | Ergebnisse in SI-Einheiten (optional, Standardwert=vbTrue) |

Das folgende Code-Fragment kopiert einen Ergebnisverlauf aus der stationären Simulation in die Spalte einer Excel-Tabelle.

```
const simSteadyStateArgument                = 0
const simSteadyStateAmplitude               = 1
const simSteadyStateFluctuationCoefficient  = 2
```

```
const simSteadyStateExcitation                = 3
const simSteadyStatePhase                     = 4
const simSteadyStateRealPart                  = 5
const simSteadyStateImagPart                  = 6

const simSteadyStateSumMask                   = 16
const simSteadyStateMeanMask                  = 32

res  = simSteadyStateExcitation
order    = 2.0

values = om.SteadyStateProtValues(res, order, vbFalse)

row = 2
col = 1
For Each v In val
     excel.Cells(row, col) = v
     row = row + 1
Next
```

Um auf die jeweilige Summenkurve bzw. den Mittelwert zuzugreifen, muss der Ergebnistyp mit der entsprechenden Maske versehen werden. Der Ordnungsparameter hat in diesem Fall keine Bedeutung.

```
om.SteadyStateProtValues(res+simSteadyStateSumMask, …
om.SteadyStateProtValues(res+simSteadyStateMeanMask, …
```

Das vollständige Beispiel *SteadyStateResults2Excel.vbs* finden Sie in der SimulationX-Beispielsammlung.

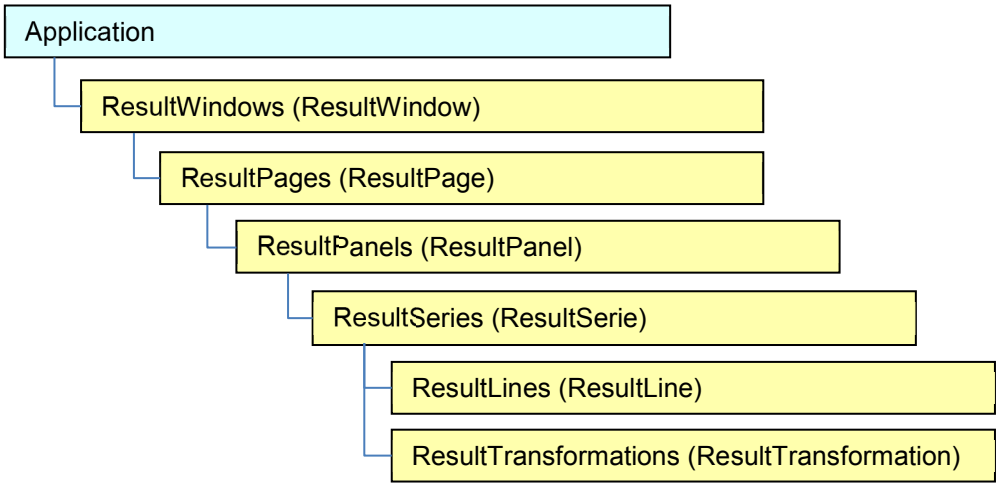
#### 9.2.1.8 Selection

Die Klasse **Selection** bietet Zugriff auf die aktuelle Auswahl eines Simulationsmodells. Über die Eigenschaften **SimObjects** und **Connections** ist es möglich, auf alle aktuell selektierten Elemente bzw. Verbindungen zuzugreifen. Folgende Befehle zeigen die Namen und Beschreibungen (Kommentare) aller selektierten Elemente an.

```
For Each obj In doc1.Selection.SimObjects
    MsgBox "Name: " & obj.name & ", Comment: " &
obj.comment
Next
```

#### 9.2.1.9 ResultWindows

Die Auflistungsklasse **ResultWindows** beinhaltet die Ergebnisfenster von SimulationX. Da Ergebnisfenster Verläufe aus unterschiedlichen Modellen (Documents) enthalten dürfen, werden ResultWindows durch das Application-Objekt verwaltet.




Mit den folgenden Anweisungen öffnen Sie ein neues Ergebnisfenster für die Größe "Connection1.x" und stellen den Darstellungsbereich für die X- bzw. Y-Achse ein.

```
Set wnd = sim.ResultWindows.Add("Connection1.x")
wnd.SetMinMax 0.0, 1.0, -0.01, +0.01
```

Die Eigenschaften aller für die SimulationX-Ergebnisfenster relevanten Klassen können Sie über die Methoden *GetProperty* bzw. *SetProperty* abfragen und setzen. Die Namen der Eigenschaften der jeweiligen Klasse können Sie der Eigenschaftsleiste der Ergebnisfenster entnehmen.
Aus Kompatibilitätsgründen wurden bestehende Eigenschaften und Methoden beibehalten.

Die Eigenschaft *ResultWindow.FrozenResultMode* erlaubt die Einstellung der Behandlung von eingefrorenen Verläufen. Folgende Werte werden unterstützt:

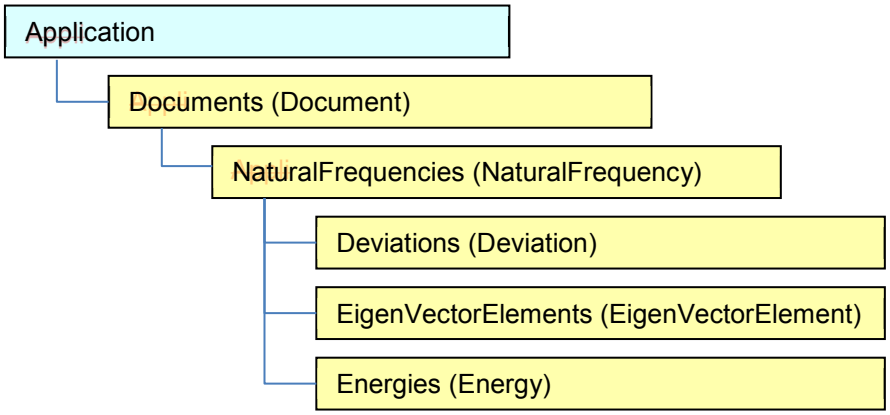
- simFrozenResultModelCurve = 0 für eigenständige Kurven
- simFrozeResultsModeHistory = 1 für nichtselbständige Verläufe

### 9.2.1.10 NaturalFrequencies

Die Auflistungsklasse **NaturalFrequencies** ermöglicht den Zugriff auf die Ergebnisse der Eigenfrequenzanalyse.

Mit folgender Anweisungsfolge können Sie eine Eigenfrequenzanalyse starten:

```
set nf = doc.NaturalFrequencies
nf.DoCalculation vbFalse
```




Der optionale Parameter *bIgnoreFilter* (Standardwert: vbTrue) gibt an, ob die im Eigenschaftsdialog für die Eigenfrequenzanalyse angegebenen Filter ignoriert und somit alle ermittelten Eigenfrequenzen zurückgegeben werden sollen.
Mit dem folgenden Code-Fragment können Sie die ermittelten Eigenwerte in eine Excel-Tabelle kopieren.

```
row = 1
excel.Cells(row, 1) = "No."
excel.Cells(row, 2) = "Re"
excel.Cells(row, 3) = "Im"
excel.Cells(row, 4) = "Abs"
excel.Cells(row, 5) = "Arg"
excel.Cells(row, 6) = "f[Hz] (damped)"
excel.Cells(row, 7) = "f[Hz] (undamped)"
excel.Cells(row, 8) = "D[-]"
excel.Cells(row, 9) = "Time Constant[s]"
ApplyFormat row, 1, 9, vbTrue, vbFalse, RGB(0, 0, 0)

for each f in nf
     row = row + 1
     excel.Cells(row, 1) = "f" & CStr(row-1)
     excel.Cells(row, 2) = f.Re
     excel.Cells(row, 3) = f.Im
     excel.Cells(row, 4) = f.Abs
     excel.Cells(row, 5) = f.Arg
     excel.Cells(row, 6) = f.FDamped
     excel.Cells(row, 7) = f.FUndamped
     excel.Cells(row, 8) = f.D
     excel.Cells(row, 9) = f.T
next
```

Die Aufzählungsklassen **Deviations**, **EigenVectorElements** und **Energies** stellen Eigenschaften bereit, die den Zugriff auf die während der Analyse berechneten Werte erlauben. Über die folgenden Anweisungen ermitteln Sie die Energieverteilung für eine bestimmte Eigenfrequenz. Auf die Auslenkungen bzw die Eigenvektoren greifen sie in der gleichen Art und Weise zu.

```
for each f in nf
     row = 0
     for each e in f.PotentialEnergies
          row = row + 1
          excel.Cells(row, 1) = e.Owner.GetRelIdent(doc)
          excel.Cells(row, 2) = e.Value
     next
     for each e in f.KineticEnergies
          row = row + 1
          excel.Cells(row, 1) = e.Owner.GetRelIdent(doc)
          excel.Cells(row, 2) = e.Value
     next
     for each e in f.LossEnergies
          row = row + 1
          excel.Cells(row, 1) = e.Owner.GetRelIdent(doc)
          excel.Cells(row, 2) = e.Value
     next
     for each e in f.AddedEnergies
          row = row + 1
          excel.Cells(row, 1) = e.Owner.GetRelIdent(doc)
          excel.Cells(row, 2) = e.Value
     next
next
```

Das vollständige Beispiel *NaturalFrequencies2Excel.vbs* finden sie in der SimulationX-Beispielsammlung.

### 9.2.2 Beispiele

Folgende Beispiele sollen Ihnen die Nutzung der COM-Schnittstelle von ITI SimulationX demonstrieren.

Unter der Adresse http://msdn.microsoft.com/scripting/ finden Sie weitere Informationen zur Nutzung der Sprachen VBScript bzw. JScript.
Für die Ausführung der Programme in VBScript bzw. JScript auf Ihrem Computer sind folgende Arbeitsschritte nötig:

1. Erfassen Sie den Quelltext mit einem beliebigen Editor.
2. Speichern Sie die Datei unter einem frei gewählten Namen mit der Dateierweiterung *.vbs bzw. *js.
3. Starten Sie den Script mit einem Doppelklick auf das zugehörige Symbol in der Dateiansicht des Windows-Explorers.

Für die Ausführung des ITI-Scripts starten Sie den ITI-Script Editor (Menü *Extras/Makro/ITI-Script Editor*). Um diesen zu nutzen, benötigen Sie den Microsoft Script Debugger. Weitere Informationen finden Sie unter http://msdn.microsoft.com/scripting/.

#### 9.2.2.1 VBScript-Beispiel

```
' Konstanten
simStopped = 16
' Starten von SimulationX mit einem neuen Dokument
Set sim = CreateObject("iti.simx3")
Set doc = sim.Documents.Add()

' Entfernen Sie den Kommentar vor folgender Zeile,
' wenn SimulationX während der Ausführung des Makros
' sichtbar bleiben soll
' sim.visible = true
' Aufbau eines einfachen Simulationsmodells
doc.SimObjects.Add "Mechanics.Translation.Mass", "Mass1", 90, 90
doc.SimObjects.Add "Mechanics.Translation.Spring", "Spring1", 180, 90
doc.SimObjects.Add "Mechanics.Translation.Damper", "Damper1", 180, 150
doc.Connections.Add "Mass1.ctr2", "Spring1.ctr1", "Connection1"
doc.Connections("Connection1").AddLine "Damper1.ctr1", 145, 105

' Parametrieren des Modells
doc.SimObjects("Mass1").Parameters("m").Value = 0.5
doc.SimObjects("Spring1").Parameters("k").Value = 1000
doc.Connections("Connection1").Parameters("x0").Value = 0.01

doc.Connections("Connection1").Results("x").Protocol = True

' Öffnen und Konfigurieren eines Ergebnisfensters
Set wnd = sim.ResultWindows.Add("Connection1.x")
wnd.SetMinMax 0.0, 1.0, -0.01, +0.01

' Variantenrechnung Dämpfung b = 0 ... 10
For b = 0.0 To 10.0 Step 1.0

    doc.Parameters("Damper1.b").Value = b
    doc.Reset
    doc.Start

    ' warten, bis Rechnung beendet
    While (doc.SolutionState <> simStopped)
      wscript.Sleep(100)
    Wend

Next

' SimulationX beenden
MsgBox "SimulationX wird geschlossen..."
sim.Quit
```

#### 9.2.2.2 JScript-Beispiel

```
// Konstanten
var simStopped = 16
// Start von SimulationX mit einem neuen Dokument
var sim = new ActiveXObject("iti.simx3");
var doc = sim.Documents.Add();
```

```
// Entfernen Sie den Kommentar vor folgender Zeile,
// wenn SimulationX während der Ausführung des Makros
// sichtbar bleiben soll
// sim.visible
// Aufbau eines einfachen Simulationsmodells
doc.SimObjects.Add( "Mechanics.Translation.Mass", "Mass1", 90, 90);
doc.SimObjects.Add( "Mechanics.Translation.Spring", "Spring1", 180,
90);
doc.SimObjects.Add( "Mechanics.Translation.Damper", "Damper1", 180,
150);
doc.Connections.Add( "Mass1.ctr2", "Spring1.ctr1", "Connection1");
doc.Connections("Connection1").AddLine( "Damper1.ctr1", 145, 105);
// Parametrieren des Modells
doc.SimObjects("Mass1").Parameters("m").Value = 0.5;
doc.SimObjects("Spring1").Parameters("k").Value = 1000;
doc.Connections("Connection1").Parameters("x0").Value = 0.01;
doc.Connections("Connection1").Results("x").Protocol = true;
doc.Connections("Connection1").Results("v").Protocol = true;
// Öffnen von Ergebnisfenstern
var wnd = sim.ResultWindows.Add("Connection1.x");
wnd.SetMinMax(0.0, 1.0, -0.01, +0.01);
// Variantenrechnung Dämpfung b = 0...10
for (b=0.0; b<=10.0; b+=1.0)
{
    doc.Parameters("Damper1.b").Value = b;
    doc.Reset();
    doc.Start();
    while(doc.SolutionState != simStopped)
        WScript.Sleep(100);
}
var WSHShell = WScript.CreateObject("WScript.Shell");
WSHShell.Popup("Variantenrechnung beendet");
// SimulationX beenden
sim.Quit();
```

### 9.2.2.3 ITI-Script-Beispiel

```
Sub Sample()
   ' Konstanten
   simStopped = 16
   ' Starten von SimulationX mit einem neuen Dokument
   Set doc = Documents.Add()
   ' Aufbau eines einfachen Simulationsmodells
   doc.SimObjects.Add "Mechanics.Translation.Mass", "Mass1", 90, 90
   doc.SimObjects.Add "Mechanics.Translation.Spring", "Spring1", 180,
   90
   doc.SimObjects.Add "Mechanics.Translation.Damper", "Damper1", 180,
   150
   doc.Connections.Add "Mass1.ctr2", "Spring1.ctr1", "Connection1"
   doc.Connections("Connection1").AddLine "Damper1.ctr1", 145, 105
   ' Parametrieren des Modells
   doc.SimObjects("Mass1").Parameters("m").Value = 0.5
   doc.SimObjects("Spring1").Parameters("k").Value = 1000
   doc.Connections("Connection1").Parameters("x0").Value = 0.01
   doc.Connections("Connection1").Results("x").Protocol = True
' Öffnen und Konfigurieren eines Ergebnisfensters
   Set wnd = ResultWindows.Add("Connection1.x")
   wnd.SetMinMax 0.0, 1.0, -0.01, +0.01
```

```
   ' Variantenrechnung Dämpfung b = 0 ... 10
   For b = 0.0 To 10.0 Step 1.0
      doc.Parameters("Damper1.b").Value = b
      doc.Reset
      doc.Start
      ' warten, bis Rechnung beendet
      While (doc.SolutionState <> simStopped)
          itiscript.Sleep(100)
      Wend
   Next
End Sub
```

### 9.2.3 Code-Export über die COM - Schnittstelle

In diesem Abschnitt soll die COM-Schnittstelle des Code-Exports beschrieben werden. Die Schnittstelle besteht im Wesentlichen aus den Funktionen *ExportSourceCode* und *ExportSourceCode2*, die zur Klasse *Document* gehören.

```
ExportSourceCode(   [in] SimCodeExportProject prjType,
                    [in] SimCodeExportSaveOutputsApproach,
                    [in] BSTR prjName,
                    [in] BSTR prjPath,
                    [in] int prjParam,
                    [in] SAFEARRAY(VARIANT) inputs,
                    [in] SAFEARRAY(VARIANT) outputs,
                    [in] SAFEARRAY(VARIANT) parameters,
                    [out, retval] SAFEARRAY(VARIANT)* msgs)

ExportSourceCode2(  [in] SimCodeExportProject prjType,
                    [in] SimCodeExportSaveOutputsApproach
                         outputsApproach,
                    [in] BSTR prjName,
                    [in] BSTR prjPath,
                    [in] int prjParam,
                    [in] SAFEARRAY(VARIANT) inputs,
                    [in] SAFEARRAY(VARIANT) outputs,
                    [in] SAFEARRAY(VARIANT) parameters,
                    [in] BSTR additionalLibs,
                    [in] BSTR additionalLibPath,
                    [out, retval] SAFEARRAY(VARIANT)* msgs)
```

Die Funktion *ExportSourceCode2* unterscheidet sich nur durch die zwei zusätzlichen Parameter *additionalLibs* und *additionalLibPath* von der Funktion *ExportSourceCode*. Die Angabe dieser Parameter ist dann notwendig, wenn das exportierte Modell **externe Funktionen** enthält, die in einer DLL (Dynamic Link Library) implementiert sind. Die so spezifizierten Bibliotheken werden dann beim Erstellen des ausführbaren Modells während des Link- Prozesses hinzugefügt.

<table>
<tr><th>Parameter</th><th>Beschreibung</th></tr>
<tr><td>prjType</td><td>Dieser Parameter spezifiziert den Code-Export Projekttyp. Folgende Projekttypen werden unterstützt:
<table>
<tr><th>Projekttyp</th><th>Wert</th></tr>
<tr><td>simCodeExportProjectWithoutSolver (C-Code ohne Solver)</td><td>0</td></tr>
<tr><td>simCodeExportProjectEmbeddedSolver (Ausführbares Modell)</td><td>1</td></tr>
<tr><td>simCodeExportProjectSFunction (Matlab/Simulink S-Function)</td><td>2</td></tr>
</table></td></tr>
<tr><td>outputsApproach</td><td>Dieser Parameter bestimmt wie die Simulationsergebnisse (die Ausgänge) protokolliert (gespeichert) werden. Folgende Protokollierungsarten werden unterstützt:
<table>
<tr><th>Projekttyp</th><th>Wert</th></tr>
<tr><td>simCodeExportSaveOutputsEqidistant (Äquidistantes Speichern der Ausgänge mit dtProtMin)</td><td>0</td></tr>
<tr><td>simCodeExportSaveOutputsAll (Speichern der Ausgänge zu jedem gültigen Schritt)</td><td>1</td></tr>
<tr><td>simCodeExportSaveOutputsAtleastwithdtProt (Speichern nach mindestens dtProtMin)</td><td>2</td></tr>
</table></td></tr>
<tr><td>prjName</td><td>Spezifiziert den Projektnamen.</td></tr>
<tr><td>prjPath</td><td>Bestimmt den Speicherort der zum Projekt gehörenden Dateien</td></tr>
<tr><td>prjParam</td><td>Dieser Parameter ist nur für den Projekttyp Ausführbares Modell relevant und legt den verwendeten Löser fest. Folgende Löser werden unterstützt:
<table>
<tr><th>Projekttyp</th><th>Wert</th></tr>
<tr><td>ITI Standard Solver (Löser mit variabler Schrittweite, geeignet für nicht sehr steife Modelle mit vielen Unstetigkeiten)</td><td>0</td></tr>
<tr><td>ITI Fixed-Step Solver (Löser mit konstanter Schrittweite, geeignet zum Schrittweitentest für Real-Time Umgebungen)</td><td>1</td></tr>
<tr><td>Sundials CVode Solver (Löser mit variabler Schrittweite, geeignet für steife Modelle)</td><td>2</td></tr>
</table></td></tr>
</table>

| | |
|---|---|
| `inputs` | Dieser Parameter stellt ein Array der Eingangsgrößen dar. Die einzelnen Arrayelemente sind vom Typ String und entsprechen den Namen der Eingangsgrößen. |
| `outputs` | Dieser Parameter stellt ein Array der Ausgangsgrößen dar. Die einzelnen Arrayelemente sind vom Typ String und entsprechen den Namen der Ausgangsgrößen. |
| `parameters` | Dieser Parameter stellt ein Array der Parametergrößen dar. Die einzelnen Arrayelemente sind vom Typ String und entsprechen den Namen der Parametergrößen. |
| `additionalLibs` | Dieser Parameter ist eine Zeichenkette mit zusätzlichen Bibliotheken (.lib- Dateien). Die einzelnen Bibliotheken werden durch ein Leerzeichen getrennt. |
| `additionalLibPath` | Dieser Parameter ist eine Zeichenkette mit Verzeichnissen, in denen die zusätzlichen Bibliotheken zu finden sind. Die einzelnen Verzeichnisse sind durch ein Semikolon getrennt. |

| Rückgabewert | Beschreibung |
|---|---|
| `msgs` | Die Funktion gibt ein Array mit Statusmeldungen zurück. Diese können vom Typ *Fehler*, *Warnung* oder *Information* sein. Im Normalfall ist das zurückgegebene Array leer. |

**Hinweis**: Werden keine Eingangs-, Ausgangs-, bzw. Parametergrößen übergeben oder sind diese nicht gültig, werden die entsprechenden Größen, falls vorhanden, direkt aus dem Modell verwendet.

### Beispiel (VBScript)

In dem folgenden Beispiel soll ein Modell, das im Wesentlichen aus zwei miteinander verbundenen Funktionsgliedern (*function1*, *function2*) besteht, exportiert werden. Der Ausgang von *function1* stellt den Eingang von *function2* dar, wobei der Ausgang von *function1* über einen Parameter gesteuert werden soll. *Function2* berechnet einen Sinus. Als Ausgangsgrößen sind jeweils die Ausgänge der beiden Funktionsglieder vorgesehen. *Function1.x* wurde lediglich als Eingangsgröße ausgewählt, um eine Warnung zu provozieren.

```
const projectTypeExecutableModel = 1        'prjType
const SaveOutputsAtleastwithdtProt = 2      'outputsApproach
const CVodeSolver = 2                       'prjParam

'Festlegen der Eingaenge, Ausgaenge und Parameter
opts = Array("function1.y", "function2.y")
ipts = Array("function1.x")
params = Array("function1.F")
```

```
'Start von SimulationX
set sim = CreateObject("iti.simx3")

sim.Visible = true
sim.Interactive = true

'Oeffnen des zu exportierenden Modells
set doc = sim.Documents.Open(" C:\Benutzer\Öffentlich\Öffentliche
Dokumente\SimulationX 3.4\Samples\Scripting\Beispiel.ism")
'Exportieren des Modells
msgs = doc.ExportSourceCode(projectTypeExecutableModel,
SaveOutputsAtleastwithdtProt, "COMTest", "C:\CodeExport\ComTest",
CVodeSolver,  Array(ipts(0)), Array(opts(0),opts(1)),
Array(params(0)))

'notwendig für den Fall, dass das Array msgs leer ist
size = -1
On Error Resume Next
      size = UBound(msgs)
On Error GoTo 0

'Ausgeben der messages
if size > -1 then
size = UBound(msgs)
      for i=0 to size
            MsgBox msgs(i)
      Next
end if
```

# 9.3 Co-Simulation

## 9.3.1 Co-Simulation mit SimulationX Koppelelement

### 9.3.1.1 Beschreibung des Kommunikations-Protokolls

Die Co-Simulation von SimulationX nutzt das TCP/IP-Protokoll. SimulationX kann die Kontrolle über den Simulationsprozess übernehmen (Master) oder vom Co-Simulator gesteuert werden (Slave).

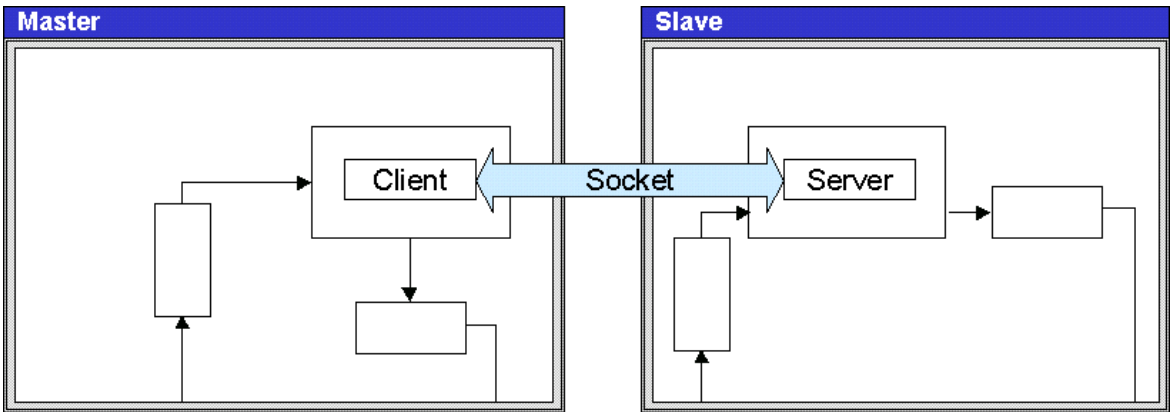
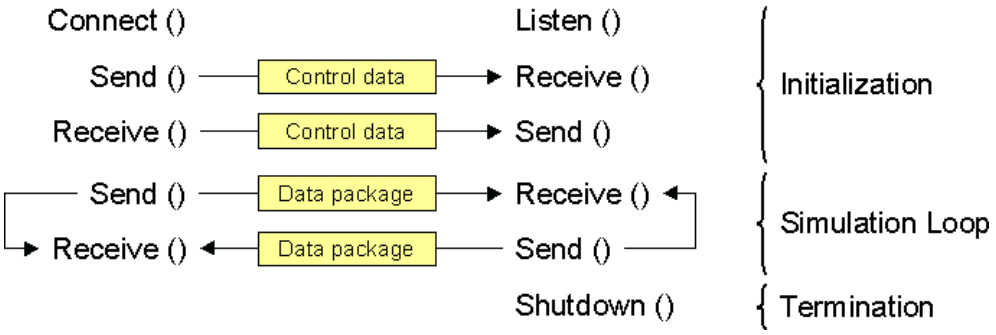



**Bild 9.9:** Kommunikation zwischen den Systemen

Um die Kommunikation so einfach, wie möglich zu gestalten, muss der Empfänger die exakte Länge der ankommenden Pakete kennen.

Es ist nützlich, dass Sender und Empfänger sich zu Beginn der Kommunikation über die Struktur der zu übertragenden Datenpakete verständigen. Um diese Informationen zu übertragen sendet der Client ein Paket fester Größe.

Das Kommunikations-Protokoll zwischen den Partnern wird folgendermaßen beschrieben:




Struktur des Steuerpakets:

| Feld | Typ | Größe [Byte] | Kommentar |
|---|---|---|---|
| Paket Code | Unsigned | 4 | PACKAGE_CONTROL = 0x0011 |
| Kommunikationsschrittweite $dt_{Comm}$ | Double | 8 | |
| Anzahl der Sendekanäle n | Unsigned | 4 | |
| Anzahl der Empfangskanäle m | Unsigned | 4 | |

Der Paket-Code enthält ergänzende Informationen zum Paket. Im gezeigten Prototyp ist nur ein Eintrag zur Unterscheidung von Steuer- und Datenpaket enthalten.
Struktur des Datenpaketes :

| Feld | Typ | Größe [Byte] | Kommentar |
|---|---|---|---|
| Paket Code | Unsigned | 4 | PACKAGE_DATA = 0x0010 |
| Aktuelle Simulationzeit t | Double | 8 | |
| Aktuelle Rechenschrittweite dt | Double | 8 | |
| Anzahl der Kanäle n | Unsigned | 4 | |
| $Wert_t$[Kanal 1] | Double | 8 | |

| | | | |
|---|---|---|---|
| $Wert_{t-dt}$[Kanal 1] | Double | 8 | Kann zusammen mit dt für die Extrapolation genutzt werden |
| … | | | |
| $Wert_t$[Kanal n] | Double | 8 | |
| $Wert_{t-dt}$[Kanal n] | Double | 8 | |

Der Standard bei der Kommunikation über Sockets ist die TCP/IP-Byte-Order für alle Pakete. Jeder Kommunikationspartner muß vor dem Senden seine Daten unter Nutzung der C-Runtime-Funktion `htonl` (host to network) in dieses Format konvertieren. Der Empfänger des Paketes kann die Runtime-Funktion `ntohl` (network to host) für das Rekonvertieren der Daten in das für seine Arbeit notwendige Zahlenformat nutzen. Das Koppel-Element enthält den Parameter zum Umschalten der Byte-Order nur für Testzwecke.

Jeder Simulator kann seine eigene Rechenschrittweite zwischen den Kommunikationsschritten nutzen. Die Werte zwischen den Kommunikationszeitpunkten können mittels der linearen Extrapolation berechnet werden. Deshalb sind in jedem ausgetauschten Paket die aktuellen Werte jeder Variablen und die entsprechende Rechenschrittweite enthalten.

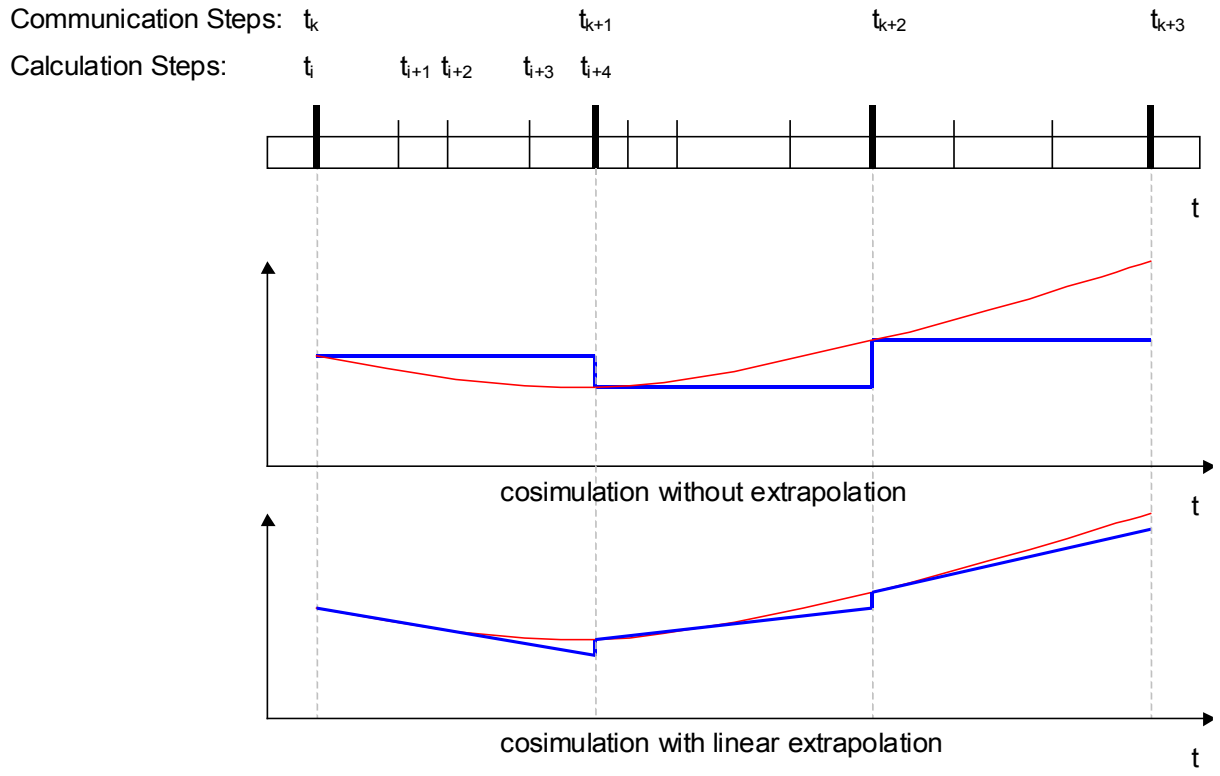



**Bild 9.10:** Kommunikations-Zeitdiagramm

### 9.3.1.2 Nutzung der Koppel-Elemente in SimulationX

Das Koppel-Element hat einen Signaleingang und einen –ausgang.
Wenn mehr als eine Variable zum anderen Simulator übertragen werden sollen sind die Eingangssignale mit einem Multiplexer zu einem Vektor zusammen zufassen und mit einem Demultiplexer ist der Ausgansvektor in Einzelgrößen zu splitten.

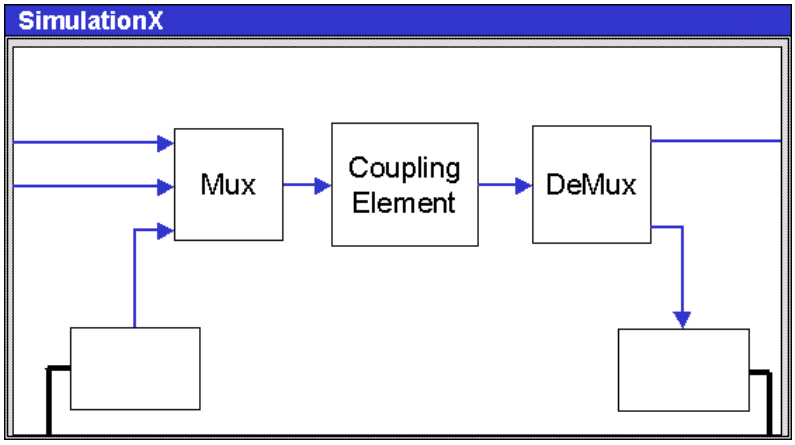



**Bild 9.11:** Koppel-Element

Multiplexer und Demultiplexer sind nicht Bestandteil der Bibliothek. Nutzen sie den TypeDesigner um ein Koppel-Element zu erstellen.

Parameter des Koppel-Elements in SimulationX :

| Parameter | Typ | Kommentar |
|---|---|---|
| Modus | Aufzählung | Client (Standard) oder Server (noch nicht implementiert) |
| IP-Address oder Host Name | String | Nur im Client Modus verfügbar |
| Port Nummer | Integer | |
| Kommunikations-schrittweite | Double | Simulationszeit zwischen zwei Daten-austauschen |
| Extrapolation Mode | Aufzählung | Keine oder lineare Extrapolation |
| Timeout | Double | Maximale Antwortzeit |
| Use TCP/IP Byte Order | Boolean | Nutzung TCP/IP Byte-Übertragung(standard) |
| Dimension of output vector | integer | Anzahl ankommender Kanäle<br>Anmerkung: Die Anzahl der aus-gehenden Kanäle berechnet sich aus der Größe der Eingangssignale x. |

## 9.3.2 Co-Simulation mit Simulink

Die Co-Simulation von SimulationX und Simulink erfolgt über ein Koppel-Element in SimulationX und auf der Simulink-Seite über eine s-Funktion.

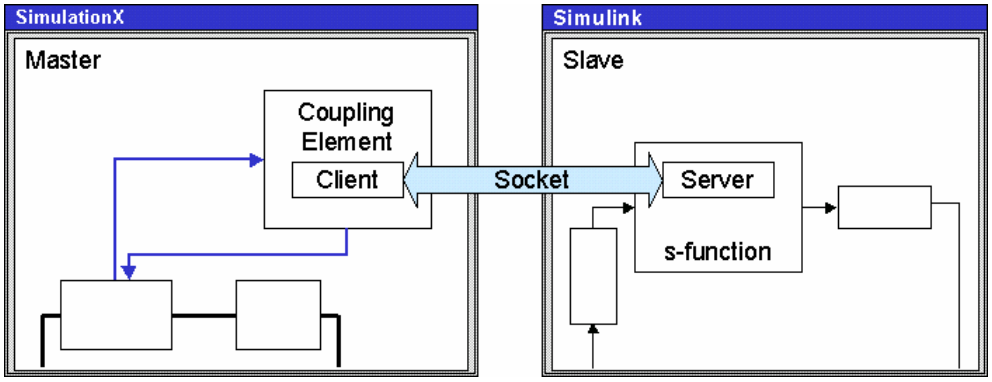



**Bild 9.12:** Kopplung SimulationX mit Simulink

Das Koppel-Element in SimulationX enthält folgende Parameter:

| Parameter | Typ | Kommentar |
|---|---|---|
| Modus | Aufzählung | Client (Standard) oder Server (noch nicht implementiert) |
| IP-Address oder Host Name | String | Nur im Client-Modus verfügbar |
| Port Nummer | Integer | |
| Kommunikationsschrittweite | Double | Simulationszeit zwischen zwei Datenaustauschen |
| Extrapolation Modus | Aufzählung | Keine oder lineare Extrapolation |
| Timeout | Double | Maximale Antwortzeit |
| Use TCP/IP Byte Order | Boolean | Nutzung TCP/IP Byte-Übertragung(Standard) |
| Dimension of output vector | integer | Anzahl ankommender Kanäle<br>Anmerkung: Die Anzahl der ausgehenden Kanäle berechnet sich aus der Größe der Eingangssignale x. |

Die s-Funktion von Simulink hat folgende Parameter:

| Parameter | Typ | Bemerkung |
|---|---|---|
| IP-Address oder Host Name | String | |
| Port Nummer | Integer | |

| | | |
|---|---|---|
| Kommunikationsschritt-weite | Double | Ergibt einen Fehler, wenn diese von der von SimulationX gesendeten abweicht |
| Dimension des input vector | Integer | Anzahl ausgehender Kanäle zu SimulationX |
| Dimension des output vector | Integer | Anzahl eingehender Kanäle von SimulationX – Fehler wenn die Anzahl nicht mit der Anzahl der gesendeten von SimulationX übereinstimmt |

Bevor Sie die Co-Simulation nutzen, sollten Sie sicherstellen, dass die folgenden Werte in SimulationX und Simulink übereinstimmen:

- Simulationszeit
- Kommunikationsschrittweite
- Größe des Ausgangsvektors der s-Funktion und des Eingangsvektors des Koppel-Elements in SimulationX

Um die Co-Simulation zu starten, müssen Sie zuerst Simulink starten. Die s-Funktion wartet auf die Rechnung in SimulationX. Sollte einer der oben genannten Werte nicht übereinstimmen, erfolgt eine Fehlermeldung.

Nutzung eines Beispiels:

- Stellen Sie das Verzeichnis des Beispiels als Arbeitsverzeichnis in MATLAB ein.
- Führen Sie "***start.m***" aus. Dieser Script startet Simulink und lädt das Modell und die Parameter
- Starten Sie SimulationX und laden Sie "drive_cosim.ism"
- Starten der Simulation in Simulink
- Starten von SimulationX

Die Simulationsrechnung läuft jetzt.

## 9.3.3 Simulatorkopplung mit ADAMS

### 9.3.3.1 Einführung

Die Kopplung gestattet die Übertragung skalarer Größen zwischen einem SimulationX- und einem ADAMS-Modell in beide Richtungen. Dazu müssen beim Anwender beide Programme vorhanden sein. SimulationX und ADAMS lösen jeweils die Teilsysteme der Aufgabenstellung. Ein Austausch von nutzerdefinierten Ein- und Ausgangsgrößen erfolgt in festgelegten Zeitschritten.

#### 9.3.3.2 Voraussetzungen und Installation

Um die Kopplung zu ADAMS einsetzen zu können, benötigen Sie das SimulationX Co-Simulation Interface mit der Option *MSC.Adams*. Für ADAMS ist das Release 2005r2 erforderlich.

#### 9.3.3.3 Einrichten der Kopplung in SimulationX und ADAMS

- **SimulationX Einstellungen**

Die Kopplung wird in SimulationX über das Koppelelement (Bibliothek Signalglieder/Simulatorkopplung) realisiert. Als Arbeitsweise ist in jedem Fall die Einstellung *Server* erforderlich.

Die Anzahl der von MSC.ADAMS nach SimulationX zu übertragenden Größen wird mit dem Parameter *nOut* festgelegt. Die Anzahl der von SimulationX nach MSC.ADAMS zu übertragenden Größen richtet sich nach der Elementenanzahl des Eingangsvektors am Signaleingang *x*.

Die Kommunikationsschrittweite (ts) ergibt sich aus den im ADAMS/View – Simulation Control eingestellten Werten für **EndTime** und **Steps** bzw. aus **Stepsize**.

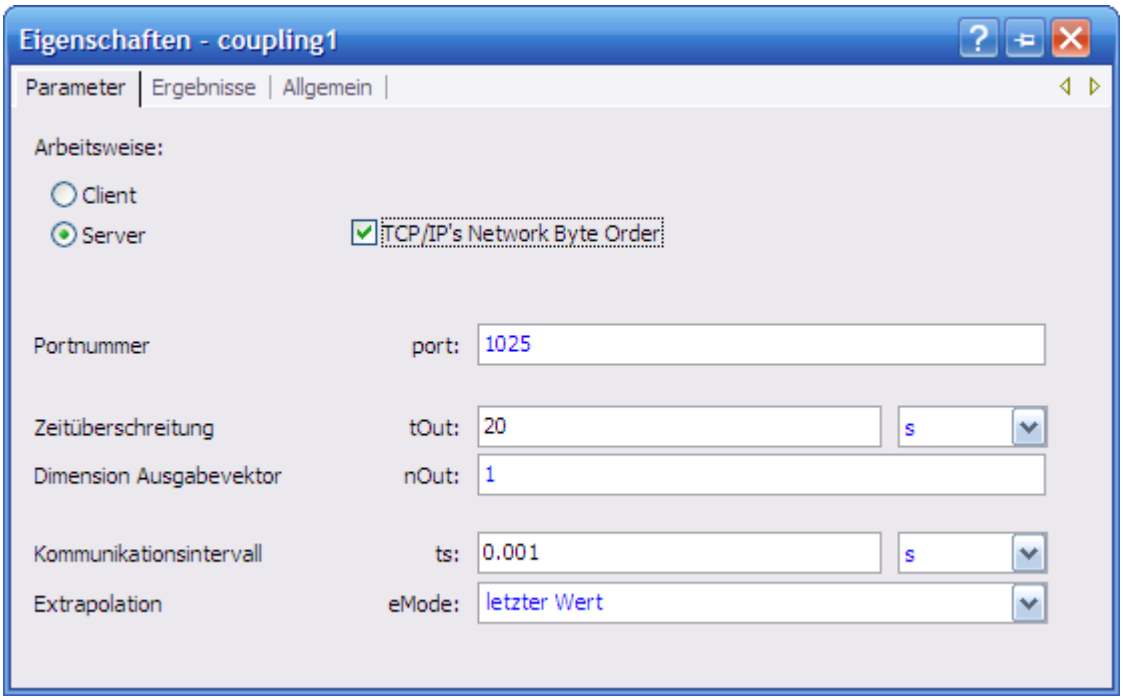



**Bild 9.13:** Eigenschaftsdialog Koppelelement

- **ADAMS/View-Einstellungen**

Zunächst muss das ADAMS-Modell mit der Schnittstelle zu SimulationX verknüpft werden.

Dazu öffnen Sie den Dialog ***Solver Settings*** und geben unter ***Solver Library*** den Pfad zu der benötigten Datei ein.

Es gibt 3 Versionen:

| Datei | nutzbar für |
|---|---|
| simXAdams152.dll | ADAMS 2005r2 |
| simXAdams2007r1.dll | ADAMS 2007r1 |
| simXAdams2011.dll | ADAMS 2011 |

Z.B. die Datei *simxAdams152.dll* in Ihrem SimulationX-Programmverzeichnis.

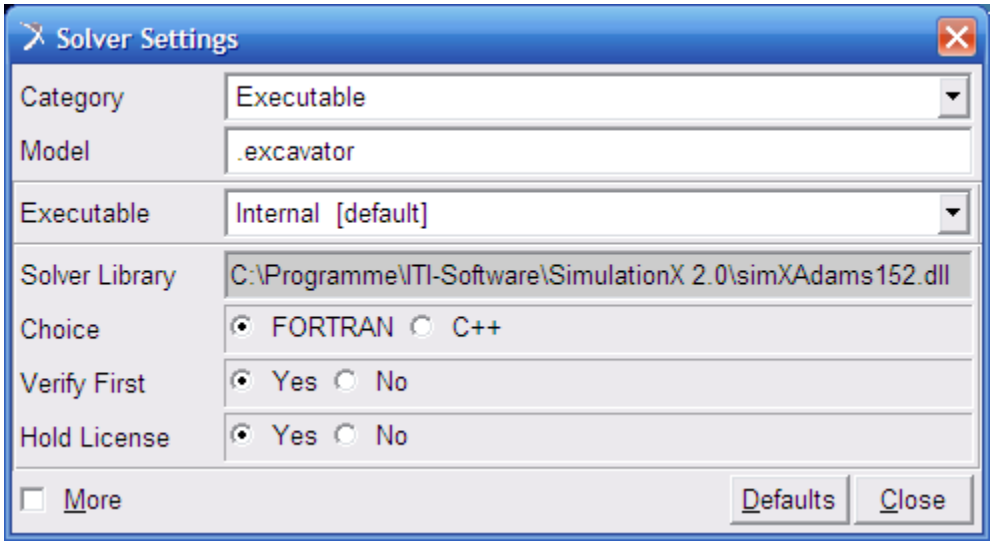



**Bild 9.14:** ADAMS-Solvereinstellungen

Die Kopplung wird in ADAMS über State-Variablen realisiert. Für jede zu übertragende Größe (Ein- und Ausgang) ist eine solche Variable anzulegen.

- ***Ausgänge***

Für jede Größe, die von ADAMS nach SimulationX übertragen werden soll, ist eine ADAMS-State-Variable anzulegen.

Als F(time, ...) kann ein beliebiger ADAMS-Funktionsausdruck angegeben werden.

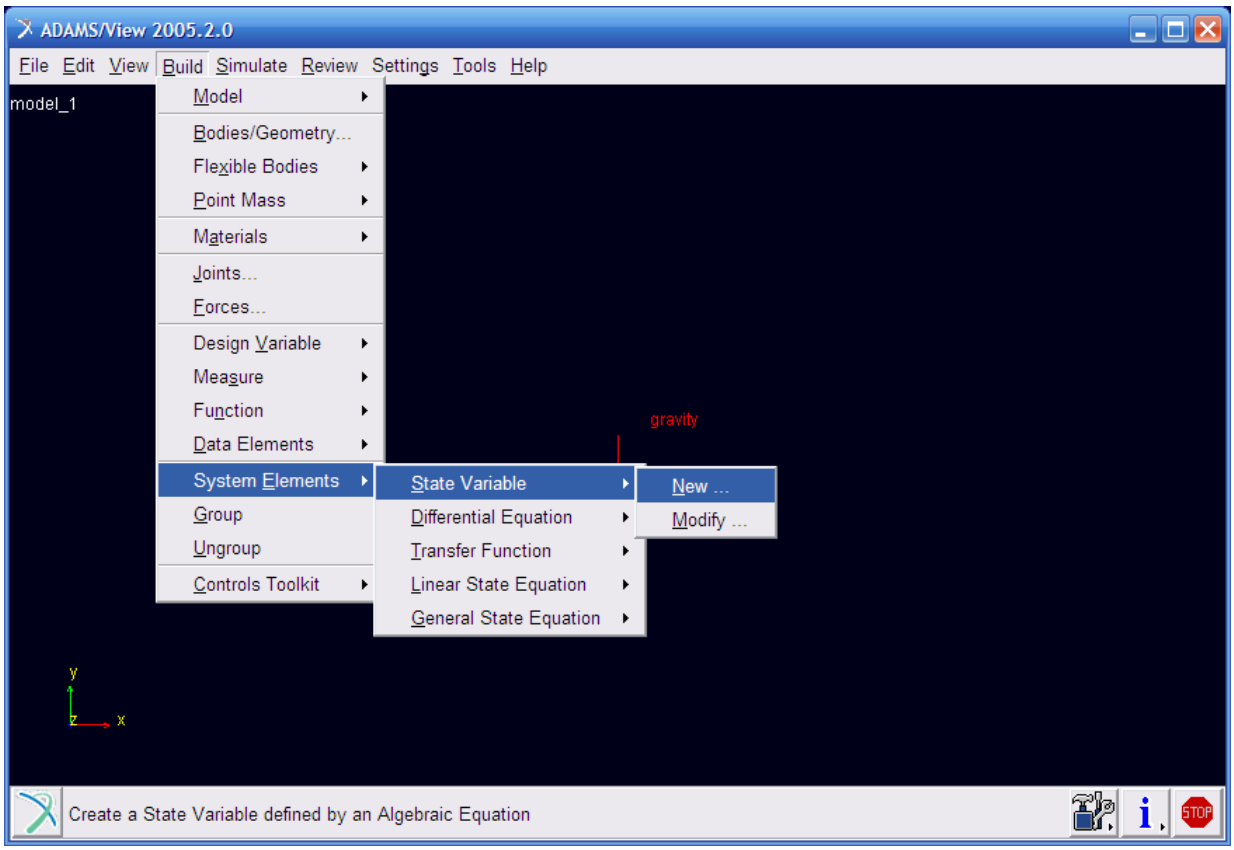



**Bild 9.15:** Anlegen von State-Variablen in ADAMS

**Beispiel:** Die State-Variable Output1 übergibt den Abstand zwischen den Markern .model_1.P1.M1 und .model_1.P2.M1.

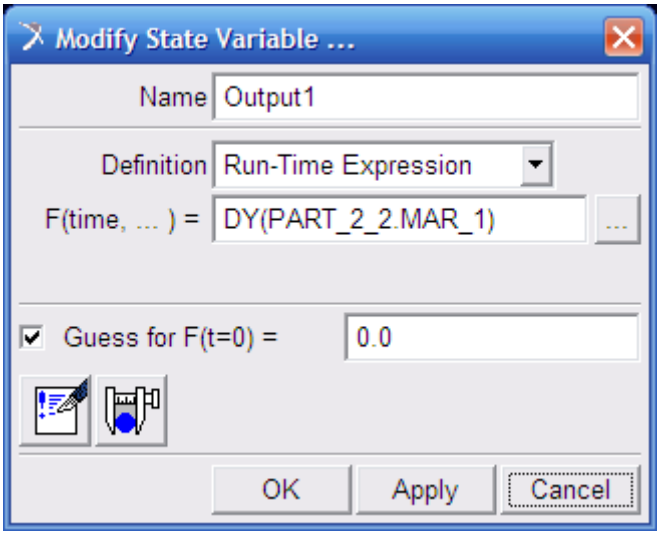



**Bild 9.16:** Beispiel für eine Ausgangsgröße in ADAMS

- ***Eingänge***

Für jede Größe, die von SimulationX nach ADAMS übertragen werden soll, ist ebenfalls eine ADAMS-State-Variable anzulegen (siehe **Bild 9.17**).
Als F(time, ...) muss hier der Aufruf der Funktion *user(<num>)* erfolgen. Als *<num>* ist der 1-basierte Index für den Eingang des SimulationX-Koppelelements anzugeben.

**Beispiel:** Mit user(1) wird auf das erste Element des Eingangsvektors des zugehörigen SimulationX-Koppelelements zugegriffen.

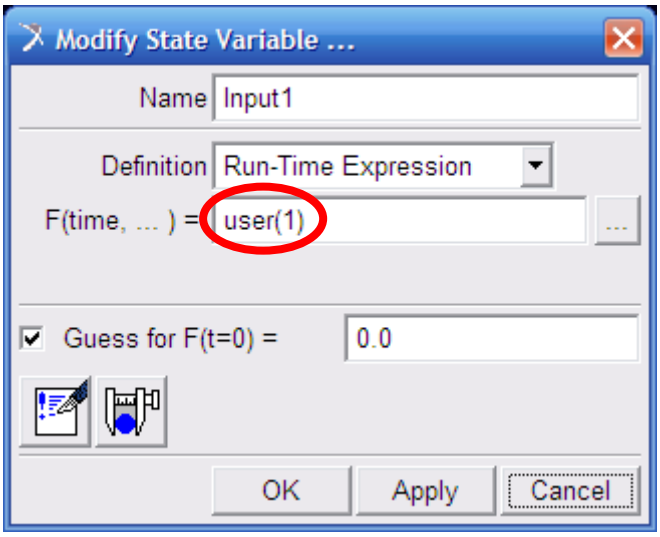



**Bild 9.17:** Beispiel Eingangsgröße in ADAMS

Bei der Übergabe der Größen sind unbedingt die in ADAMS eingestellten Maßeinheiten zu beachten!

Die Zuordnung der Ein- und Ausgänge wird über eine REQUEST-Anweisung realisiert.

Unter *Adams Id* muss als Kennung für das SimulationX/ADAMS – Interface eine *200* eingegeben werden. Unter *User Function* ist eine mit Komma getrennte Liste von Parametern einzugeben, die im Einzelnen folgende Bedeutung haben:

Parameter 1: Anzahl der von SimulationX nach MS.ADAMS zu übertragenden Größen

Parameter 2: ADAMS ID der ersten von ADAMS nach SimulationX zu übertragenden Größe

…

Parameter n: ADAMS ID der letzten von ADAMS nach SimulationX zu übertragenden Größe

Die Anzahl der von ADAMS nach SimulationX zu übertragenden Größen ergibt sich aus der Anzahl der Listeneinträge (n-1).

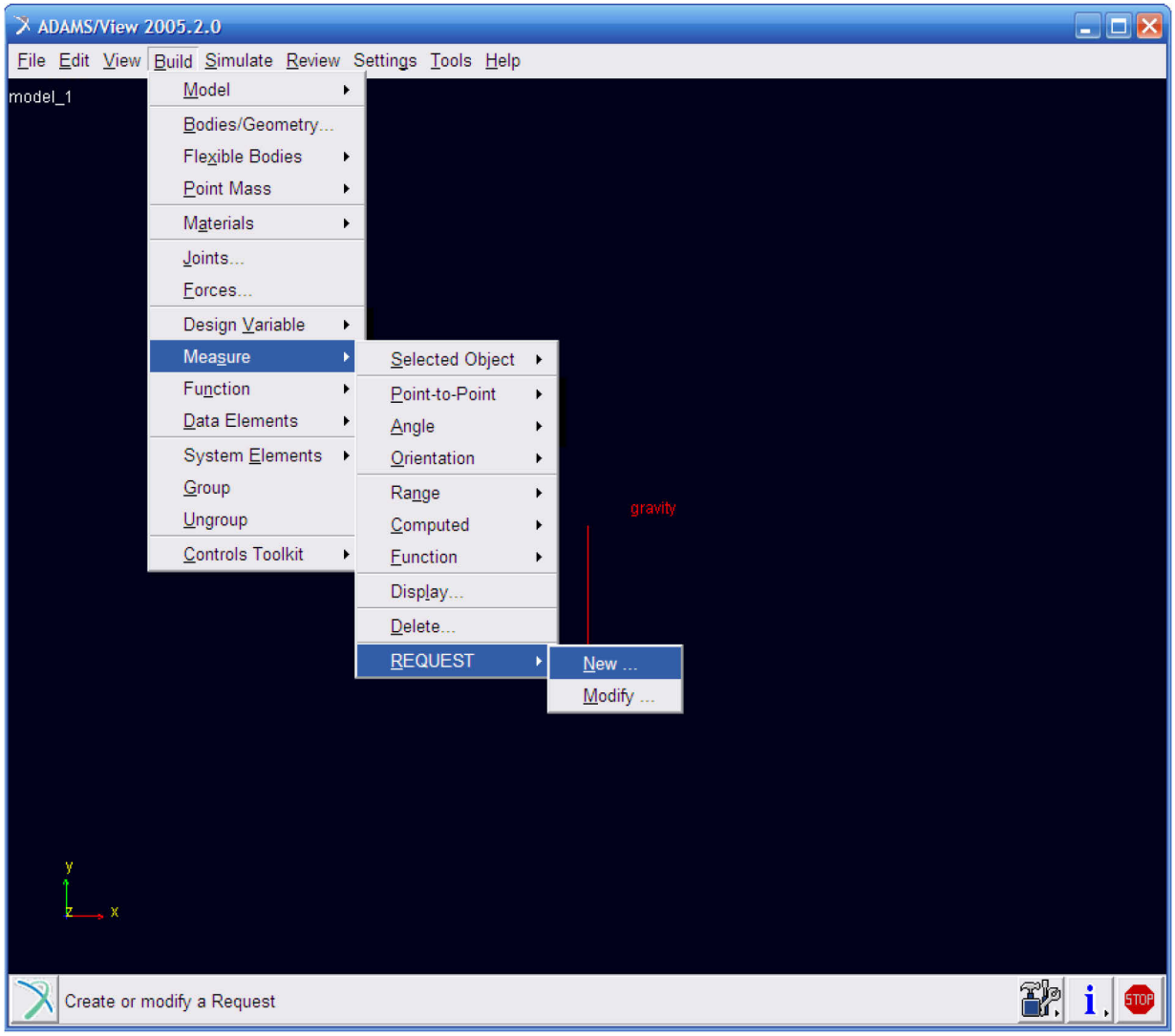



**Bild 9.18:** Beispiel für 3 Eingänge und 6 Ausgänge (ADAMS ID 4…9)

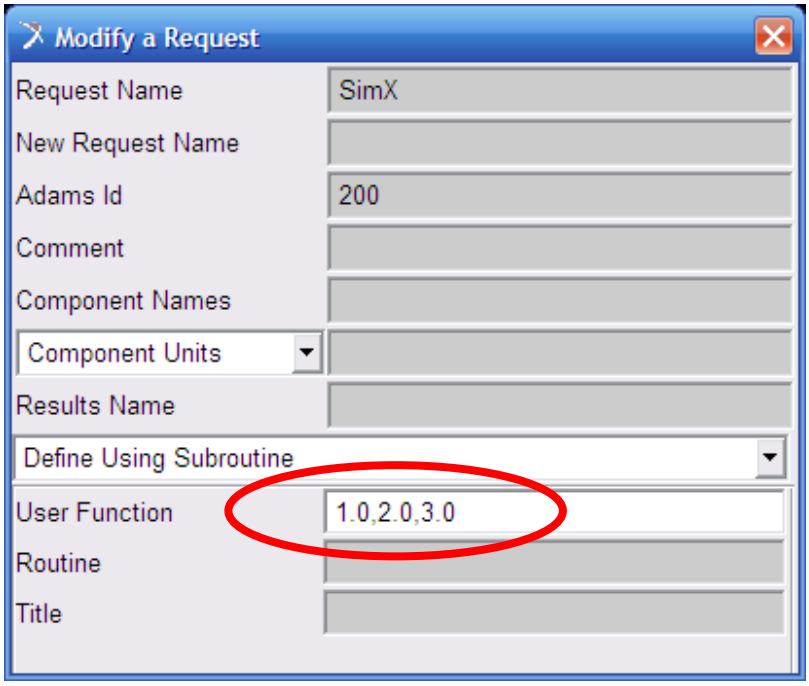



**Bild 9.19:** Beispiel für einen Eingang und zwei Ausgänge (ADAMS ID 2 und 3)

- **Co-Simulation ausführen**

Die Steuerung der Simulationsrechnung erfolgt über das Interactive Simulation Control von ADAMS/View. Vor Start der Berechnung ist sicherzustellen, dass die im Koppelelement von SimulationX eingestellte Kommunikationsschrittweite *ts* mit der Schrittweite übereinstimmt, die sich aus den Einstellungen für **EndTime** und **Steps** bzw. **Stepsize** im ADAMS/View Simulation Control Dialogfeld ergeben.

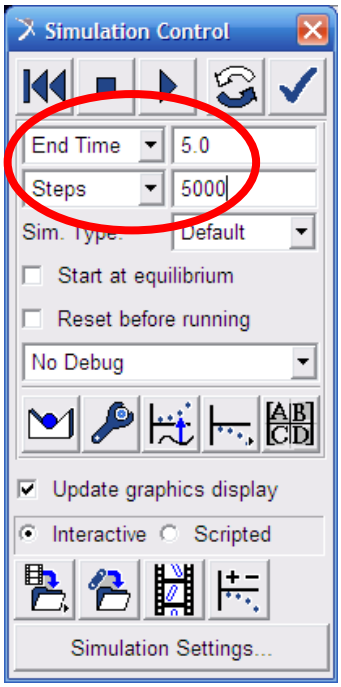



**Bild 9.20:** Simulation Control von ADAMS/View

Beim ersten Start der Berechnung wird ein Dialog geöffnet. Hier kann eingestellt werden, wie die Verbindung zu SimulationX hergestellt werden soll. Folgende Möglichkeiten stehen dabei zur Verfügung:

1. Die Verbindung erfolgt mit dem aktiven Modell von SimulationX auf dem lokalen Rechner. Dazu muss SimulationX zuvor gestartet und das jeweilige Modell geladen werden. Nach dem Schließen des Dialogfeldes mit OK wird die Rechnung gestartet. SimulationX wird dabei ferngesteuert.
2. Die Verbindung erfolgt mit dem aktiven Modell von SimulationX auf einem entfernten Rechner im Netzwerk. Dazu ist die Eingabe von IP-Adresse bzw. Rechnernamen sowie der Portnummer erforderlich. Die Portnummer muss mit der im Koppelelement des SimulationX-Modells eingestellten Portnummer übereinstimmen. Nach dem Schließen des Dialogfeldes erfolgt die Aufforderung, SimulationX auf dem entfernten Rechner zu starten. Das Quittieren der Nachricht mit OK startet dann die Berechnung für ADAMS und damit des gekoppelten Gesamtsystems.
3. Es wird das SimulationX-Modell ausgewählt, zu dem die Verbindung hergestellt werden soll. Nach dem Schließen des Dialogfeldes mit OK wird

SimulationX gestartet, das betreffende Modell geladen und die Rechnung gestartet.

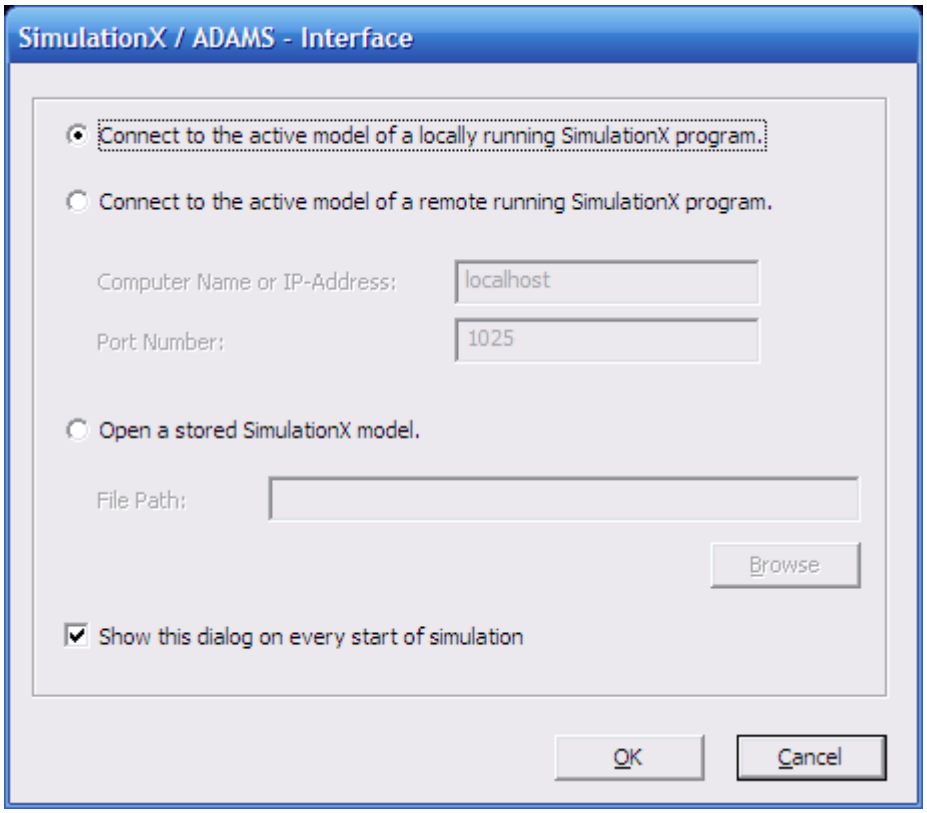



**Bild 9.21:** Einstellungsdialog für das SimulationX/ADAMS-Interface

Des Weiteren können Sie einstellen, ob die Anzeige dieses Dialogfelds bei jedem Start der Berechnung erfolgen soll oder nicht.

- **Anwendungsbeispiel: Hydraulischer Klappenantrieb**

Ein einfaches Modell für eine hydraulisch angetrieben Klappe soll die Arbeitsweise des Co-Simulation-Interfaces zu MSC.ADAMS demonstrieren.

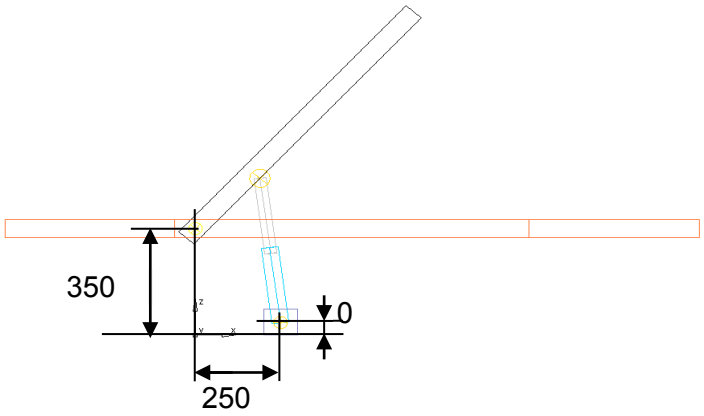



**Bild 9.22:** Funktionsskizze für Klappenantrieb

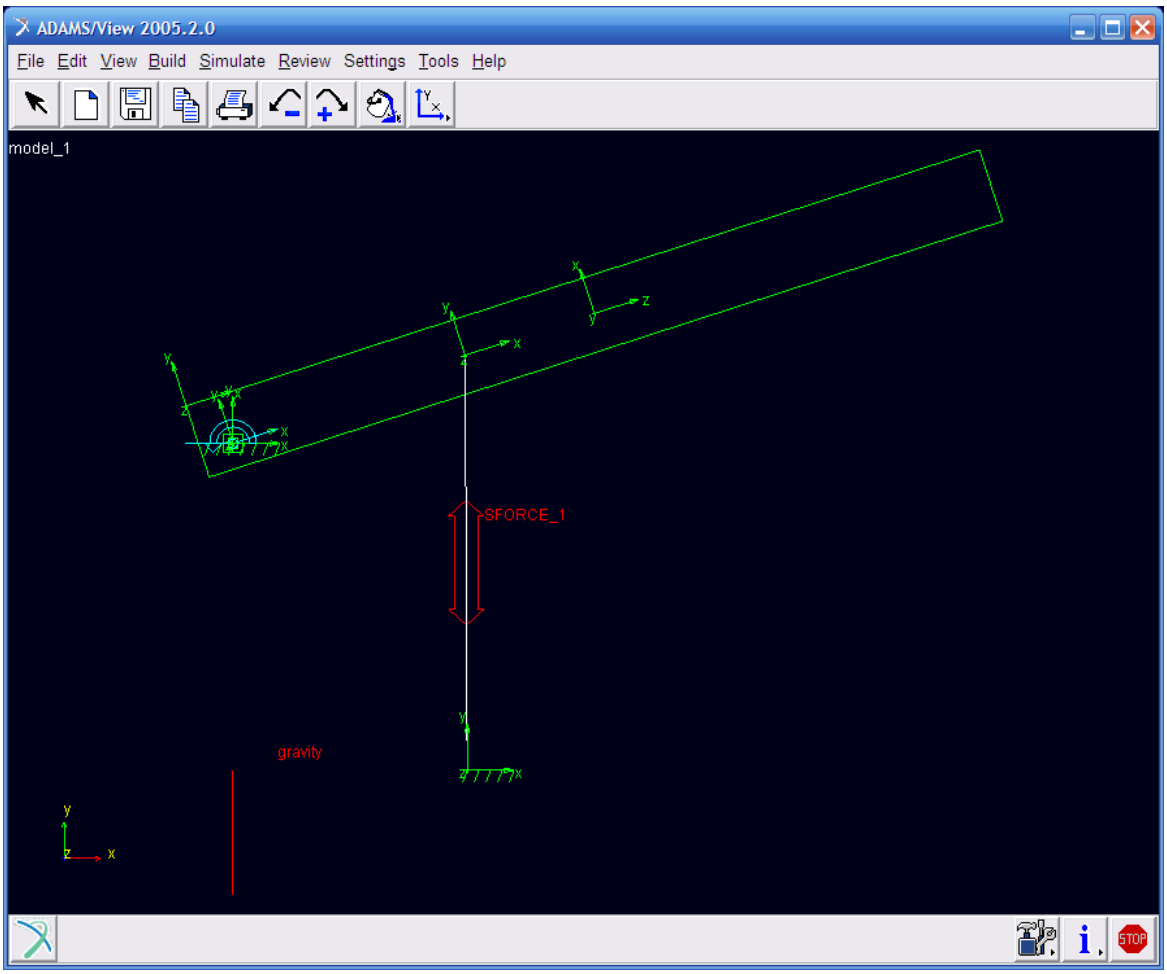



**Bild 9.23:** Modell der Klappe in ADAMS/View

Die Modellierung des mechanischen Teils erfolgt in ADAMS/View (**Bild 9.23**). Die Einleitung der Zylinderkraft wird über ein SFORCE-Element realisiert. Für die Messung von Zylinderauslenkung und Kolbengeschwindigkeit werden die Funktionen DM() bzw. VR() verwendet.
Für die Kopplung mit dem SimulationX-Modell werden folgende State-Variablen sowie der entsprechende REQUEST-Block vereinbart:

| Koppelgröße | SimulationX Ident | ADAMS State Variable |
|---|---|---|
| Zylinderkraft | coupling1.x[1] | .model_1.InF |
| Auslenkung des Zylinders | coupling1.y[1] | .model_1.OutDispl |
| Kolbengeschwindigkeit | coupling1.y[2] | .model_1.OutVelo |

```
Object Name      :  .model_1.InF
Object Type      :  ADAMS_Variable
Parent Type      :  Model
Adams ID         :  5
Active           :  NO_OPINION
Initial Condition    : 0.0
Function         : user(1)
Object Name      :  .model_1.OutDispl
```

```
Object Type      :  ADAMS_Variable
Parent Type      :  Model
Adams ID         :  2
Active           :  NO_OPINION
Initial Condition    : 0.0
Function         : DM(MARKER_13,MARKER_12)

Object Name      :  .model_1.OutVelo
Object Type      :  ADAMS_Variable
Parent Type      :  Model
Adams ID         :  3
Active           :  NO_OPINION
Initial Condition    : 0.0
Function         : VR(MARKER_13, MARKER_12)

Object Name      :  .model_1.SimX
Object Type      :  Request
Parent Type      :  Model
Adams ID         :  200
Active           :  NO_OPINION
Comment              : None
Title                : None
Functions        :
User Function    : 1.0, 2.0, 3.0
```

Das zugehörige SimulationX-Modell besteht aus dem hydraulischen Antrieb mit Wegvorgabe und Regler, dem Koppelelement zum Datenaustausch mit dem ADAMS-Modell sowie dem physikalischen Interface.

Das physikalische Interface beruht auf einer Geschwindigkeitsvorgabe sowie einer geschwindigkeitsabhängigen Dämpfung. Diese kompensiert den bei der Kopplung mit äquidistanter Schrittweite auftretenden Fehler und verhindert so ein Aufschwingen des Modells in Ruhelage (v=0).

| | | |
|---|---|---|
| Dateien: | FlapCosimAdams.ism | (SimulationX-Modell) |
| | FlapCosimSimX.bin | (ADAMS/View Database) |
| Simulationszeit: | 10 s | |
| Kommunikationsschrittweite: | 0.001 s | |

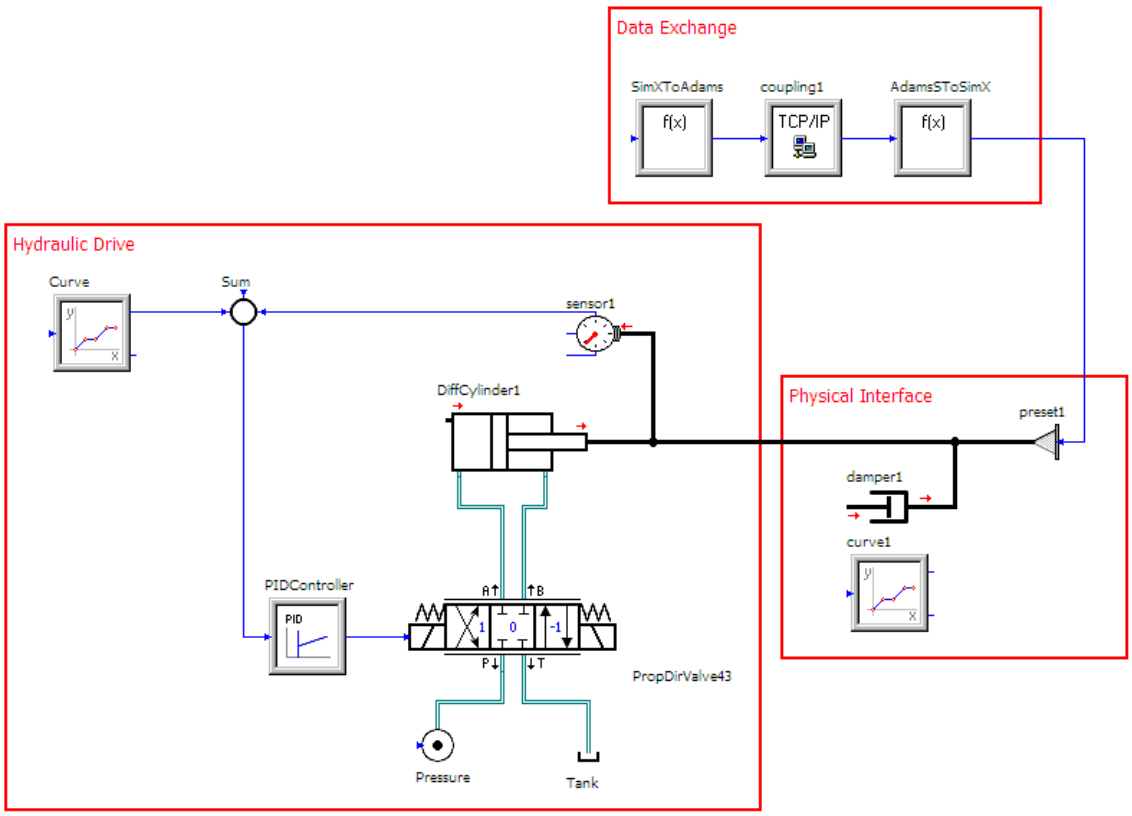



**Bild 9.24:** SimulationX-Modell für Klappenantrieb

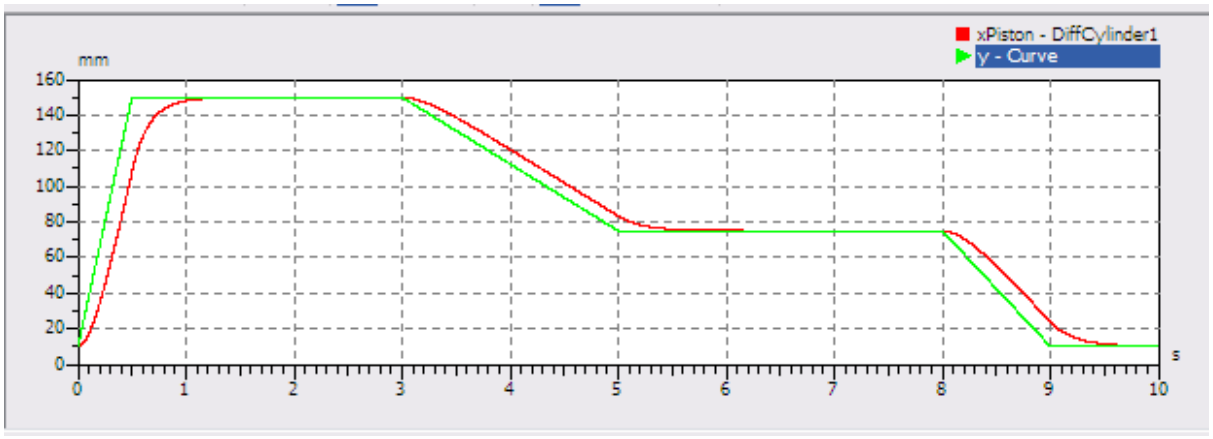



**Bild 9.25:** Ergebnisse der Co-Simulation

## 9.3.4 Co-Simulation mit CarSim, BikeSim, TruckSim

### 9.3.4.1 Einführung

VehicleSim ist ein Toolset für die effiziente Simulation von PKW, Lastkraftwagen und Motorrädern. Das SimulationX-Modell wird durch einen speziellen Co-Simulationsblock (enthalten im Package *Co-Simulation/CarSim, BikeSim TruckSim*) mit dem VehicleSim-Modell gekoppelt. Dieser Block nutzt die VehicleSim API (Application Programming Interface) [7] für den Datenaustausch und die Synchronisation zwischen SimulationX und VehicleSim.

### 9.3.4.2 Workflow

Für eine Co-Simulation zwischen SimulationX und VehicleSim sind die folgenden Schritte notwendig:

- Vorbereitung des VehicleSim-Modells
- Vorbereitung des SimulationX-Modells
- Simulation in SimulationX und VehicleSim
- Ergebnisauswertung in VehicleSim und SimulationX

Die Co-Simulation wurde mit CarSim getestet. Daher verwenden wir in der folgenden Beschreibung den Begriff CarSim. Mit BikeSim und TruckSim funktioniert die Co-Simulation in gleicher Weise.

### 9.3.4.3 Beispielmodell

Dieser Abschnitt zeigt, wie ein SimulationX Modell eines Motors mit einem Antriebsstrang in einem CarSim-Modell gekoppelt wird. Das SimulationX-Modell ist in der folgenden Abbildung dargestellt.

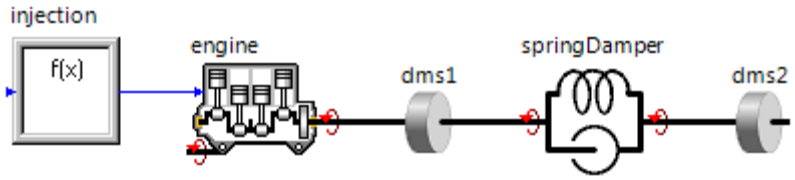



**Bild 9.26:** SimulationX-Modell eines Motors mit einem sehr vereinfachten Zweimassenschwungrad

Alle Motor-Parameter werden auf ihren Standardwerten belassen. Die geänderten Parameter der anderen Modell-Elemente sind in der folgenden Tabelle dargestellt.

| Name | Value | Unit |
|---|---|---|
| `dms1.J` | 0.02 | kgm² |
| `dms1.om` | 800 (fixed) | rpm |
| `dms2.J` | 0.02 | kgm² |
| `dms2.om0` | 800 (fixed) | rpm |
| `springDamper.kind` | Spring-Damper | |
| `springDamper.k` | 10000 | Nm/rad |
| `springDamper.b` | 10 | Nms/rad |
| `injection.F` | 1 | |

### 9.3.4.4 Vorbereitung des CarSim-Modells

Zuerst erfolgt die Vorbereitung des CarSim-Modells. Dazu ist die Antriebsstrang-Konfiguration zu ändern, so dass der vorhandene Motor mit dem von SimulationX ersetzt wird.

Öffnen Sie die Antriebsstrang-Ansicht in CarSim und stellen auf "***External Engine Model***", wie in **Bild 9.27** gezeigt.

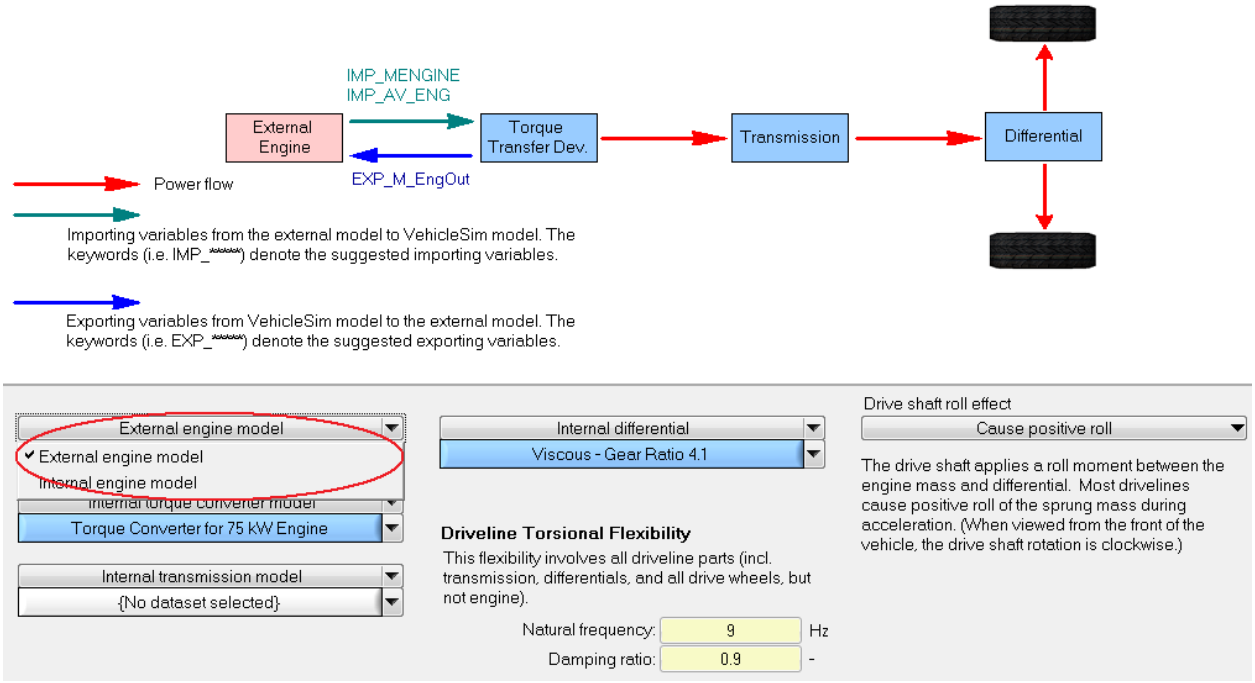



**Bild 9.27:** Wechsel zu External Engine Model

Das Blockdiagramm ändert sich, wie oben zu sehen. Um das interne Motormodell zu ersetzen muss die Motordrehzahl von SimulationX berechnet und CarSim als `IMP_AV_ENG` bereitgestellt werden. Das Drehmoment (`EXP_M_EngOut`) als Reaktion des Antriebes ist an den Antriebstrang in SimulationX zurückzugeben.

Zur Vorbereitung des Datenaustausches ist die CarSim-Variable `IMP_AV_ENG` von SimulationX nach CarSim zu importieren. Dazu wechseln Sie zur CarSim Run Control Seite und betätigen die Schaltfläche ***Models,*** um ***Models: Self-Contained Solvers*** auszuwählen.

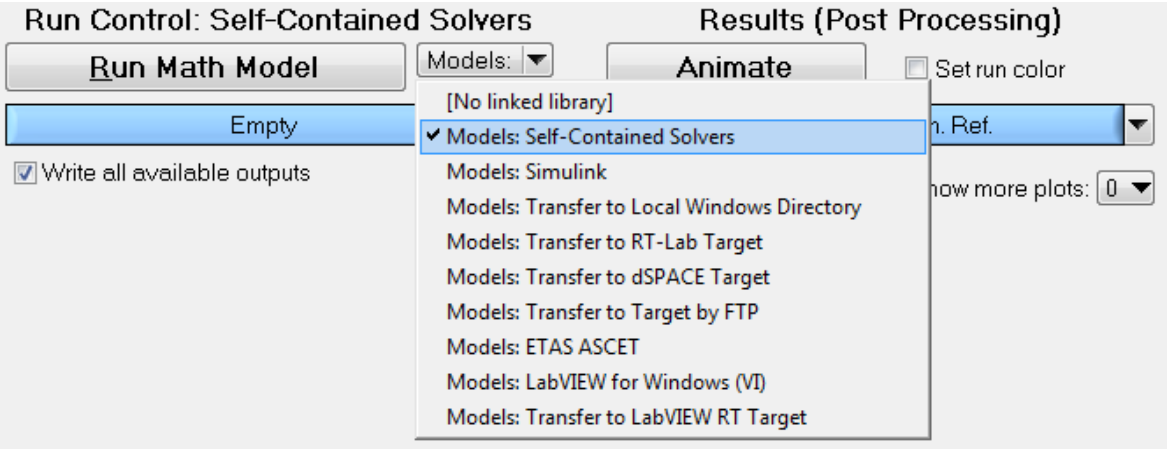



**Bild 9.28:** Umschalten auf Self-Contained Solvers

Erzeugen Sie einen neuen Datensatz mit der Bezeichnung “SimulationX” und öffnen Sie diesen.

In dieser Ansicht wechseln Sie unter ***Misc.Links*** zum Eintrag ***I/O Channels*** und setzen diesen auf ***I/O Channels: Import***, wie in der folgenden Abbildung gezeigt:

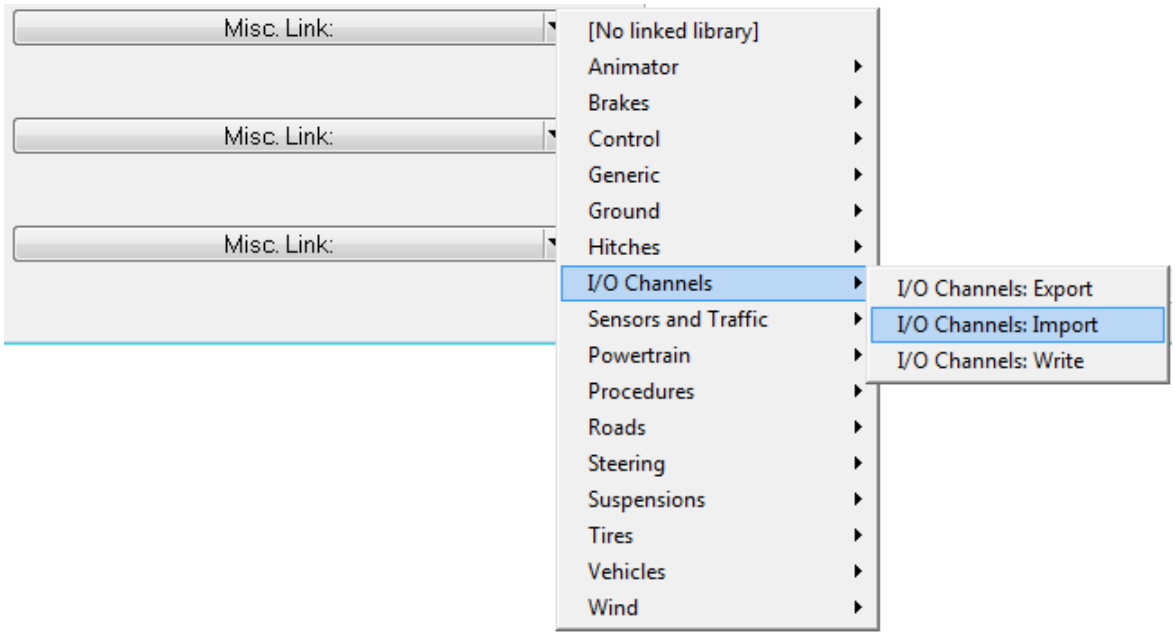



**Bild 9.29:** Vorbereitung der Import-Variable

Erzeugen Sie einen neuen Datensatz und fügen die Variable `IMP_AV_ENG` zur Liste der aktiven Importvariablen hinzu.

Wählen Sie unter ***Mode*** den Eintrag ***Replace*** aus (siehe **Bild 9.30)**. Ist die ***Mode***-Auswahl deaktiviert, so öffnen Sie das zugehörige Parsfile und führen die Änderungen dort durch.

Wie diese Variable mit SimulationX-Ausgängen und CarSim-Ausgänge mit SimulationX-Eingängen verbunden werden, ist im nächsten Abschnitt beschrieben.

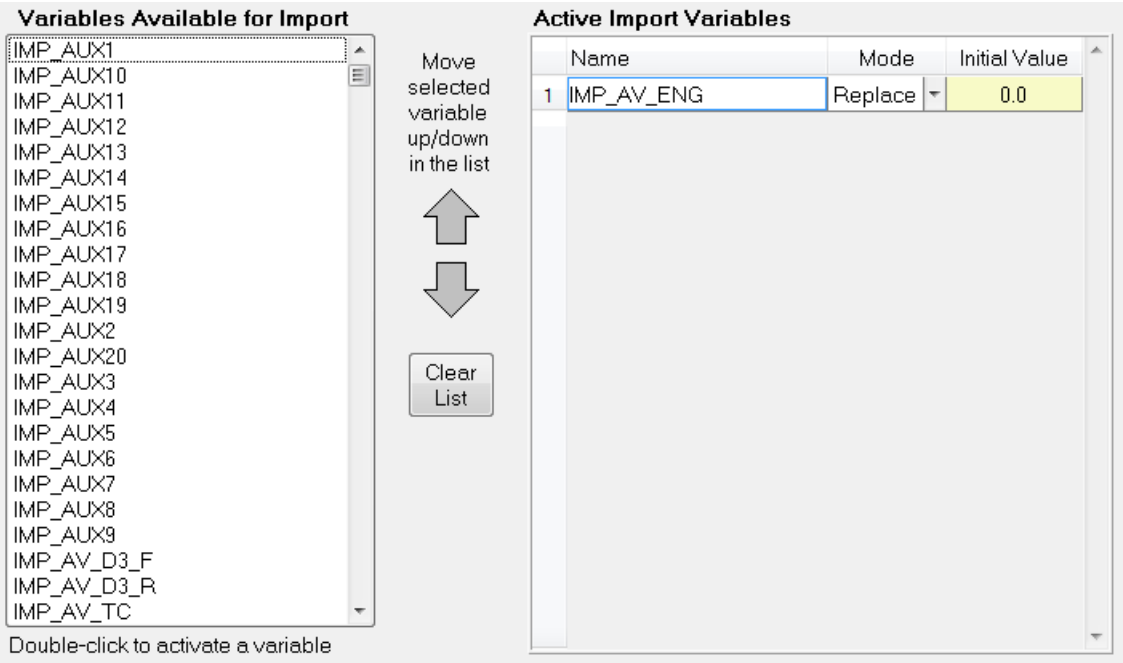



**Bild 9.30:** Hinzufügen der Import-Variable

### 9.3.4.5 Vorbereitung des SimulationX-Modells

Für die Co-Simulation muss das SimulationX-Modell ein wenig modifiziert werden. Um das Drehmoment `EXP_M_EngOut` von CarSim an SimulationX zurückzugeben ist ein Element **Äußeres Moment** (Mechanics.Rotation.Source) mit dem Namen loadTorque hinzuzufügen, wie in **Bild 9.31** dargestellt. Der Parameter `T` behält die Standardeinstellung "in1". Auf diese Weise wird der durch den Signaleingang "in1" eingebrachte Wert als Drehmoment in den Antriebsstranges eingebracht. Wenn Sie die anderen Variablen mit CarSim verbinden wollen, achten Sie auf die Vorzeichen.

Die Drehgeschwindigkeit des Antriebsstranges wird über die Variable `IMP_AV_ENG` an CarSim übergeben. Dafür ergänzen wir das Modell um einen Drehzahlsensor (Mechanics.Rotation.Sensor).

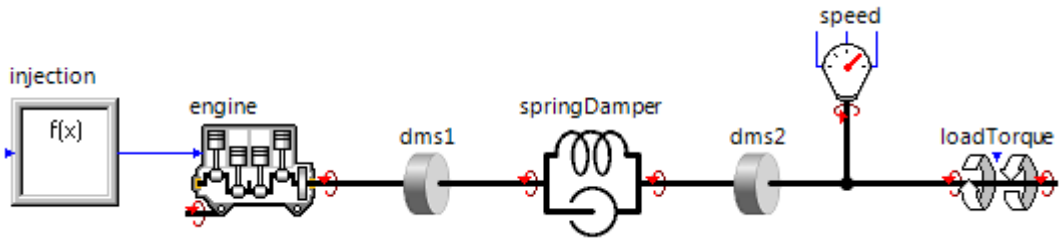



**Bild 9.31:** Motormodell mit Lastmoment und Drehzahlsensor

Jetzt wird der VehicleSim Co-Simulationsblock hinzugefügt. Dieser ist im Package CoSimulation.VehicleSim enthalten.

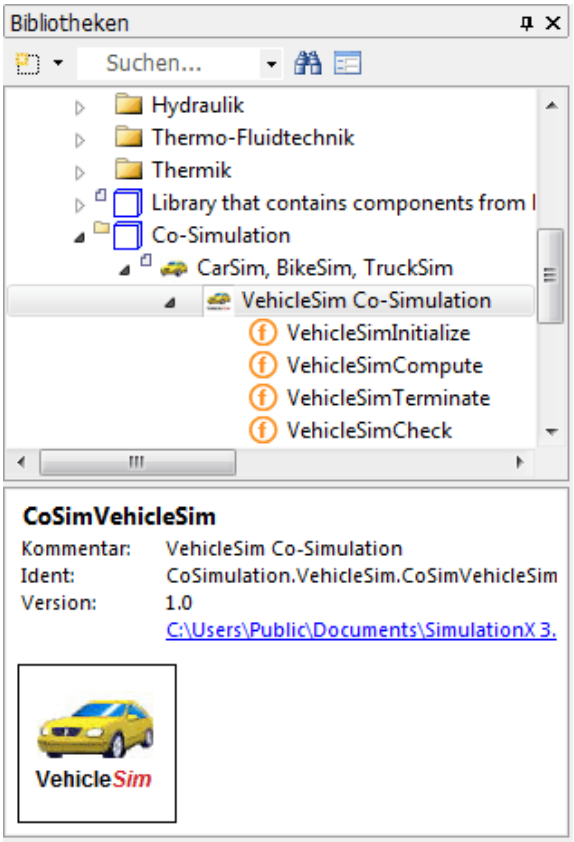



**Bild 9.32:** SimulationX VehicleSim Co-Simulation block

Der Co-Simulationsblock hat die folgenden Eigenschaften:

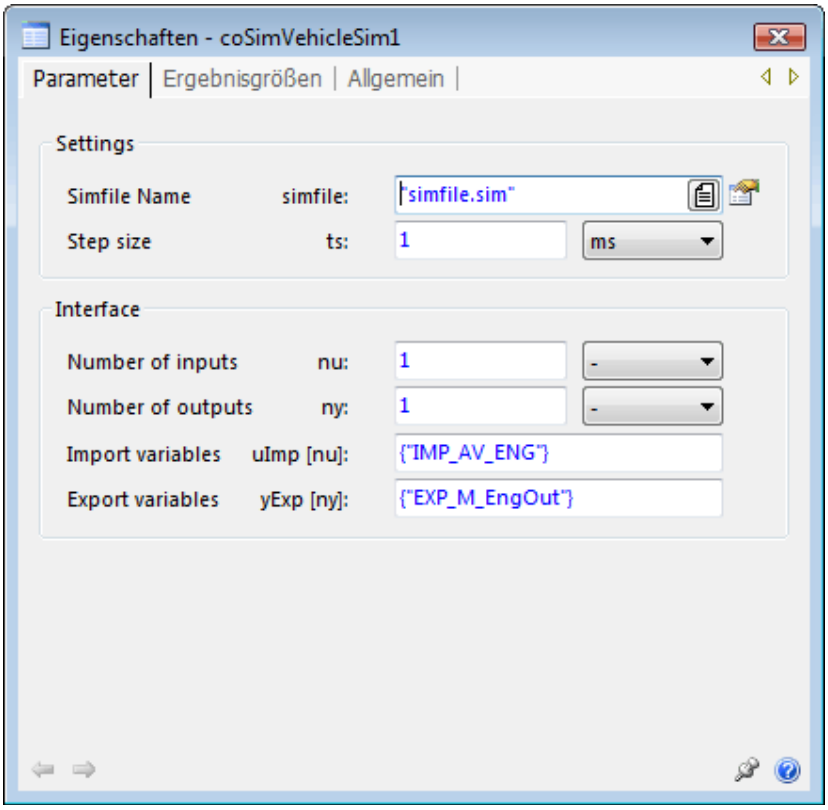



**Bild 9.33:** Eigenschaften des VehicleSim Co-Simulation block

- **Simfile Name (`simfile`)**

Für die Kommunikation mit CarSim über dessen API-Schnittstelle muss das verwendete simfile.sim angegeben werden. Diese Datei enthält Links zu anderen Daten, die von CarSim genutzt werden. Die Datei befindet sich im aktuellen CarSim Datenbankverzeichnis. Dieses Verzeichnis wird während des Starts von CarSim über die folgende Dialogbox eingestellt.

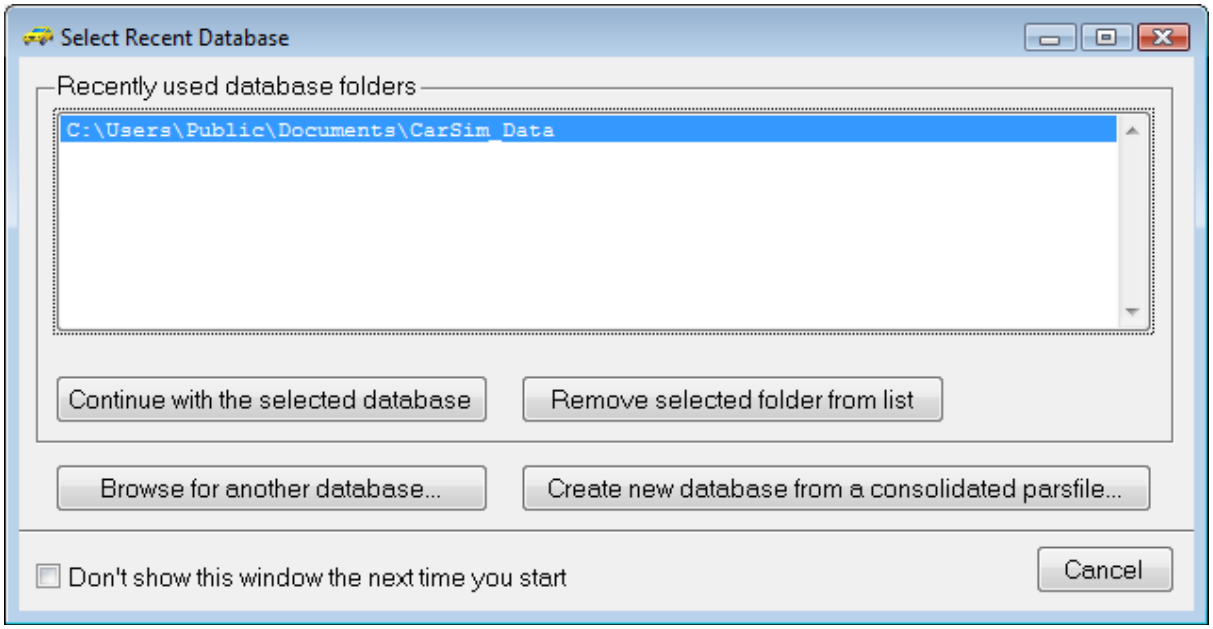



**Bild 9.34:** Datenauswahl während des CarSim Starts

Ein Datei-Auswahldialog öffnet sich, wenn die Schaltfläche  am Ende des Eingabefeldes des Parameters simfile im SimulationX Eigenschaftsfenster des VehicleSim Co-Simulation Blockes betätigt wird. Gehen Sie zum aktuellen CarSim Datenbankverzeichnis und wählen das aktuelle CarSim simfile.sim.

- **Schrittweite `ts`**

Die Schrittweite wird für die Kommunikation mit CarSim und die Aktualisierung der Ein- und Ausgänge genutzt. Die CarSim-Integrationsschrittweite muss den gleichen Wert haben, ist das nicht der Fall, erscheint eine Fehlermeldung und die Simulation startet nicht.
Wie in [7] Kapitel "4. Extending Math Models Using Import and Export Arrays " erwähnt, entsteht durch den Import und Export von CarSim Variablen eine Zeitverzögerung von einem Schritt zwischen CarSim und SimulationX. Dies kann zu numerischen Instabilitäten für Closed-Loop Modelle mit einer engen Kopplung führen. Reduzieren Sie in solchen Fällen die Schrittweite `ts`.

Für das Beispiel wird eine Schrittweite von 1 ms genutzt.

- **Anzahl Eingänge `nu`, Anzahl Ausgänge `ny`**

Der Co-Simulationsblock kommuniziert über den Signaleingang `u` und den Signalausgang `y` mit dem umgebenden SimulationX-Modell.  Diese Signalanschlüsse

können Vektoren sein. Die Dimensionen dieser Vektoren werden mit `nu` (für Eingänge `u`) bzw. `ny` (für Ausgänge `y`) definiert. Für die Verbindung von skalaren Signalanschlüssen mit diesen Vektor-Anschlüssen können die im Package CoSimulation.MuxDemux vorhandenen Multiplexer und Demultiplexer genutzt werden (siehe **Bild 9.36**).

In unserem Beispiel wird ein Signal (die Drehgeschwindigkeit des Antriebstranges) nach CarSim übergeben. Deshalb wird `nu=1` gesetzt. Die Reaktion des CarSim-Models in Form des Drehmomentes wird an das Antriebsstrangmodell zurückgegeben. Außerdem möchten wir die CarSim Fahrzeuggeschwindigkeit in SimulationX beobachten. Deshalb wird `ny` = 2 gesetzt.

- **Import Variable `uImp[nu]`, Export Variables `yExp[ny]`**

Diese Parameter sind Vektoren von Zeichenketten. Die Dimension sind `nu` bzw. `ny`. Hier sind die Namen der CarSim-Variablen anzugeben, die mit den Co-Simulationsblock Eingängen und Ausgängen verbunden werden. Wird einer der Namen in der CarSim-Datenbank nicht gefunden, wird während des Starts der Simulation eine Warnung ausgegeben.
Die Antriebsstrang-Geschwindigkeit wird als `IMP_AV_ENG` nach CarSim importiert. Deshalb erfolgt die Eingabe von {"IMP_AV_ENG"} für `uImp` (siehe **Bild 9.35**). Zusätzlich zum Drehmoment als Reaktion des CarSim-Models möchten wir die Fahrzeuggeschwindigkeit (`VX`) beobachten. Dazu wird für {"EXP_M_EngOut","VX"} bei `yExp` eingegeben.

Damit sollte der Co-Simulationsblock folgende Parameter besitzen:

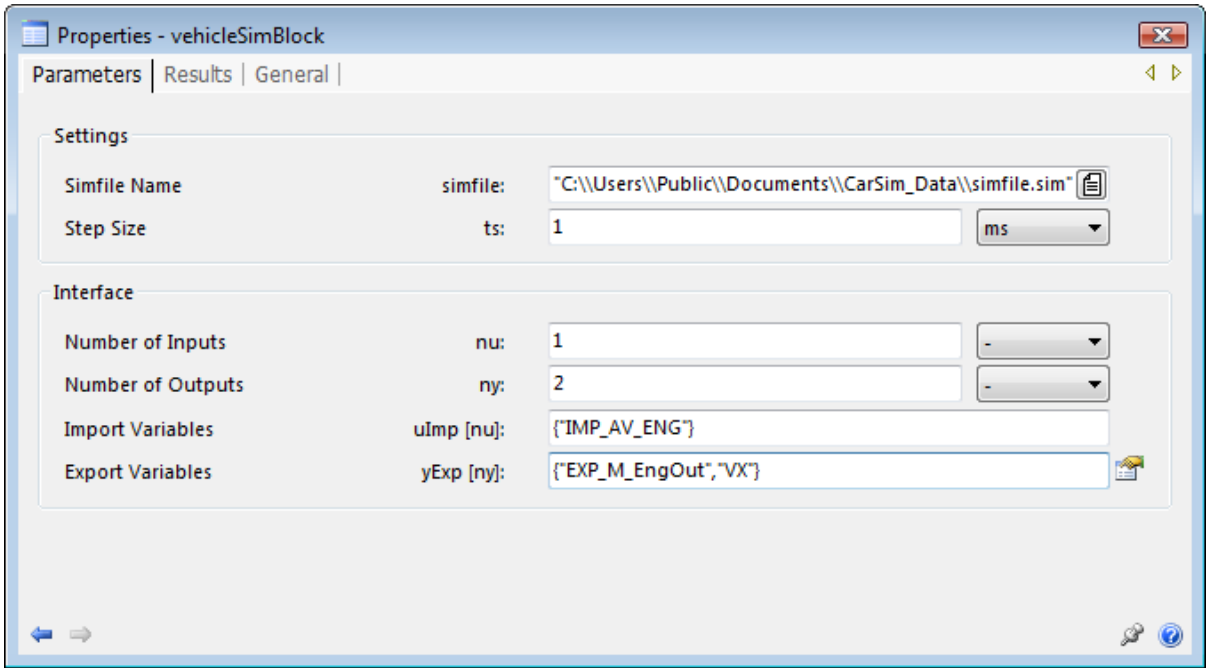



**Bild 9.35:** Parameter des Co-Simulationsblocks

Als letzten Schritt ist den Co-Simulationsblock mit dem Antriebsstrangmodell zu verbinden. Wir verbinden den Eingang `u` mit dem Ausgang des Geschwindigkeitssensor (`speed.om`). Der Ausgang hat die Dimension `ny`=2. Deshalb wird ein Demultiplexer (MuxDemux.Demultiplex2) verwendet, um diesen Vektor in Skalare aufzuteilen.

Das fertige Modell zeigt **Bild 9.36**.

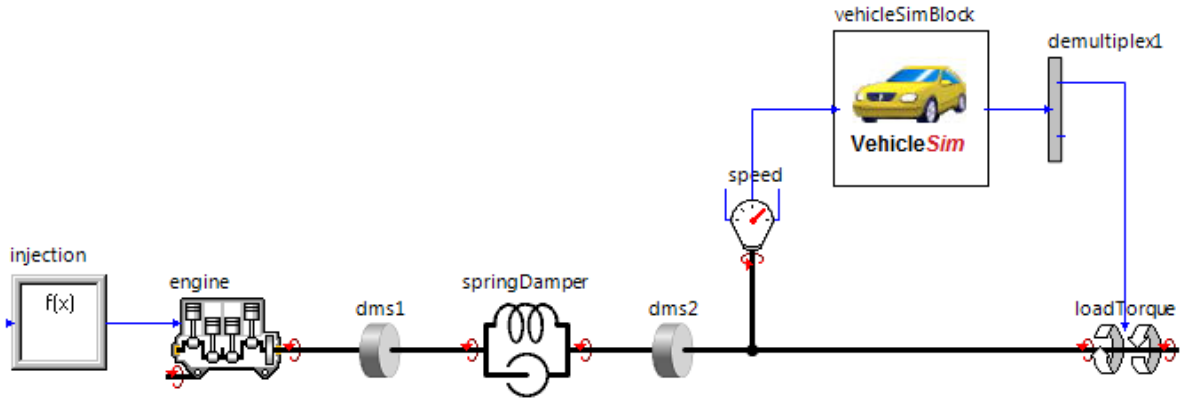



**Bild 9.36:** SimulationX- Modell bereit für die Co-Simulation mit CarSim

Jetzt ist das SimulationX-Modell bereit für die Co-Simulation mit CarSim. Wollen Sie mehr als 1 Signal nach CarSim übergeben, so ist die Dimension von `nu` zu ändern und es ist ein entsprechender Multiplexer zur Umwandlung der Signale in einen Vektor zu nutzen. Im Package CoSimulation.MuxDemux werden Multiplexer und Demultiplexer der Dimension 2 bis 5 bereitgestellt. Werden Multiplexer/Demultiplexer mit höherer Dimension benötigt, so können Sie mit dem TypeDesigner eigene Blöck auf der Basis des vorhandenen Blocks erstellen (siehe **Kapitel 8**). Die Multiplexer/Demultiplexer Blöcke sind offen und können mit dem TypeDesigner eingesehen werden.

### 9.3.4.6 Co-Simulation

Nach den folgenden Einstellungen auf der CarSim “Self Contained Solvers”-Seite kann die Co-Simulation gestartet werden. Wechseln Sie zu CarSim und öffnen Sie das Dataset, das wir zu Anfang erzeugt haben, indem Sie auf die Schaltfläche ***SimulationX*** im CarSim Run Control betätigen.

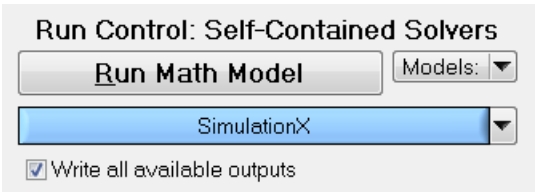



**Bild 9.37:** SimulationX Datensatz im CarSim Run Control

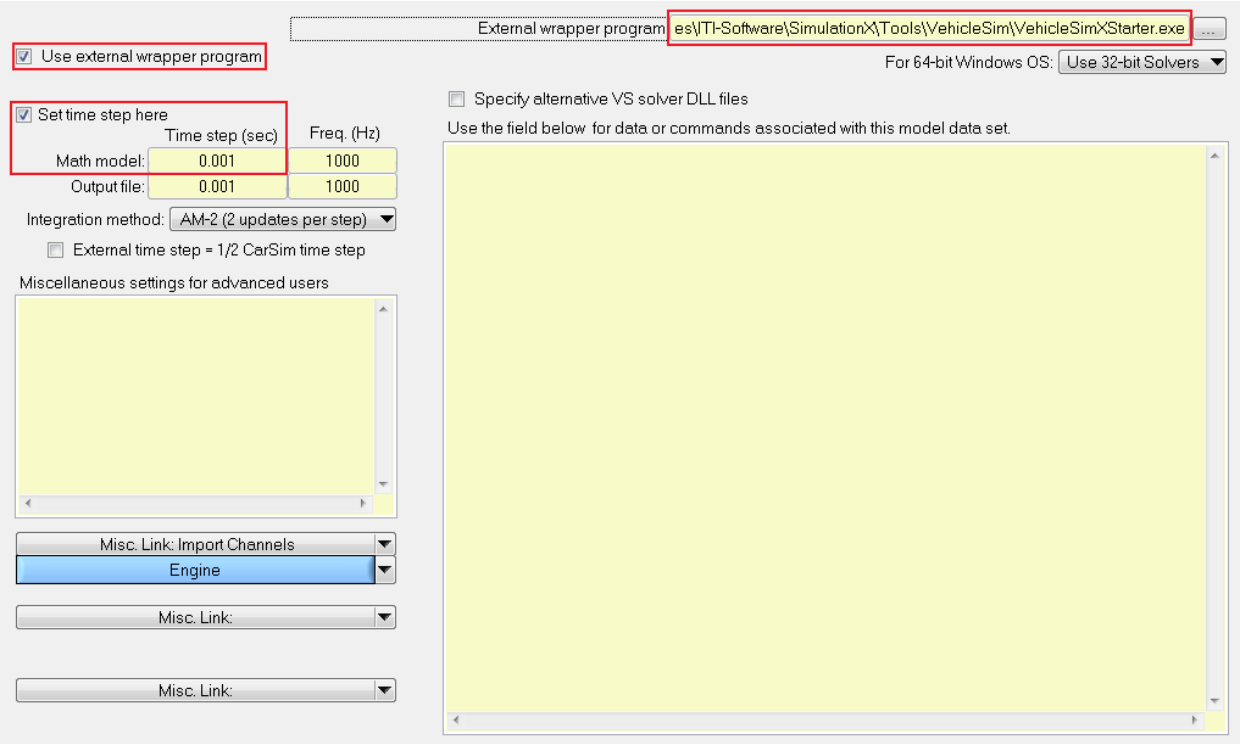



**Bild 9.38:** Select the VehicleSimXStarter.exe as external wrapper program

Schalten Sie die ***Use external wrapper program*** ein und wählen als External Wrapper Program die Datei "VehicleSimXStarter.exe". Die VehicleSimXStarter-Anwendung ist im Verzeichnis *SimulationX/Tools/VehicleSim* gespeichert.

Weiterhin ist die Größe für die Zeitschritte mit 0.001 s festzulegen. Der Wert muss mit dem Zeitschritt `ts` des SimulationX Co-Simulationsblocks übereinstimmen. Nun kann die Simulation durch Drücken der Schaltfläche ***Run Math Model*** im CarSim Run Control gestartet werden. Stellen Sie sicher, dass SimulationX läuft und das Modell mit dem VehicleSim Co-Simulationsblock, das aktive ist. Anstelle der CarSim internen Berechnungsroutine wird jetzt die VehicleSimXStarter-Anwendung gestartet.

Diese Anwendung führt folgende Aktionen aus:

- Verbindet sich selbständig mit der laufenden SimulationX Instanz oder öffnet eine neue SimulationX Instanz.
- Verbindet sich mit dem aktiven Modell.
- Prüft, ob ein VehicleSim Co-Simulationsblock im Modell vorhanden ist.
- Setzt das aktive Modell zurück, falls notwendig.
- Startet die Simulation in SimulationX. Jetzt kommuniziert der VehicleSim Co-Simulationsblock über die VehicleSim API mit CarSim.
- Zeigt Statusanzeigen während der Simulation an.
- Schließt sich selbstständig nach Ende der Simulation.

- Wenn SimulationX nicht gefunden bzw. nicht geöffnet werden kann oder das aktive Modell keinen VehicleSim Co-Simulationsblock enthält, wird eine Fehlermeldung ausgegeben und die Simulation startet nicht.

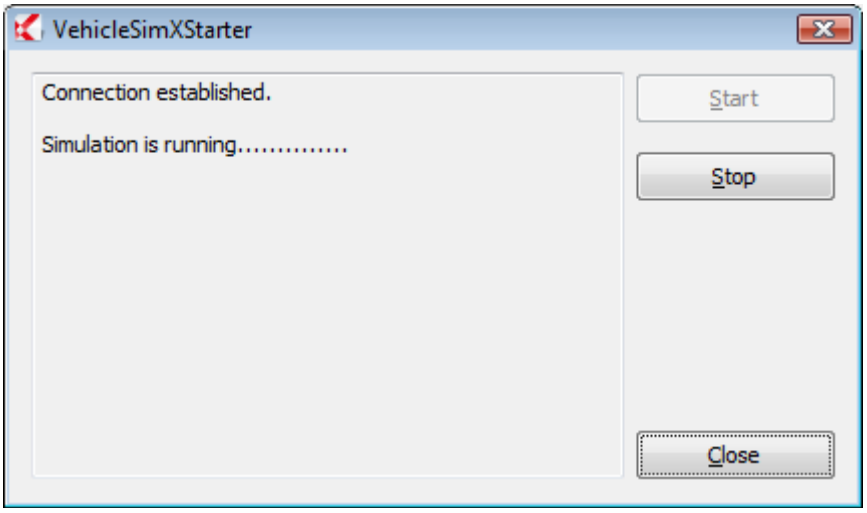



**Bild 9.39:** Der VehicleSimXStarter ist mit SimulationX verbunden und die Simulation läuft

Die Co-Simulation stoppt unter folgenden Bedingungen:

- Wenn im VehicleSimXStarter die Schaltfläche ***Stop*** betätigt wurde. Mit der Schaltfläche ***Start*** kann fortgesetzt werden.
- Wenn in CarSim die Stopzeit erreicht wurde oder eine CarSim Stop-Bedingung wahr wurde. In diesem Fall wird in SimulationX eine Meldung ausgegeben.
- Wenn in SimulationX die Schaltfläche ***Stop*** betätigt oder eine Abbruch-bedingung wahr wurde. VehicleSimXStarter schließt sich in diesem Fall automatisch. Die Simulation kann durch betätigen der Schaltfläche ***Start*** in SimulationX fortgesetzt werden.

Die Simulation kann auch von SimulationX gestartet werden. In diesem Fall ist es möglich, das die CarSim Modelldaten nicht an die CarSim Datenbankdateien übertragen werden. Selbst wenn Sie die CarSim Datenbank speichern, werden die Modelldaten nicht in die Datenbank geschrieben. Dies erfolgt nur durch CarSim, wenn Sie das externe Wrapper-Programm (in unserem Fall VehicleSimXStarter) starten. Daher ist es sicherer die Co-Simulation von CarSim aus zu starten. Es ist nur sicher aus SimulationX zu starten, wenn vorher ein Start aus CarSim erfolgte und keine Änderungen am CarSim Model vorgenommen wurden.

### 9.3.4.7 Post processing in CarSim and SimulationX

Nach der Co-Simulation stehen die üblichen Postprocessing Funktionen von CarSim (Animation, Plot) und in SimulationX (Ergebnisfenster) zur Verfügung. Das folgende Bild zeigt ein SimulationX Ergebnisfenster und den CarSim Plot der Fahrzeuggeschwindigkeit.

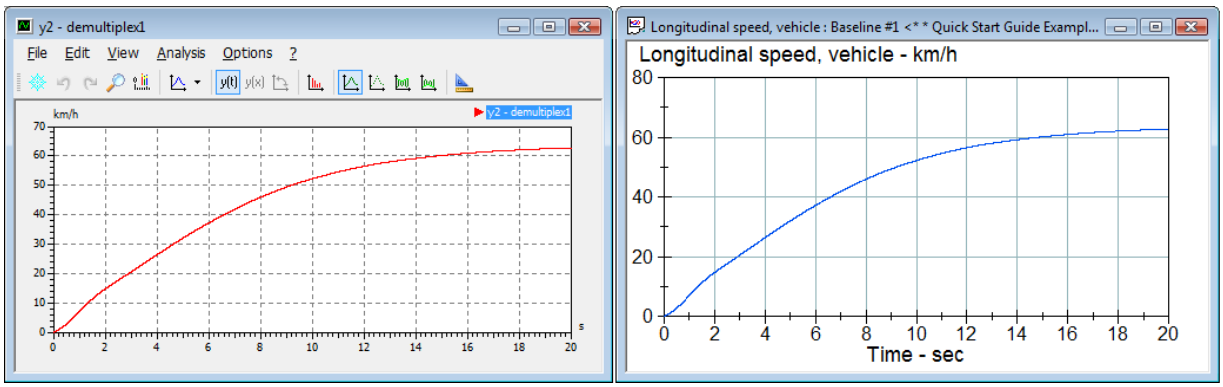
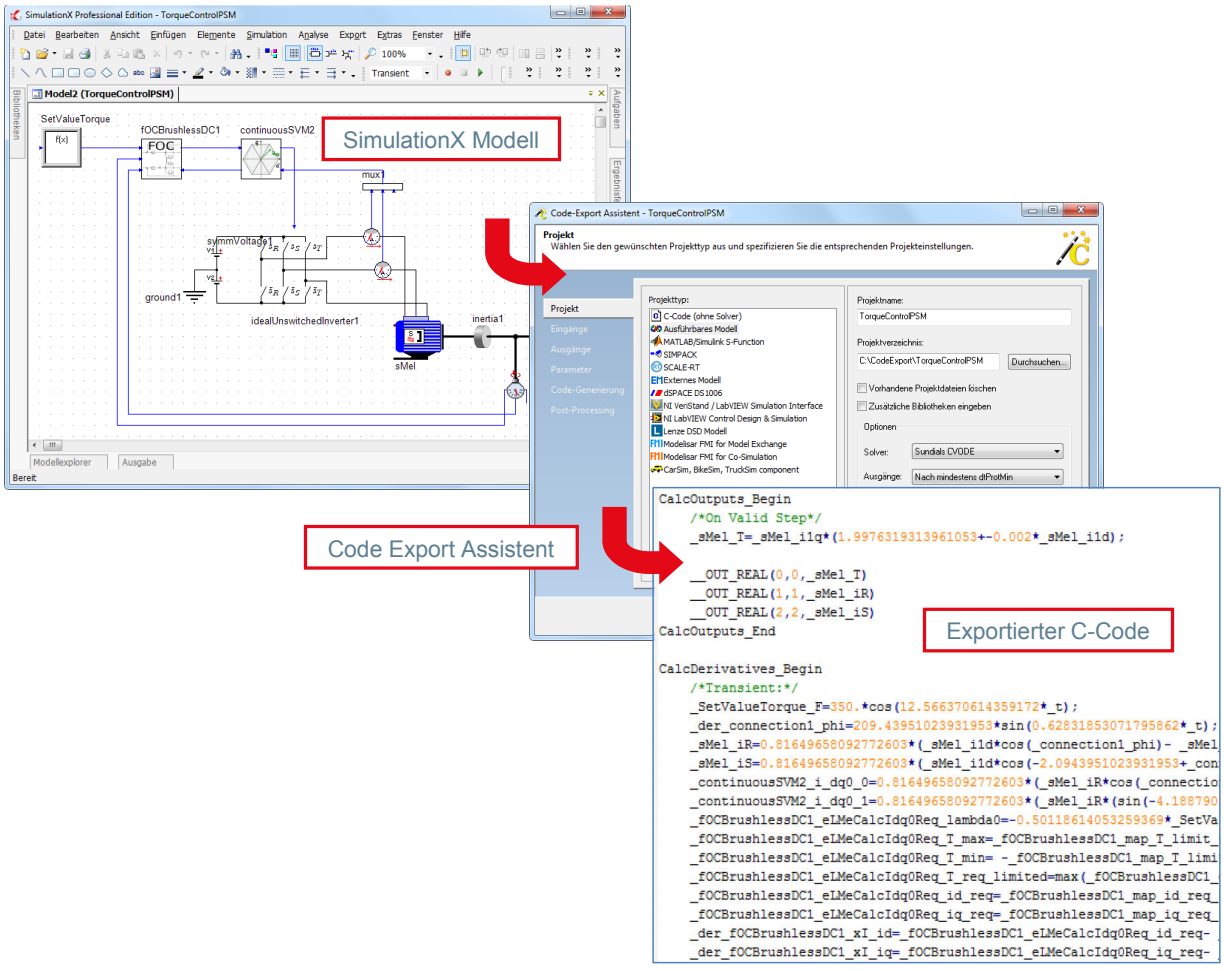



**Bild 9.40:** Vehicle speed in SimulationX and CarSim

# 9.4 Code-Export

## 9.4.1 Einführung




Mittels Code-Export lässt sich die Funktionalität eines kompletten SimulationX Modells in Quellcode der Programmiersprache C überführen.
Dieser Quellcode kann beispielsweise benutzt werden für:

- Beschleunigte Simulationsrechnungen.
- Modellintegration in andere Simulationsprogramme
- Hardware-in-the-Loop Anwendungen (HIL).

Vor dem Export wird festgelegt, welche Eingänge, Ausgänge, sowie Parameter im exportierten Modell als Interfaces bereitstehen sollen (siehe **Bild 9.41**). Bei der Interface-Definition werden Sie von einem C-Code-Export-Assistenten (siehe Abschnitt **9.4.2**) unterstützt.

Beim Export entstehen mehrere C-Quellcode- und Headerdateien, die die Modellfunktionalität bzw. Hilfsfunktionen enthalten.

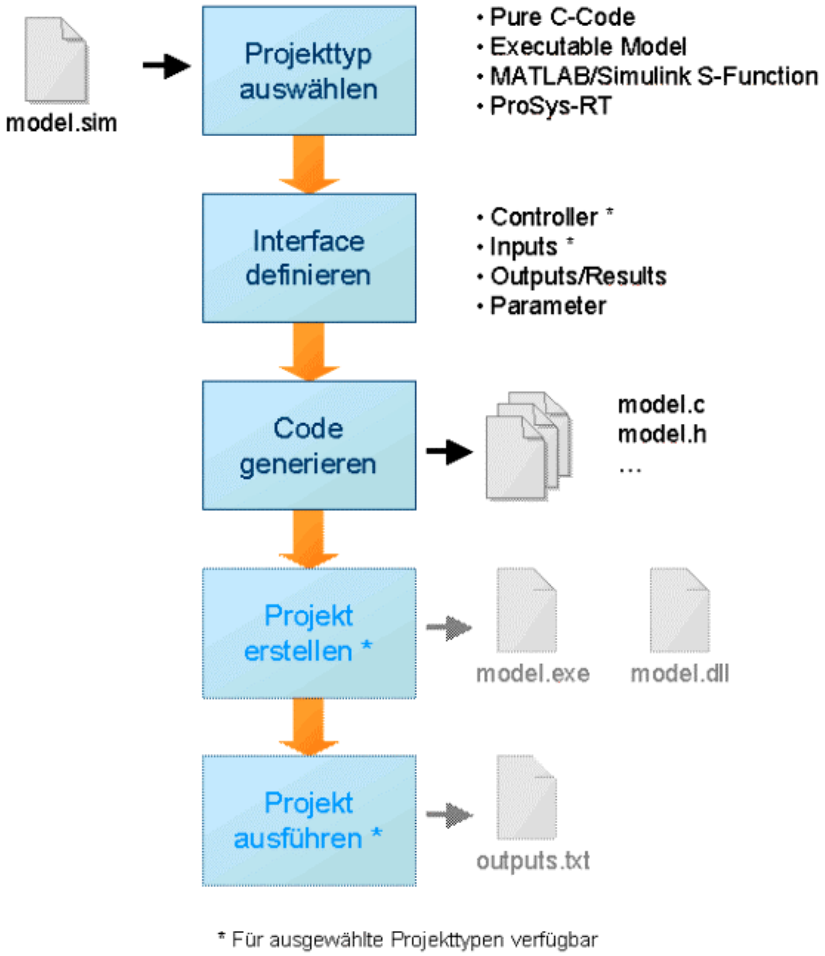



**Bild 9.41:** Projektablauf beim Code-Export

Vor dem Codeexport wird das Modell einer speziellen symbolischen Analyse unterzogen und in eine explizite gewöhnliche Differenzialgleichung überführt:

$$\begin{aligned} \dot{x} &= f(x,u,p,z,zf,t) \\ y &= g(x,u,p,z,zf,t) \\ z &= h_1(x,u,p,zf,t) \\ zf &= h_2(x,\dot{x},u,p,z,t) \end{aligned} \qquad \textbf{(9.1)}$$

Dabei sind:

x ..... Zustandsgrößen
u ..... Eingangsgrößen
y ..... Ausgangsgrößen

p ..... Parameter
z ..... diskrete Variablen
zf..... Zerofunctions
t..... Zeit

Die zur Berechnung dieser Differenzialgleichung nötige Funktionalität ist in der Datei "<Projektname>.c" implementiert, dessen Grundgerüst die folgenden drei Funktionen bilden:

1. Festlegung der Anfangswerte (*InitializeConditions*)
2. Berechnung der Ausgänge (*CalcOutputs*)
3. Berechnung der Ableitungen der Zustandsgrößen *(CalcDerivatives)*

Folgende Projekttypen stehen Ihnen zur Verfügung:

| Projekttyp | Bemerkungen |
|---|---|
| C-Code (ohne Solver) | • Nur die Modellfunktion wird exportiert |
| Ausführbares Modell | • Modellfunktion und Löser<br>• Für beschleunigte Simulationsrechnungen auf beliebigen Rechnerplattformen<br>• Einbindung in eigene Berechnungs- bzw. Auslegungstools möglich |
| MATLAB/Simulink S-Function | • Modellintegration in MATLAB/Simulink<br>• Weiterverarbeitung mit dem Real-Time Workshop möglich. Dadurch können folgende HiL-Systeme angesprochen werden:<br>o dSPACE<br>o ETAS<br>o OpalRT |
| SIMPACK | • Modellintegration in den MKS Simulator SIMPACK der Fa. Intec |
| SCALE-RT | • HiL- Umgebungen der Fa. cosateq GmbH |
| Externes Modell | • Export der Modellfunktionalität (ohne Solver) in einer DLL |
| dSPACE DS1006 | • DS1006 basierte Single-Prozessor-RCP-Systeme der Fa. dSPACE GmbH |
| NI VeriStand | • Direkte Unterstützung der Echtzeit-Test- und Simulationssoftware NI VeriStand der Firma National Instruments |

| | |
|---|---|
| NI LabView Control Design & Simulation | • Integration von SimulationX-Modellen in NI LabVIEW |
| Modelisar FMI for Model Exchange | • Export der Modellfunktionalität in eine FMU (für Model Exchange) |
| Modelisar FMI for Co-Simulation | • Export der Modellfunktionalität in eine FMU (für Co-Simulation) |
| CarSim, BikeSim, TruckSim component | • Modellintegration in VehicleSim, einer Werkzeugsammlung für die effiziente Simulation von Autos, Trucks und Motorrädern der Firma Mechanical Simulation Corporation |

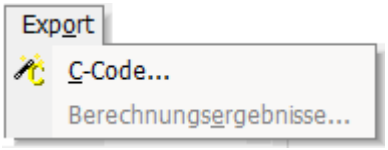
**Tabelle 9.1:** Projekttypen und ihre Verwendung

In Abhängigkeit vom Verwendungszweck des Modellcodes (dem Projekttyp) werden automatisch Interfacefunktionen erstellt, die die Passfähigkeit zum Zielsystem herstellen.

Falls Sie den Code Export zur Anbindung eines Modells an weitere Zielsysteme verwenden möchten, können Sie den Projekttyp C-Code (ohne Solver) benutzen und die notwendigen Interfacefunktionen selbst schreiben. Dabei unterstützen wir Sie gern.

## 9.4.2 Code-Export Assistent

Alle für den Code-Export des Modells notwendigen Einstellungen nehmen Sie im Code-Export Assistenten vor.




Sie erreichen ihn über das Menü ***Export/C-Code…***.

Die einzelnen Dialogseiten des Assistenten führen Sie nun durch die für Ihr gewähltes Zielprojekt notwendigen Schritte. Alle Einstellungen werden zusammen mit dem Simulationsmodell gespeichert.

### 9.4.2.1 Dialogseite Projekt

Auf der Projektseite bestimmen Sie den Projekttyp, Projektnamen, sowie den Speicherort der zum Projekt gehörenden Dateien. Standardmäßig bekommt das Projekt den gleichen Namen wie das SimulationX-Modell. Dieser kann editiert werden.

Während der Codegenerierung werden die Namen aller automatisch erzeugten Dateien in der Datei “generated.txt“ gespeichert. Wenn Sie das Häkchen bei “Vorhandene Projektdateien löschen“ gesetzt haben, werden vor der Codegenerierung alle Dateien, die in der Datei “generated.txt“ aufgeführt sind, gelöscht.

In Abhängigkeit vom selektierten Projekttyp werden weitere Einstellungsmöglichkeiten sichtbar. Diese werden in den entsprechenden Abschnitten weiter unten beschrieben.

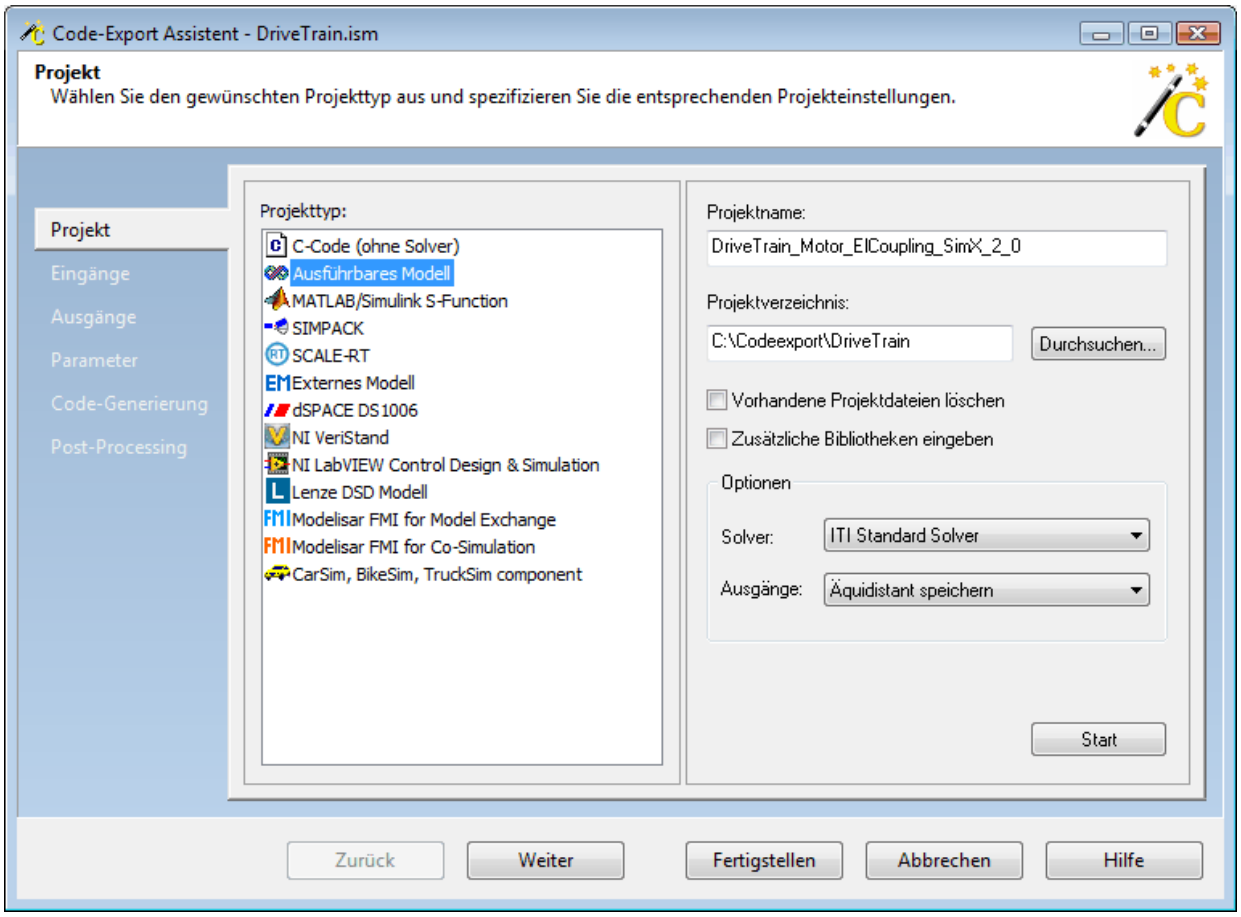



**Bild 9.42:** Code-Export Assistent – Dialog Projekt

Setzen Sie das Häkchen bei “Zusätzliche Bibliotheken eingeben“ wenn das Modell externe Funktionen enthält und Sie festlegen wollen für welche Bibliotheken Code zum dynamischen Aufruf von Funktionen in einer DLL generiert werden sollen. Mehr Informationen zu diesem Thema finden Sie im Abschnitt **9.4.2.7**.

Mit der Schaltfläche Start starten Sie bereits hier die Code-Generierung für ein bereits definiertes Projekt. Die Dialogseiten zur Definition von Eingängen, Ausgängen und Parametern werden in diesem Fall übersprungen.

Die Zuordnung der Parameter, sowie der Ein- und Ausgänge wird in der Datei Readme.txt zur Dokumentation gespeichert.

### 9.4.2.2 Auswahl der Signaleingänge

Diese Auswahl steht nur bei:

- C-Code (ohne Solver)
- Ausführbares Modell
- MATLAB/Simulink S-Funktion

- SIMPACK
- SCALE-RT
- Externes Modell
- NI VeriStand
- NI LabVIEW Control Design & Simulation
- Modelisar FMI für Model Exchange
- Modelisar FMI für Co-Simulation
- CarSim, BikeSim, TruckSim component

zur Verfügung.

Als Eingänge stehen alle freien Signaleingänge der Modellobjekte zur Verfügung (siehe **Bild 9.43**). Um einen Signaleingang einzufügen, ziehen Sie diesen per *Drag & Drop* von der Objektübersicht in das Auswahlfeld (oder per Doppelklick). Mit können Sie den Signaleingang in der Auswahl auf- bzw. abwärts verschieben. In der so definierten Reifenfolge werden beispielsweise beim Projekttyp Simulink S-Function die Eingänge im entstehenden Block von oben nach unten angelegt.

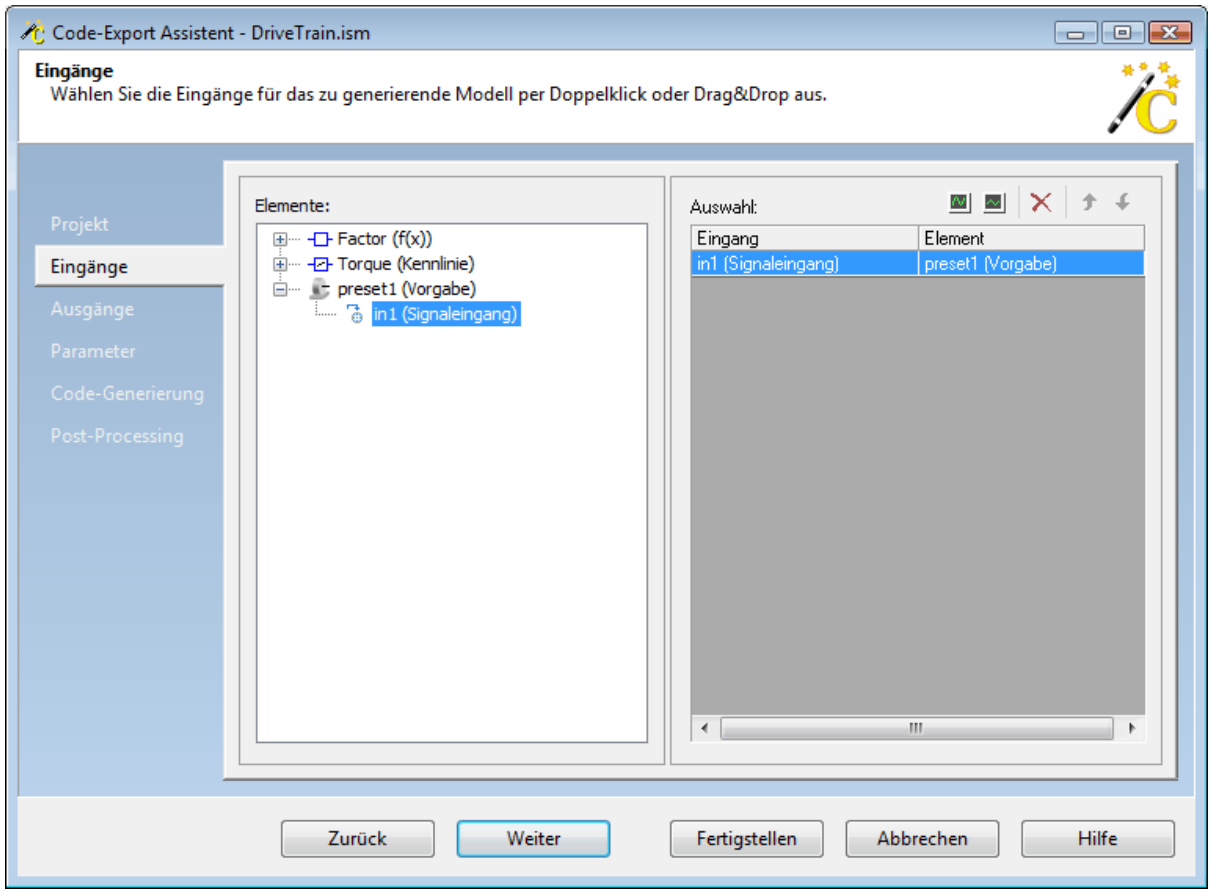



**Bild 9.43:** Auswahl der Signaleingänge

Zusätzlich haben Sie Möglichkeit mit dem Schalter alle Eingänge, oder mit dem Schalter nur die obersten Eingänge, des Modells komplett in die Auswahl zu übernehmen.

Über den Schalter kann ein ausgewählter Eingang wieder gelöscht werden.

### 9.4.2.3 Auswahl der Ausgänge

Als Ausgänge für das zu generierende Modell können alle Ergebnisgrößen und Signalausgänge genutzt werden. Im **Bild 9.44** ist eine entsprechende Auswahl dargestellt.

Es gelten dieselben Bedienfunktionen, wie bei den Signaleingängen. Zusätzlich haben Sie Möglichkeit mit dem Schalter alle Ergebnisgrößen, oder mit dem Schalter nur die obersten Ergebnisgrößen, des Modells komplett in die Auswahl zu übernehmen.

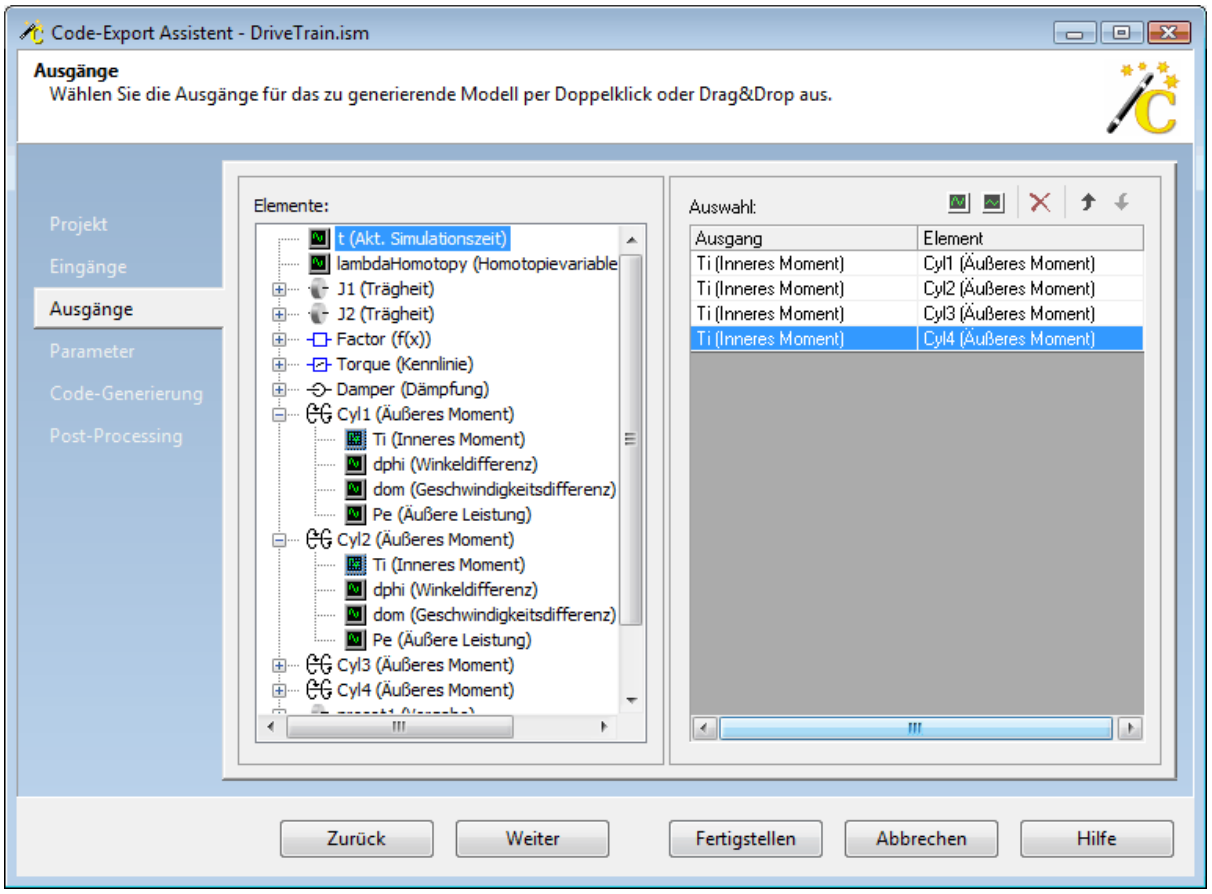



**Bild 9.44:** Auswahl der Ausgänge

### 9.4.2.4 Auswahl der Parameter

Auf dieser Seite können bestimmte Parameter des Modells für den Codeexport nach außen zugänglich gemacht werden. Die Bedienung erfolgt analog zu den vorherigen Abschnitten.

Hinweis: Es kann nur ein Parameter einer Parametergruppe als Parameter im Code-Export Assistenten ausgewählt werden, damit keine Zuweisungsgleichung in der GSA erzeugt wird. Weiterhin können Parameter die auf andere Parameter verweisen, nicht ausgewählt werden.

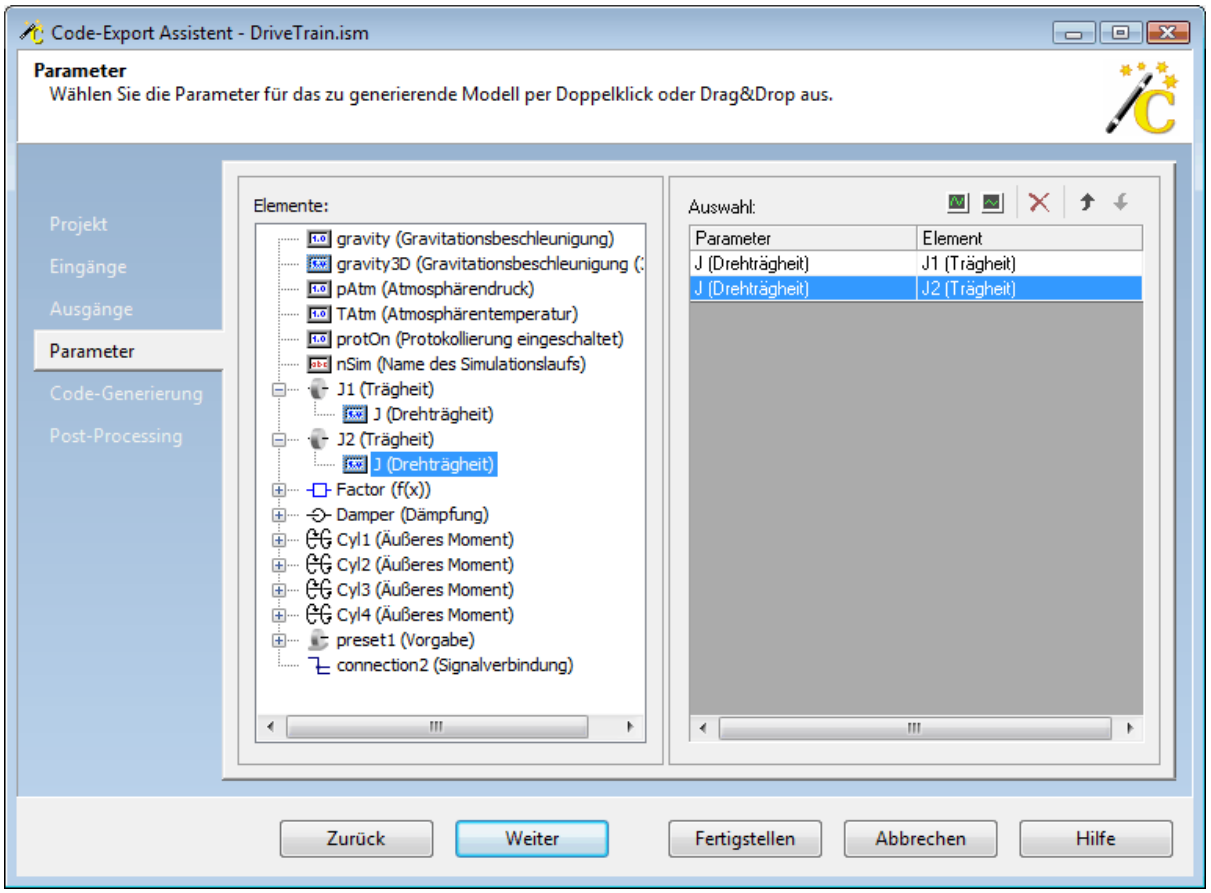



**Bild 9.45:** Auswahlfenster für Parameter

### 9.4.2.5 Auswahl der Kennlinien (nur dSPACE DS1006)

Auf dieser Seite bestimmen Sie die Kennlinien, über die das exportierte SimulationX-Modell Daten aus dem Zielsystem entgegen nehmen kann.

### 9.4.2.6 Task Handling (nur dSPACE DS1006)

Sie können die Eigenschaften der zyklischen Timer-Task sowie den initialen Simulationszustand der Echtzeitapplikation spezifizieren.

### 9.4.2.7 Zusätzliche Bibliotheken

Auf dieser Seite werden, nach der Durchführung der Globalen Symbolischen Analyse (GSA), die im Modell für den Aufruf von externen Funktionen notwendigen zusätzlichen Bibliotheken angezeigt. Für jede Bibliothek kann festgelegt werden, ob Code zum dynamischen Aufruf von Funktionen in einer DLL generiert werden soll. Falls nicht, sind zum Erstellen die entsprechenden Bibliotheken notwendig. Die Pfade zu diesen Bibliotheken können in dem Eingabefeld „Zusätzliche Bibliothekspfade“ angegeben werden.

Einzelne Bibliotheken lassen sich manuell über die entsprechenden Schaltflächen hinzufügen und löschen.
Sollte das zu exportierende Modell ITI-Bibliotheken enthalten, die über externe Funktionen angesprochen werden (z.B. das Element “Weißes Rauschen“) und das Häkchen für das dynamische Laden gesetzt sein, wird die entsprechende Bibliothek automatisch in das spezifizierte Projektverzeichnis kopiert. Dabei sollte man aber daran denken, dass in Abhängigkeit von der spezifischen Bibliothek gegebenenfalls weitere ITI-Bibliotheken nachgeladen werden.

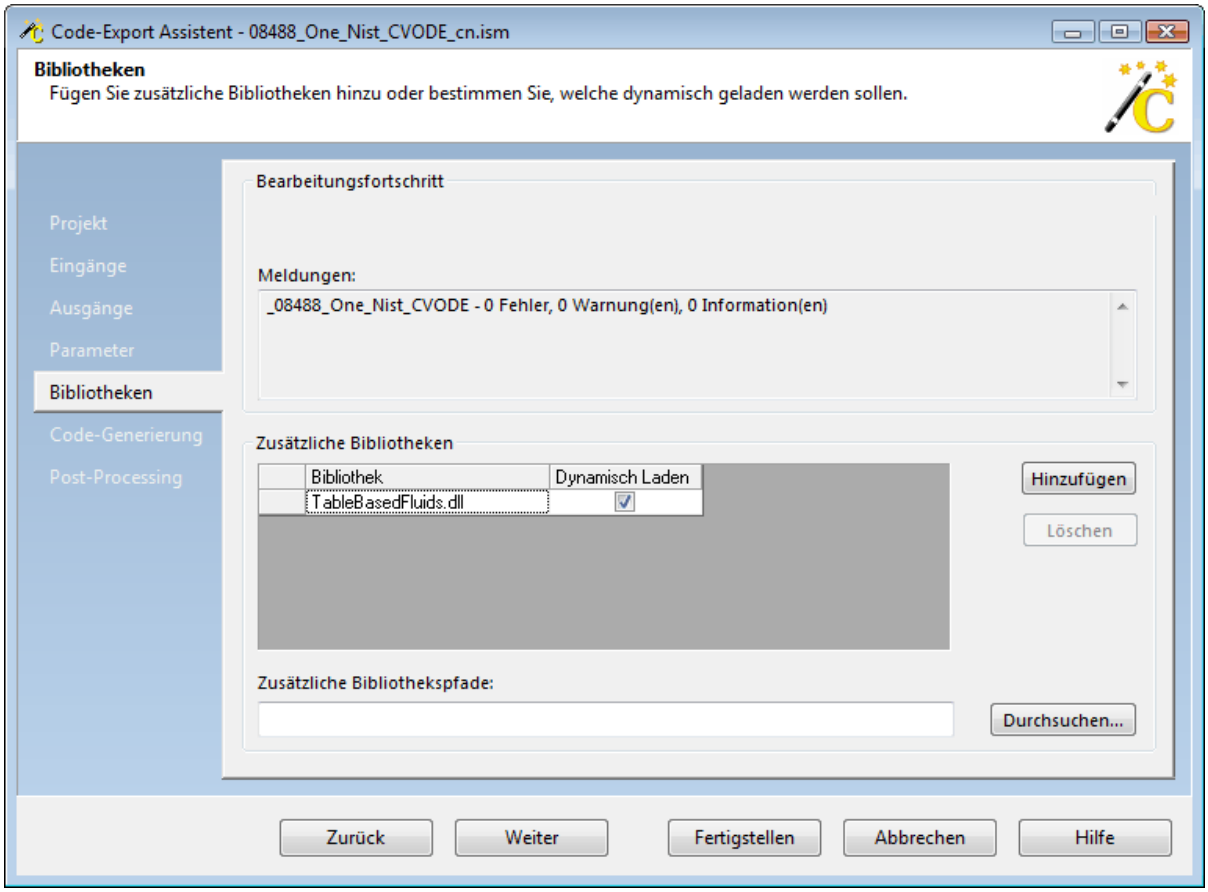



**Bild 9.46:** Auswahl der zusätzlichen Bibliotheken

### 9.4.2.8 Code-Generierung

Mit dem Aufruf dieser Seite des Assistenten startet die eigentliche Generierung der C-Quelltexte.
Zuvor wird das Modell einer speziellen symbolischen Analyse unterzogen. Sie hat das Ziel das Modell in ein System expliziter gewöhnlicher Differenzialgleichungen umzuformen. Dabei erfolgt auch eine Optimierung des Berechnungsalgorithmus.

Während des Exports können Sie den Fortschritt im Generierungsfenster (siehe **Bild 9.47**) beobachten.

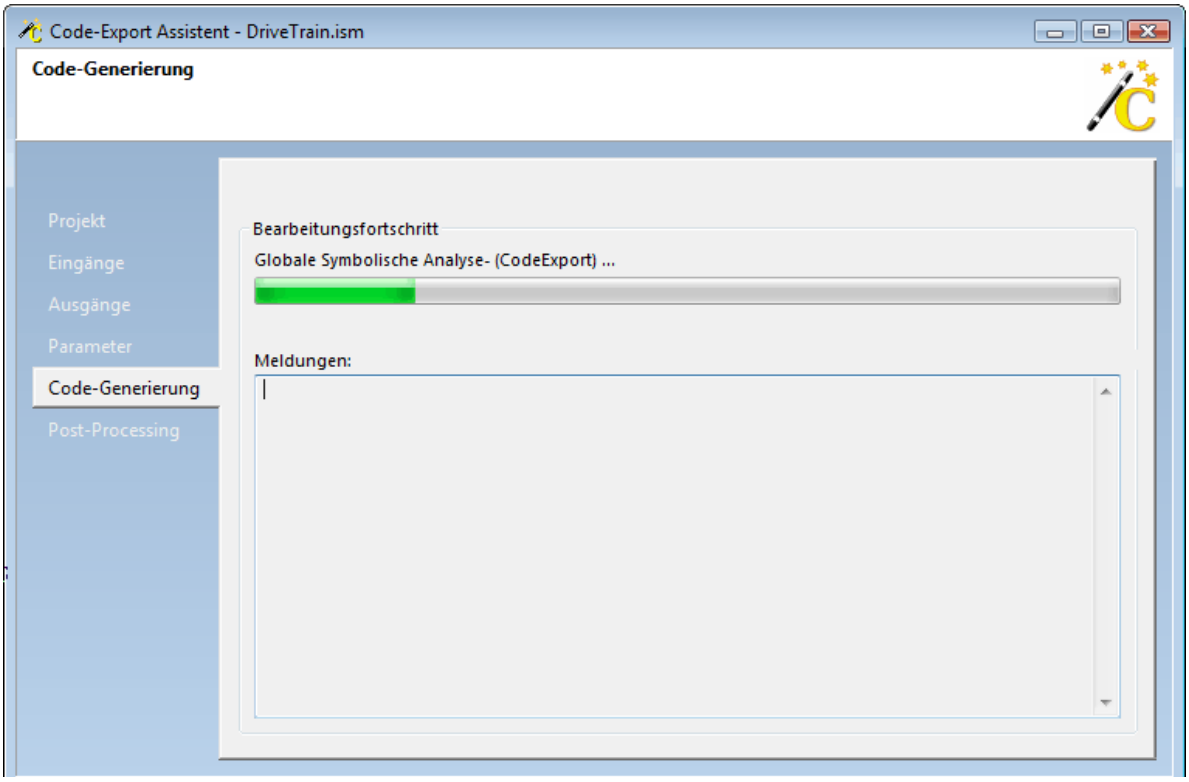



**Bild 9.47:** Anzeige des Fortschritts der Codegenerierung

Nach erfolgreicher Erstellung des exportierten Codes erhalten Sie eine Meldung, wie in **Bild 9.48**.

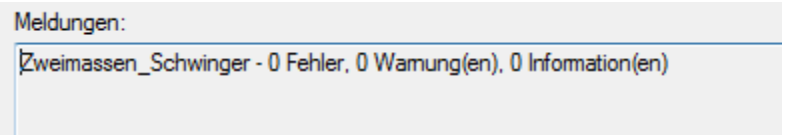



**Bild 9.48:** Meldung bei erfolgreichem Export

Alle generierten Dateien finden Sie in dem von Ihnen vorgegebenen Projektverzeichnis. Diese können nun für die Weiterverarbeitung (z. B. Kompilieren, Linken) auf der Zielplattform genutzt werden. Die Namen der generierten Dateien finden Sie in der Datei „generated.txt“.
Die Datei „Readme.txt“ enthält eine Auflistung der eingestellten Ein- und Ausgänge sowie der Parameter.

### 9.4.2.9 Nachbereitung

Auf dieser Seite können Sie das generierte Projekt kompilieren und starten. Diese Dialogseite steht bei folgenden Projekttypen zur Verfügung:

- Ausführbares Modell
- MATLAB/Simulink S-Function
- SCALE-RT
- Externes Modell
- dSPACE DS1006
- NI VeriStand

- NI LabVIEW Control Design & Simulation
- Modelisar FMI für Model Exchange
- Modelisar FMI für Co-Simulation
- CarSim, BikeSim, TruckSim component

Wählen Sie die Entwicklungsumgebung, unter welcher das generierte Projekt kompiliert werden soll. SimulationX erkennt automatisch, welche Microsoft C-Compiler auf ihrem System vorhanden sind. Im Abschnitt **2.1.2** finden sie mehr Informationen zu den Entwicklungsumgebungen, die von SimulationX unterstützt werden.

## 9.4.3 Projekttypen

### 9.4.3.1 C-Code (ohne Solver)

Es entstehen die Quelltexte für das Modell und Dateien mit verschiedenen Funktionen, die das Modell verwendet. Diese Quelltexte können in Ihr Zielprojekt eingebunden werden.

| Dialogseite | Beschreibung |
|---|---|
| Eingänge | Die aktuellen Werte der Eingänge werden als Übergabeparameter _u der exportierten Funktionen erwartet. |
| Ausgänge | Die aktuellen Werte der Ausgänge werden in der Funktion CalcOutputs in das Feld _y geschrieben. |
| Parameter | Die aktuellen Werte der Parameter werden als Übergabeparameter _p der exportierten Funktion. |

Zur beschleunigten Simulation kann das Modell zusammen mit einem Löser exportiert werden. Die erzeugten plattformunabhängigen Quelltexte müssen auf dem gewünschten Zielsystem kompiliert und gelinkt werden. Es entsteht eine ausführbare Datei (im weiteren „*Ausführbares Modell*“), die nach dem Start die Simulationsrechnung durchführt und die Ergebnisse in eine Textdatei schreibt. **Bild 9.34** zeigt den Ablauf bei der Generierung eines Ausführbaren Modells. Zusätzlich zu den Quelltexten werden Projektdateien für die Entwicklungsumgebung Microsoft Visual C++ 6.0 erzeugt. Das so erzeugte Projekt kann aber auch z.B. in Microsoft Visual Studio.Net 2003 geöffnet werden.
Weiterhin wird das makefile „makefile.txt“ für das Erstellen unter anderen Betriebssystemen erzeugt.

### 9.4.3.2 Ausführbares Modell

Auf der Dialogseite Projekt kann eingestellt werden, mit welchem Löser das Modell berechnet werden soll. Folgende Verfahren stehen zur Auswahl:

- Sundials CVODE (Löser mit variabler Schrittweite, geeignet für steife Modelle)

- ITI Fixed-Step Solvers (Löser mit konstanter Schrittweite, geeignet zum Schrittweitentest für Real-Time Umgebungen)
- ITI Standard Solver (Löser mit variabler Schrittweite, geeignet für nicht sehr steife Modelle mit vielen Unstetigkeiten)

Auch kann hier eingestellt werden, wie die Simulationsergebnisse (die Ausgänge) protokolliert (gespeichert) werden:

- Komplett speichern (Speichern der Ausgänge zu jedem gültigen Schritt)
- Äquidistant speichern (äquidistantes Speichern der Ausgänge mit dtProtMin)
- Nach mindestens dtProtMin (früher „Analog zu SimulationX“)
- Nach mindestens dtProtMin und nach Ereignissen
- Nach mindestens dtProtMin sowie vor und nach Ereignissen
- Nach mindestens dtProtMin und Eventiterationsschritten
- Letzter Wert

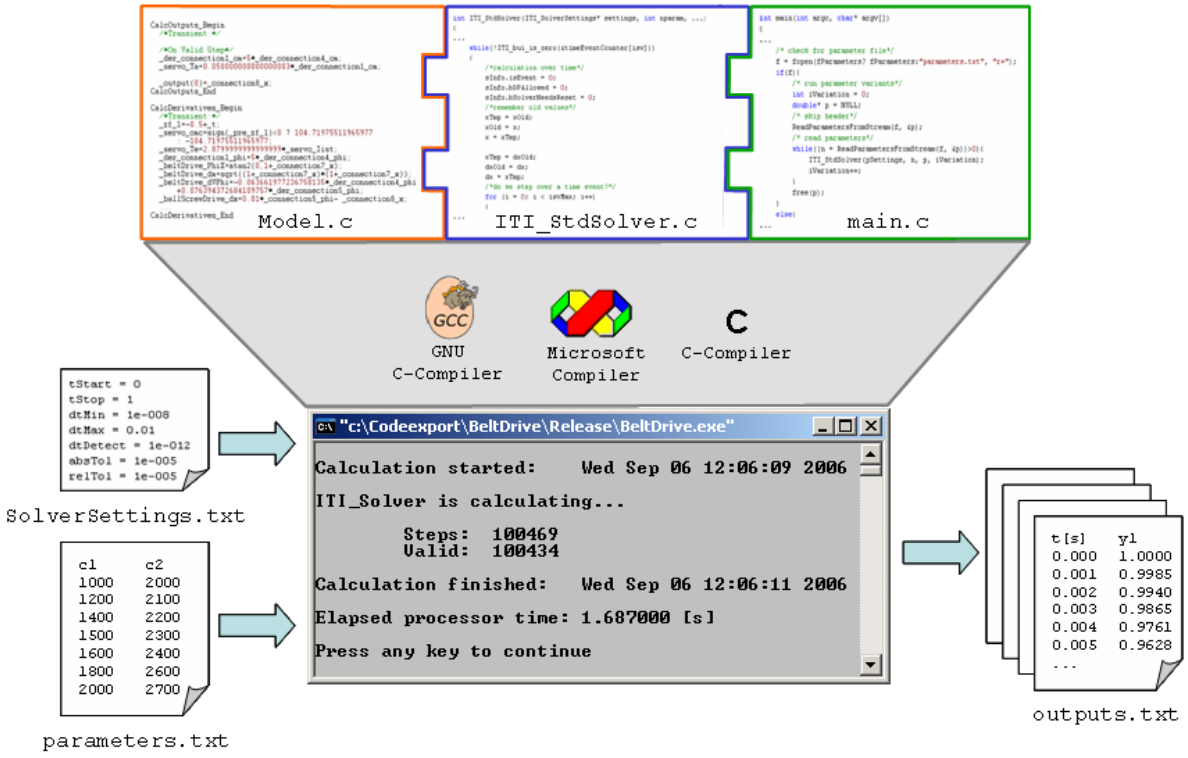



**Bild 9.49:** Ablauf bei der Erzeugung eines Ausführbaren Modells

Folgende Einstellungen müssen Sie im Code-Export Assistenten vornehmen:

| Dialogseite | Beschreibung |
|---|---|
| Projekt | Neben dem Projektnamen und dem Projektverzeichnis können Sie den zu verwendenden Löser und die Art der Ergebnisspeicherung festlegen. |

| | |
|---|---|
| Eingänge | Die ausgewählten Größen können durch das Übergeben einer selbst implementierten Funktion via Funktions zeiger (in der exportierten Datei „main.c“, siehe Kommentar „SetInputsFunc“) an den jeweiligen Löser, gesetzt werden. Die Signatur dieser Funktion ist in der exportierten Datei „ITI_Types.h“ definiert. Für weitere Details kontaktieren Sie bitte die ITI Supportabteilung. |
| Ausgänge | Die ausgewählten Größen werden während der Rechnung in die Datei outputs.txt geschrieben. |
| Parameter | Die aktuellen Parameterwerte aus dem Modell werden in die Datei “parameters.txt“ geschrieben. Diese Datei liest das Ausführbare Modell nach dem Start (siehe auch weiter unten). |
| Nachbereitung | Hier können Sie das Ausführbare Modell kompilieren und starten (siehe weiter unten). |

Die Schrittweiten- und Genauigkeitseinstellungen für den Löser werden zum einem im Modellquelltext (siehe Datei <Projektname>.h), aber auch in der Datei „solversettings.txt“ gespeichert. Diese Datei kann zum Ändern der Löserein-stellungen ohne Neukompilierung verwendet werden.

Findet das Ausführbare Modell diese Datei im Startverzeichnis, werden die dort festgelegten Einstellungen verwendet, ansonsten die Einstellungen aus dem Modellquelltext.

Wenn Sie das Ausführbare Modell mit dem Aufrufparameter „/s <Dateiname>“ aufrufen, wird versucht diese Datei zu nutzen.

Die Datei „solversettings.txt“ hat folgendes Format:

Schlüsselwort = Zahl

Die Bedeutung der Schlüsselwörter in den einzelnen Lösern zeigt die folgende Tabelle.

| Schlüsselwort | ITI Fixed-Step Solvers | ITI Standard | CVODE |
|---|---|---|---|
| `tStart` | Startzeit | | |
| `tStop` | Stoppzeit | | |
| `dtMin` | Integrationsschrittweite | Minimale Schrittweite | |
| `dtMax` | | Maximale Schrittweite | |

| | | | |
|---|---|---|---|
| limitdtMin | | | 0 - Keine Begrenzung der minimalen Schrittweite<br>1 - Begrenzung auf dtMin |
| absTol | | | Absolute Toleranz |
| relTol | | Relative Toleranz | |
| dtProtMin | Minimale Protokollschrittweite | | |
| mode | 0 – Euler-Vorwärts<br>1 – ITI Standard (mit konst. Schrittweite)<br>2 – Verfahren von Heun<br>3 – RKF23<br>4 – DOPRI5 | | 0 - Adams-Variante<br>1 - BDF-Variante |
| minmax | 0 – deaktiviert<br>1 – Minimum-Behandlung für Zustandsgrößen | | |
| zeros | | | 0 - deaktiviert<br>1 - Exakte Lokalisierung von Ereignissen unter Nutzung von Zerofunctions |
| bHysteresis | | | Algorithmen zur Lokalisierung von Ereignissen nutzen eine Hysterese epsilon. |
| epsilon | | | Hysteresewert bei der Lokalisierung von Eriegnissen. |
| saveMode | 6 - Letzter Wert<br>Bei allen anderen Einstellungen wird äquidistant gespeichert. | 0 - Alles<br>1 - Äquidistant<br>2 - Nach mindestens dtProtMin<br>3 - Nach mindestens dtProtMin und nach Ereignissen | |

| | | | |
|---|---|---|---|
| | | 4 - Nach mindestens dtProtMin sowie vor und nach Ereignissen<br>5 - Nach mindestens dtProtMin und Eventiterationsschritten<br>6 - letzter Wert | |
| `traceOn` | 0 - Unterdrückung der trace Ausgaben aus dem Modell<br>1 - Anzeige der trace Ausgaben aus dem Modell | | |
| `effJac` | | | 0 – Ohne effiziente Jacobimatrixberechnung<br>1 – Effiziente Jacobi-matrixberechnung und Gauß-Verfahren<br>2 – Effiziente Jacobi-matrixberechnung und Sparse Matrix Löser |
| `maxOrder` | | | Maximale Verfahrensordnung<br>mode = 1: {1...5}<br>mode = 0: {1...12} |
| `blTol` | Fehlertoleranz bei der Lösung impliziter Blöcke | | |
| `blLinSolv` | Linearer Gleichungslöser bei impliziten Blöcken:<br>0: LU-Zerlegung 1: QR-Zerlegung | | |
| `threadLimit` | Legt die Anzahl der verwendeten logischen Prozessorkerne bei der parallelisierten Variantenrechnung mit OpenMP fest. Detaillierte Informationen finden Sie in dem Abschnitt „Parallelisierte Variantenrechnung“. | | |

**Tabelle 9.2:** Format der Datei solversettings.txt

Der Defaultwert für `mode` ist 1, d. h. Fixed-Step Löser mit einem expliziten Zweischrittverfahren. Dieses Verfahren wird auch bei den Echtzeit-Projekttypen dSPACE, ProSys-RT, SCALE-RT und NI Veristand verwendet. CVODE verwendet ein BDF-Verfahren mit maximaler Verfahrensordnung 5. Beachten Sie, dass die maximale Verfahrensordnung der Adams-Variante 12 beträgt. Somit muss auch der Parameter `iMaxOrder` beim Setzen von `mode` auf 0 angepasst werden.

Wählen Sie hier die Entwicklungsumgebung, unter der das von Ihnen generierte Projekt übersetzt werden soll. SimulationX erkennt automatisch folgende auf Ihrem System installierten Microsoft C-Compiler:

- Microsoft Visual C/C++ 6.0
- Microsoft Visual C/C++ .Net

- Microsoft Visual C/C++ .Net 2003
- Microsoft Visual C/C++ Toolkit 7.0
- Microsoft Visual C/C++ 2005 (ab Standard Edition)
- Microsoft Visual C/C++ 2008 Express Edition (kostenlos)
- Microsoft Visual C/C++ 2008 (ab Standard Edition)
- Microsoft Visual C/C++ 2010 Express Edition (kostenlos)
- Microsoft Visual C/C++ 2010 (ab Professional Edition)

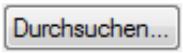
***Nachbereitung:***
Für die Nachbereitung müssen die Voraussetzungen entsprechend Abschnitt **2.1.2** erfüllt sein. Wählen Sie einen auf ihrem System vorhanden Compiler aus.

Klicken Sie auf diese Schaltfläche um zusätzliche Einstellungen für die Nachbereitung festzulegen. Detaillierte Informationen zu diesen Einstellungen finden Sie in dem folgenden Abschnitt.

Kompiliert und erstellt das von Ihnen erzeugte Projekt. Die Ergebnisse dieses Prozesses werden im Ausgabebereich angezeigt.

Startet die Ausführung des übersetzten Projektes. Informationen zur Ausführung werden im Ausgabebereich angezeigt.

***Erweiterte Einstellungen:***
Hier haben Sie die Möglichkeit zusätzliche Bibliotheken (.lib-Dateien) und die entsprechenden Verzeichnisse, in denen sich diese befinden, anzugeben. Dies ist dann notwendig, wenn das exportierte Modell externe Funktionen enthält, die in einer DLL (Dynamic Link Library) implementiert sind. Die so spezifizierten Bibliotheken werden dann beim Erstellen des ausführbaren Modells während des Link- Prozesses hinzugefügt. Anschließend können Sie das so erstellte Modell ausführen.

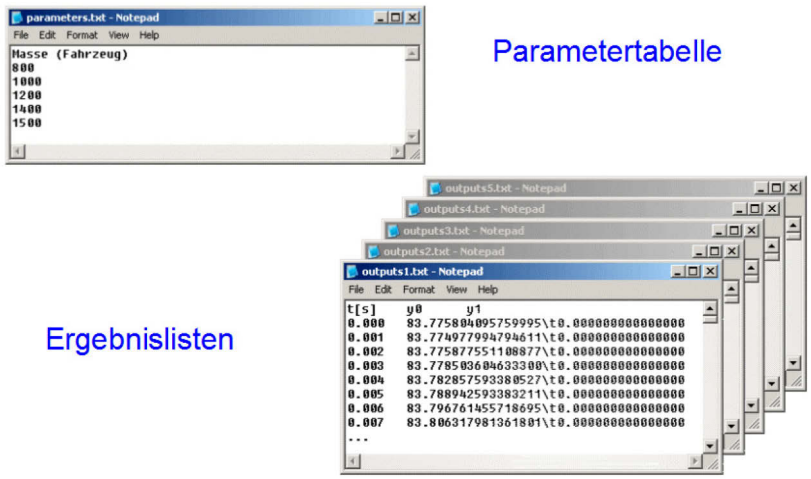



Klicken Sie auf diese Schaltfläche um über einen Dialog bequem zusätzliche Bibliotheken bzw. die entsprechenden Verzeichnisse festzulegen. Wenn Sie diese manuell eingeben stellen Sie bitte sicher, dass die einzelnen Bibliothekspfade durch ein Semikolon und die Bibliotheken durch ein Leerzeichen getrennt sind.

Durch das Aktivieren der Option „*OpenMP*“ kann die Variantenrechnung parallelisiert werden. Dafür wird die entsprechende Option im Compiler eingestellt. Mehr Informationen dazu finden Sie im folgenden Abschnitt „Parallelisierte Variantenrechnung“.

Durch das Aktivieren der Option „*Kompilierung mit Optimierung*“ kann die Rechengeschwindigkeit erhöht werden. In diesem Fall wird die Optimierungs-einstellung /O2 des Compilers eingestellt, was allerdings zu signifikant längeren Übersetzungszeiten führen kann.

***Variantenrechnungen:***
Um Variantenrechnungen durchzuführen, müssen Sie die zu variierenden Parameter beim Export angeben (siehe auch Abschnitt **9.4.2.4**).




Es wird dann die Textdatei „parameters.txt“ erzeugt, die die aktuell im SimulationX Modell eingestellten Parameterwerte als erste Zeile enthält. Wenn Sie weitere Varianten rechnen möchten, fügen Sie einfach neue Zeilen mit den gewünschten Werten hinzu. Beim Ausführen werden dann die entsprechenden Simulationsläufe durchgeführt und für jede Variante eine Ergebnisdatei erzeugt.
Um einen Parametersatz von der Variantenrechnung auszuschließen, fügen Sie in der entsprechenden Zeile der Datei „parameters.txt“ am Zeilenanfang die Zeichen „//“ für einen einzeiligen Kommentar ein.
Durch den hohen Optimierungsgrad des erzeugten Codes lassen sich so beschleunigte Simulationsrechnungen durchführen. Damit wird die Ausführung von umfangreichen Parameterstudien und komplexen Optimierungsaufgaben möglich.

Die Datei „parameters.txt“ wird vom ausführbaren Modell automatisch geladen, wenn sie sich im aktuellen Programmverzeichnis befindet. Wenn Sie das Ausführbare Modell mit dem Aufrufparameter „/p <Dateiname>“ aufrufen, wird versucht, diese Datei als Parameterdatei zu nutzen. Wird keine Parameterdatei gefunden, wird mit den in SimulationX eingestellten Werten gerechnet.

Um bestimmte Parameterbelegungen von einer Variantenrechnung mit einem ***Ausführbaren Modell*** auszuschließen, können in der Datei *parameters.txt* einzelne Zeilen mit einem vorangestellten doppelten Vorwärtsschrägstrich „//“ auskommentiert werden.

## Parallelisierte Variantenrechnung

Mit Hilfe von OpenMP ist es möglich, die Variantenrechnung durch Parallelisierung signifikant zu beschleunigen. Um eine parallelisierte Variantenrechnung durchführen zu können, ist es notwendig, dass der verwendete

Compiler „OpenMP“ unterstützt und der Rechner über mehrere logische Prozessorkerne verfügt.
Über die Einstellung „threadLimit“ in der Datei *solversettings.txt* kann festgelegt werden, wie viele logische Prozessorkerne verwendet werden sollen. Ist dieser Wert kleiner bzw. gleich 0, werden alle verfügbaren Prozessorkerne verwendet. Standardmäßig ist dieser Wert auf -1 gesetzt, so dass alle verfügbaren Prozessorkerne verwendet werden. Ist „threadLimit“ mit 1 belegt, findet keine Parallelisierung statt.

Achtung:
Bei der Verwendung von externen Funktionen, die **nicht** thread-sicher sind, kann es zu unerwarteten Ergebnissen kommen.

### 9.4.3.3 Simulink S-Function

Mit Hilfe dieses Projekttyps kann aus dem SimulationX Modell eine Simulink S-Function erzeugt werden. **Bild 9.50** zeigt das Vorgehen.

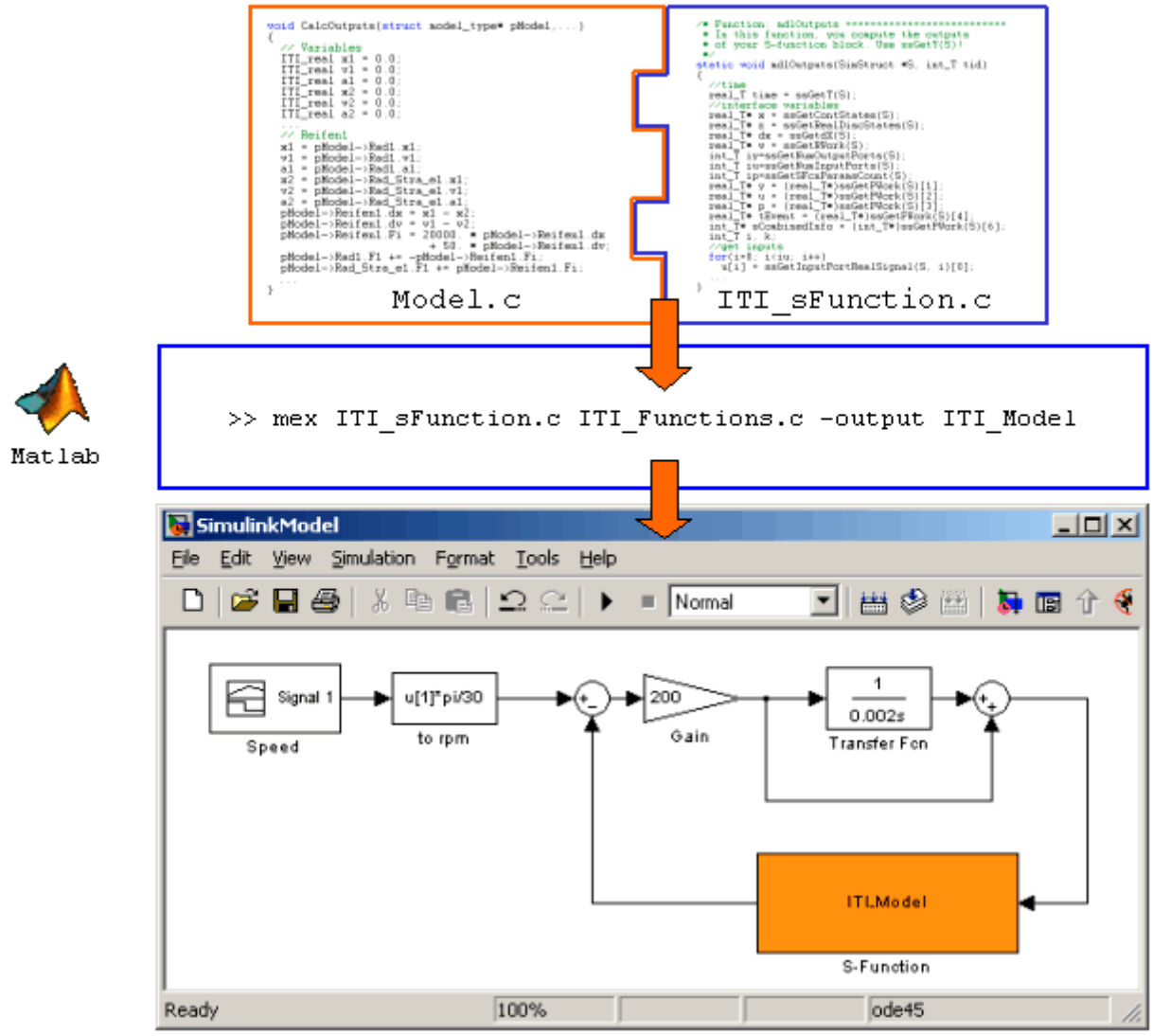



**Bild 9.50:** MATLAB/Simulink S-Function

Folgende Einstellungen müssen Sie vornehmen:

| Dialogseite | Beschreibung |
|---|---|
| Projekt | Beachten Sie, dass der Projektname beim Export als Simulink S-Function nicht mit einem Sonderzeichen oder einer Zahl beginnen darf. In der Auswahlbox „Typ“, wählen Sie aus, ob die S-Function mit oder ohne Solver generiert werden soll. |
| Eingänge | Die exportierten Eingänge entsprechen den Eingängen des S-Function Blockes. Die Reihenfolge entspricht der Reihenfolge im Auswahlfenster. |
| Ausgänge | Die exportierten Ausgänge entsprechen den Ausgängen des S-Function Blockes. Die Reihenfolge entspricht der Reihenfolge im Auswahlfenster. |
| Parameter | Die exportierten Parameter stehen als Parameter der Simulink S-Function zur Verfügung. |
| Nachbereitung | Wählen Sie hier Ihre MATLAB Version aus und erstellen Sie die S-Function. Es wird automatisch eine Maske generiert, die die Eingabe erleichtert. |

Im Code-Export-Assistenten kann auf der ersten Seite mit der Auswahlbox „Typ“ eingestellt werden, ob die S-Function ohne Solver oder mit dem Fixed-Step Solver generiert werden soll.

**S-Function ohne Solver**

Zur Berechnung der S-Function wird der Simulink-Solver verwendet. Die S-Function enthält kontinuierliche Zustandsgrößen. Einstellungen zur Schrittweitensteuerung und zur Genauigkeit müssen Sie selbst vornehmen.
Gegenüber der Co-Simulation (siehe **Kapitel 9.3.3**) ist diese Art der Modellintegration in die Simulink-Umgebung die effizientere und numerisch günstigere Variante.
Man sollte den S-Function Export ohne Solver verwenden, wenn das exportierte Modell mit einem weiteren kontinuierlichen Modellteil in Simulink verbunden werden soll. Durch das sonst mögliche Oversampling und die dadurch entstehenden Verzögerungen im Signalaustausch können falsche Ergebnisse entstehen. Das trifft auch zu, wenn in Simulink einer der Variable Step Size Solver verwendet wird.

**S-Function mit Fixed-Step Solver**

Bei dieser Variante enthält die S-Function neben der Modellfunktionalität auch noch den SimulationX Fixed-Step Solver. Die Berechnung findet unabhängig vom eingestellten Simulink-Solver statt. Die S-Function enthält keine für Simulink sichtbaren kontinuierlichen oder diskreten Zustandsgrößen oder Sample Times.

Das ermöglicht es, sie in *Triggered*, *Function-call* und *Enabled Subsystems* zu verwenden. Dadurch ist diese Art der S-Function insbesondere für Echtzeitanwendungen interessant. Die in der S-Function intern genutzte Integrationsschrittweite kann unabhängig von den Simulink-Schrittweiteneinstellungen festgelegt werden. Ist die interne Integrationsschrittweite kleiner als die Simulink-Schrittweite, führt der interne Fixed-Step Solver entsprechende Zwischenschritte aus. Damit lässt sich eine Überabtastung für besonders steife Echtzeitmodelle realisieren. Während der internen Zwischenschritte werden die Eingangssignale als konstant angesehen.

Wie oben erwähnt enthält die generierte S-Function keine für Simulink sichtbaren Abtastzeiten. Simulink ruft den Block deshalb zu Zeitpunkten auf, an denen es das für notwendig erachtet, wenn Sie einen der Solver mit variabler Schrittweite verwenden. Wollen Sie hier ein deterministisches Verhalten erreichen, sollten Sie die S-Function in ein *Triggered* oder *Function-call Subsystem* einbauen und dort selbst für eine feste Abtastrate sorgen. Um für eine genaue Synchronisierung zwischen der S-Function und dem umgebenden Simulink-Modell zu sorgen, sollte diese Abtastrate ein ganzzahliges Vielfaches der in der S-Function festgelegten Integrationsschrittweite sein.

Wird ein Simulink Fixed-Step Solver verwendet, erfolgt der Aufruf mit der dort gewählten Schrittweite. Das oben beschriebene Verfahren kann hier zur Realisierung von Überabtastungsverhalten verwendet werden.

Beim Einbinden in ein *Enabled Subsystem* sind einige Besonderheiten zu beachten. Wird das *Subsystem* deaktiviert, verharrt die S-Function in ihrem letzten berechneten Zustand. Wird es wieder aktiviert, rechnet sie von dort aus weiter. Allerdings wird die interne Simulationszeit der S-Function jetzt auf die Simulink-Simulationszeit gesetzt. Dadurch kann es bei zeitabhängigen Funktionen innerhalb der S-Function zu unerwarteten Ergebnissen kommen.

Das Integrationsverfahren *iMode* und die konstante Schrittweite *dtMin* der S-Function können im Blockdialog auf der Seite **Solver** gesetzt werden. Dabei kann das Integrationsverfahren zwischen *Euler Vorwärts*, *ITI Standard*, *Heun*, *RKF23* und *DOPRI5* gewählt werden. Als Standardverfahren ist *ITI Standard* eingestellt.

Die im Code-Export-Assistenten gewählten Modellparameter können auf der Seite **Parameters** im Blockdialog der S-Function gesetzt werden.

Man kann die S-Function mit der Option „Mit FixedStep Solver“exportieren, wenn die S-Function in Simulink lediglich mit einem zeitdiskreten Controller verbunden werden soll, oder die S-Function als solches (ohne weitere Simulink Komponenten) über den Real Time Workshop auf eine Echtzeitplattform portiert werden soll. Verglichen mit dem Simulink internen Solvern führt das oft zu Vorteilen in der Performance. Außerdem kann so (falls notwendig) Oversampling genutzt werden. Das heißt, die SimulationX S-Function arbeitet intern mit einer kleineren Integrationsschrittweite als das restliche Simulink Modell.

## Einbindung in Simulink

Um das exportierte Modell in MATLAB/Simulink zu nutzen, muss es vorher in MATLAB über den Befehl *mex* verarbeitet werden. Das können Sie auf der Dialogseite **Nachbereitung** (Schaltfläche „***Erstellen***“) aus dem Code-Export Assistenten heraus (falls MATLAB auf Ihrem Computer installiert ist) oder direkt in MATLAB vornehmen. Stellen Sie dazu das MATLAB-Arbeitsverzeichnis auf ihr Projektverzeichnis, in dem sich die Quelltexte befinden. Um Ihnen die Arbeit zu erleichtern, wird beim Export die M-Datei „*Build_s_Function.m“* erzeugt. In dieser Datei steht der auszuführende *mex*-Befehl mit den erforderlichen Parametern. Diese Datei können Sie direkt ausführen oder den Befehl aus der Datei kopieren und im MATLAB-Befehlsfenster ausführen. Beachten Sie, dass Sie beim erstmaligen Aufruf des *mex*-Befehls einige Angaben zur Kompiler-Auswahl vornehmen müssen. Nach jedem Code-Export des Modells muss der *mex*-Befehl erneut ausgeführt werden, um aus dem veränderten Quellcode den S-Function Block neu zu kompilieren.
Die S-Function generiert automatisch eine Eingabemaske mit den Namen, Standardwerten und Einheiten der Parameter. Die Beschriftung der Anschlüsse mit den Namen der entsprechenden Anschlüsse aus SimulationX wird ebenfalls durchgeführt. So werden Fehler bei Parametrierung und Verbinden des S-Function-Blocks vermieden.
Die übersetzte S-Function kann in einem Simulink-Modell über einen S-Function-Block verwendet werden. Sie können den Block entweder von Hand einfügen oder vom Assistenten (Seite **Nachbereitung**) generieren lassen, wenn auf Ihrem Computer eine MATLAB-Installation vorhanden ist.

## Automatische Erzeugung des S-Function-Blocks

Über die Schaltfläche „***Block erstellen***“ wird die Erzeugung des S-Function Blocks gestartet. Der Code-Export-Assistent öffnet oder erstellt das Simulink-Modell im Zielverzeichnis, erzeugt den S-Function Block und trägt den S-Function Namen ein. Um die Rechenfähigkeit des Modells zu garantieren, wird die übersetzte S-Function ebenfalls in das Zielverzeichnis kopiert.
Folgende Angaben sind für die Erzeugung notwendig:

| | |
|---|---|
| Zielverzeichnis | Definiert den Speicherort für das Simulink-Modell. Wenn das Verzeichnis noch nicht vorhanden ist, dann wird es erzeugt. |
| Modellname | Definiert den Namen des Zielmodells. Wenn das Modell im Zielverzeichnis noch nicht vorhanden ist, wird es erzeugt. |
| Blockname | Name des S-Function Blocks im Zielmodell. |

Beachten Sie, dass ein eventuell schon vorhandener Block mit dem gleichen Namen ersetzt wird.

## Manuelle Erzeugung des S-Function Blocks

Wenn sie den Block von Hand erstellen wollen, gehen Sie wie folgt vor:

Erzeugen Sie in Ihrem MATLAB/Simulink-Modell einen S-Function Block (Toolbox *User-Defined Functions*). Im Parameterdialog des Blocks geben Sie unter „*S-function name*“ den Projektnamen an. Achten Sie dabei auf Groß- und Kleinschreibung. Bei der ersten Übernahme der Änderungen erscheint unter Umständen eine Fehlermeldung bezüglich fehlender Parameter. Diese Meldung können Sie ignorieren, nach der Fehlermeldung wird das Feld „*S-function parameters*“ automatisch ausgefüllt.

Wenn Sie den Dialog verlassen, hat Ihr Block die Ein- und Ausgänge und die Funktionalität des exportierten SimulationX-Modells.

**Verwendung im Real-Time Workshop**

Die erzeugte S-Function kann auch vom Real-Time Workshop verarbeitet oder per Simulink Rapid Accelerator berechnet werden. Es müssen dazu die neben dem Modellquelltext notwendigen C-Dateien in den Buildprozess einbezogen werden. Dies passiert automatisch während der Initialisierung der S-Function. Das manuelle Hinzufügen durch den Benutzer ist daher nicht mehr länger nötig. Die notwendigen Dateinamen sind im Parameter „S-function modules“ eingetragen. Um zu sehen, welche Module hinzugefügt wurden, kann man den Parameter-Dialog der S-Function durch den Befehl „Look Under Mask“ im Kontextmenu des Blocks öffnen.
Für den Übergang nach dSPACE können Sie die Dateien auch im User-Make File unter USER_SRCS angeben.

### 9.4.3.4 SIMPACK

Die SimulationX-Option Code-Export für SIMPACK generiert aus einem SimulationX-Modell eine UFORCE-Routine für SIMPACK.

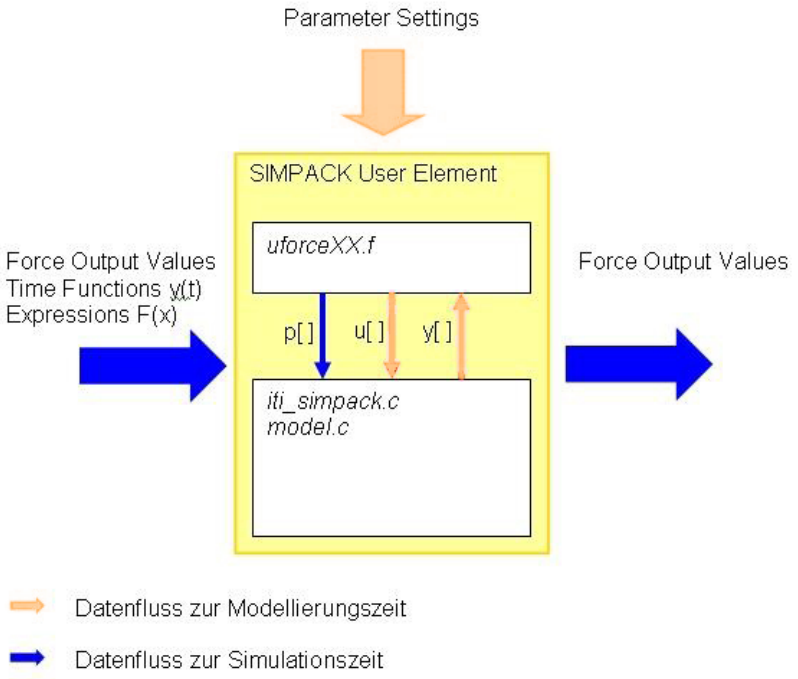
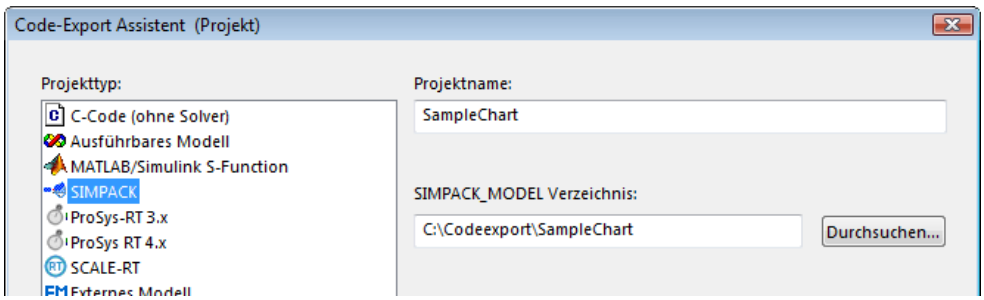



**Bild 9.51:** Code-Export nach SIMPACK

Folgende Einstellungen müssen Sie vornehmen:

| Dialogseite | Beschreibung |
|---|---|

Projekt:



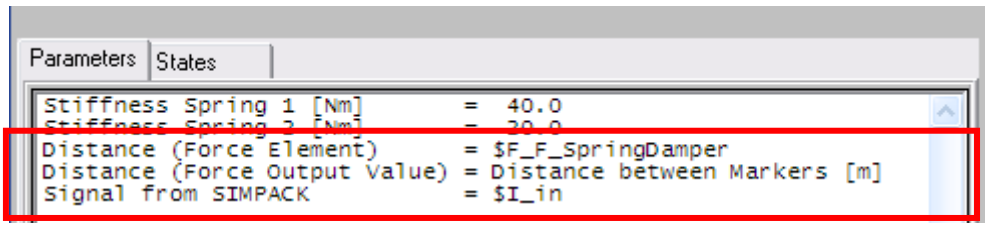

Als Projektverzeichnis für SIMPACK wird die Eingabe eines gültigen SIMPACK_MODEL-Verzeichnisses erwartet (z.B. C:\SIMPACKv8.8\user\SIMPACK.8800 bzw. e:\SIMPACK.8800).

UFORCE-Routine: Für die Einbindung des von SimulationX exportierten Modells in SIMPACK kann eine beliebige UFORCE-Routine gewählt werden.

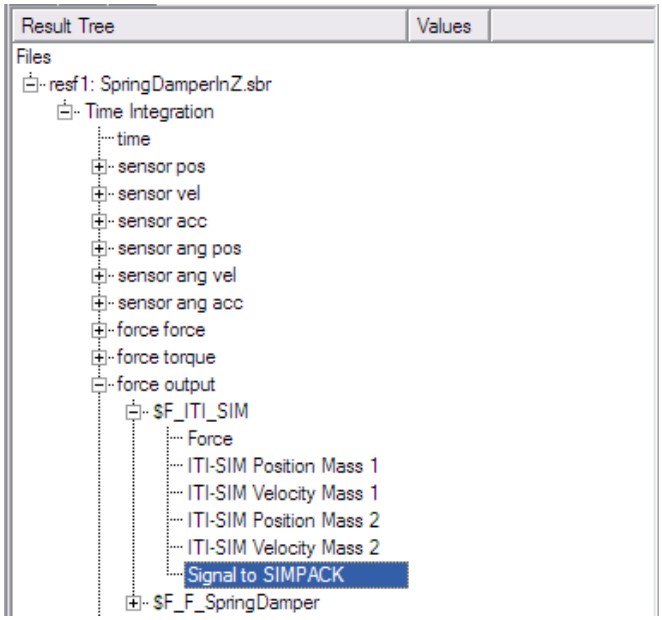
Signaleingänge: Bezeichnung, die in den Parameterdialogen von SIMPACK verwendet werden soll.
Parametertyp: Eingänge eines exportierten SimulationX-Modells können unter SIMPACK zur Laufzeit mit verschiedenen Größen belegt werden. Zur Auswahl stehen:

| | |
|---|---|
| Force Element | Wählbare Ausgabegröße eines beliebigen **Force Elements**. Für die Auswahl innerhalb von SIMPACK werden zwei Parameter benötigt (Force Element, Force Output Value). |
| Input Function y(t) | Wählbare **Input Function y(x)**, wobei x automatisch auf die Simulationszeit t gesetzt wird |
| Expression | Wählbarer Ausdruck (**Expression F(x)**) |
| Parameter | Eingabe eines Real-Wertes |

Die Auswahl der jeweiligen Elemente entsprechend des eingestellten Parametertyps erfolgt in den Parameterdialogen von SIMPACK.



Ausgänge: Bezeichnung, unter der die jeweilige Größe innerhalb von SIMPACK angezeigt werden soll.




Parameter: SIMPACK: Bezeichnung, unter der der jeweilige Parameter innerhalb der Parameterdialoge von SIMPACK angezeigt werden soll.
Bei Parametern, denen in SimulationX eine Maßeinheit zugeordnet ist, erfolgt die Eingabe von Werten unter SIMPACK stets in der jeweiligen SI-Einheit.
Parametertyp: Parameter eines exportierten SimulationX-Modells können unter SIMPACK zur Laufzeit mit verschiedenen Größen belegt werden. Zur Auswahl stehen:

| | |
|---|---|
| Force Element | Wählbare Ausgabegröße eines beliebigen **Force Elements**. Für die Auswahl innerhalb von SIMPACK werden zwei Parameter benötigt (Force Element, Force Output Value). |
| Input Function y(t) | Wählbare **Input Function y(x)**, wobei x automatisch auf die Simulationszeit t gesetzt wird. |
| Expression | Wählbarer Ausdruck (**Expression F(x)**) |
| Parameter | Eingabe eines REAL-Werts |

Die Auswahl der jeweiligen Elemente entsprechend des eingestellten Parametertyps erfolgt in den Parameterdialogen von SIMPACK.

Ausgehend vom gewählten Projektverzeichnis werden Dateien in folgenden Unterverzeichnissen generiert:

| | |
|---|---|
| .\bibl\uforce | Die FORTRAN-Quelldatei für die jeweilige UFORCE-Routine (z.B. uforce37.f) |
| .\src\ext\c | Alle übrigen Dateien |

Nach dem Export einer UFORCE-Routine sind in SIMPACK die folgenden Dateien neu zu übersetzen (XX steht für die Nummer der UFORCE-Routine):

uforceXX.f (MBS-Library: ForceElements)
iti_ufXX.c (External : C)
simulationx.c (External: C)

### 9.4.3.5 SCALE-RT

Mit diesem Projekttyp wird die komplette HiL (Hardware-in-the-Loop) Umgebung SCALE-RT ab Version 5 der Firma cosateq auf Linux RTAI Basis unterstützt. Alle notwendigen Komponenten werden beim Export erzeugt.

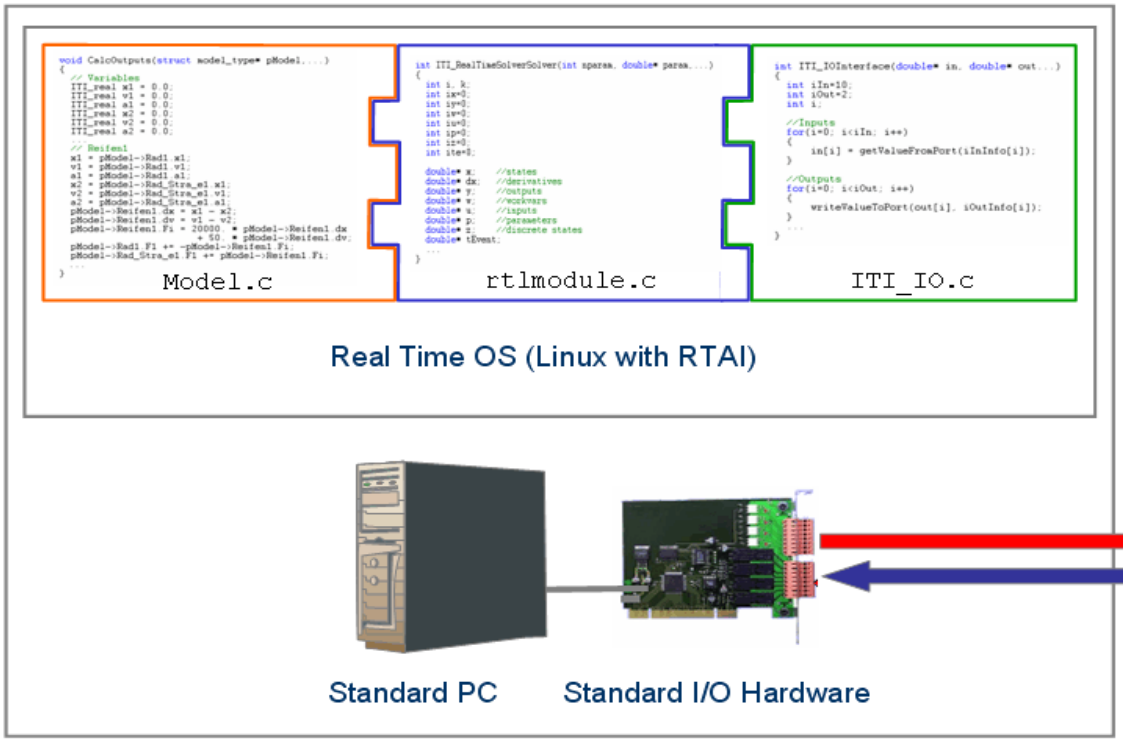



**Bild 9.52:** ProSys-RT

Die einzelnen Entwicklungsstufen mit Modellbildung, Simulation Code-Export, Realtime- Plattform, HiL, und Embedded Solutions werden vollständig und abgestimmt unterstützt. ITI ist gemeinsam mit cosateq in der Lage sämtliche Software- als auch Hardwarekomponenten (PC, I/O Karten) zu liefern.

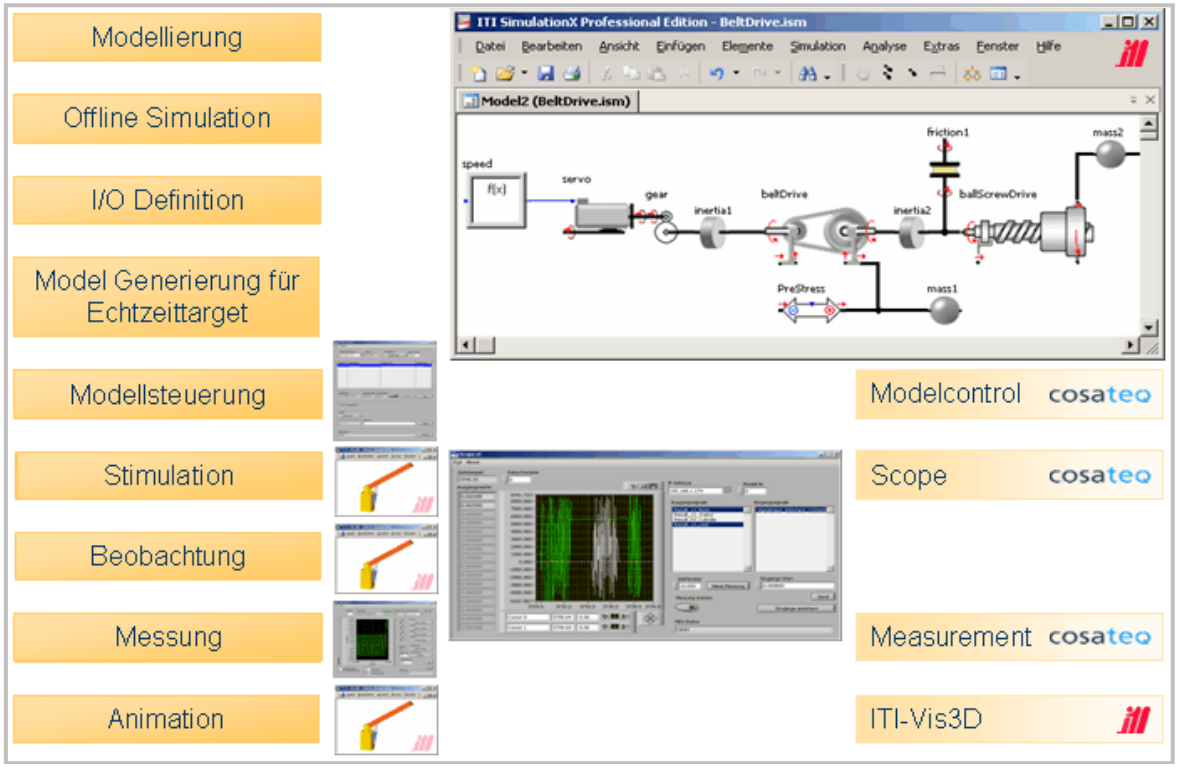



**Bild 9.53:** Workflow ProSys-RT

Folgende Einstellungen müssen Sie im Code-Export Assistenten vornehmen:

| Dialogseite | Beschreibung |
|---|---|
| Projekt | Auf dieser Seite werden der Projektname und das Projektverzeichnis festgelegt. |
| Eingänge | Legen Sie hier die Eingangsgrößen des exportierten Modells fest. Es sind nur Größen vom Typ Real zugelassen. |
| Ausgänge | Auf dieser Seite werden die Ausgangsgrößen festgelegt. Es sind nur Größen vom Typ Real zugelassen. |
| Parameter | Spezifizieren Sie auf dieser Seite die Parametergrößen. Es sind nur Größen vom Typ Real zugelassen. |
| Nachbereitung | Erstellen — Startet die Übersetzung des von Ihnen exportierten Projektes und erzeugt ein Archiv (*.tgz), das auf das Zielsystem kopiert werden kann. Im Ausgabebereich wird das Ergebnis der Übersetzung angezeigt. |

### 9.4.3.6 Externes Modell

Mit diesem Projekttyp ist es möglich die Modellfunktionalität als DLL zu exportieren, um diese über ein entsprechendes Interface anschließend in einem Simulator anzusprechen.

Folgende Einstellungen müssen Sie im Code-Export Assistenten vornehmen:

| Dialogseite | Beschreibung |
|---|---|
| Projekt | Auf dieser Seite werden der Projektname und das Projektverzeichnis festgelegt. |
| Eingänge | Legen Sie hier die Eingangsgrößen des exportierten Modells fest. |
| Ausgänge | Auf dieser Seite werden die Ausgangsgrößen festgelegt. |
| Parameter | Spezifizieren Sie auf dieser Seite die Parametergrößen. |
| Nachbereitung | Hier können Sie das exportierte Modell als DLL kompilieren. |

### 9.4.3.7 dSPACE DS1006

Mit diesem Projekttyp wird das *DS1006 Processor Board* der Firma dSPACE direkt unterstützt. Damit ist es möglich SimulationX-Modelle auf DS1006-basierter Prototyping-Hardware von dSPACE in Echtzeit zu rechnen.

Die Werkzeuge *MATLAB/Simulink/Stateflow* und *Real-Time Workshop/ Stateflow Coder* von *The MathWorks* bzw. die Simulink-Schnittstelle *Real-Time Interface* (RTI1006) von dSPACE werden nicht benötigt.

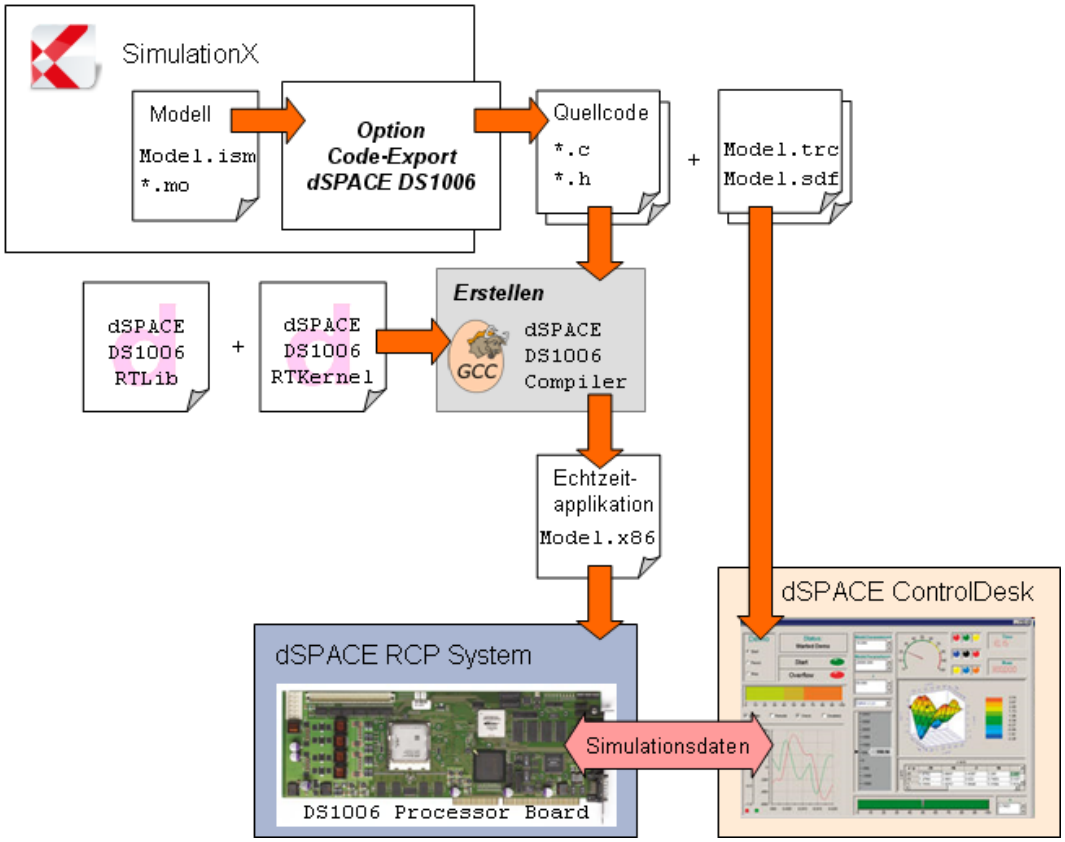



**Bild 9.54:** Ablauf Code-Export für dSPACE DS1006

Die folgenden Einschränkungen müssen berücksichtigt werden:

- Es werden nur DS1006 Single-Prozessor-Systeme unterstützt.
- Es existiert nur eine einzige zyklische Task „Timer Task“ mit Schrittweite dtProtMin.
- Mindestens *dSPACE Solutions for Control Release 6.0* wird für den Übersetzungsprozess benötigt.

Die folgenden dSPACE I/O-Karten werden (über eigene SimulationX-Typen) direkt unterstützt.

- DS4302 CAN Interface Board: CAN Receive und CAN Transmit

Folgende Einstellungen müssen Sie im Code-Export-Assistenten vornehmen:

| Dialogseite | Beschreibung |
|---|---|
| Projekt | Bestimmen Sie hier Projektname und –verzeichnis. |
| Eingänge | Wählen Sie die Eingänge, die Sie in dSPACE ControlDesk manipulieren und beobachten möchten. |
| Ausgänge | Wählen Sie die Ausgänge, die Sie in dSPACE ControlDesk beobachten möchten. |
| Parameter | Wählen Sie die Parameter, die Sie in dSPACE ControlDesk manipulieren und beobachten möchten. |
| Kennlinien | Wählen Sie die Kennlinien, die Sie in dSPACE ControlDesk manipulieren und beobachten möchten. |
| Task Handling | Beim *Initialen Simulationszustand* können Sie zwischen STOP, PAUSE und RUN wählen. Der Standardwert ist RUN.<br>Bei der Auswahl des *Simulationsendes* können Sie zwischen der Berücksichtigung von *tStop* oder einer Endlossimulation wählen. Der Standardwert ist Endlossimulation.<br>Das *Überlaufverhalten* kennt die Optionen *Stop Simulation, Overrun Queue* und *Overrun Count*.<br>Der Standardwert ist *Overrun Queue*.<br>*Overrun Stop*: Im Fall eines Überlaufs einer Task wird die Echtzeit-Simulation terminiert.<br>• *Overrun Queue*: Die Überläufe werden einer Warteschlange hinzugefügt. Übersteigt die Länge der Warteschlange die maximal erlaubte Überlaufanzahl, wird die Echtzeit-Simulation terminiert. |

- *Overrun Count*: Die Überläufe werden ohne Auswirkungen für die Echtzeit-Simulation gezählt.

Wenn *Overrun Queue* als Überlaufverhalten gewählt ist, so können Sie unter *Maximaler Überlaufanzahl* die maximale Länge der Warteschlange setzen. Der Standardwert beträgt 10.

Nachbereitung

Unter *dSPACE Umgebung* wird die aktive dSPACE-Umgebung für den Übersetzungsprozess angezeigt, falls *dSPACE Solutions for Control* installiert ist.

Unter *Zusätzliche Quellcode-Dateien* können Sie weitere Quelldateien, die sich im Projektverzeichnis befinden müssen, dem Übersetzungsprozess hinzufügen.

Unter *Zusätzliche Compiler-Optionen* können Sie weitere Optionen für den dSPACE DS1006 Compiler setzen. Als Standardwerte sind die Optionen „-DNDEBUG“, „-D_INLINE“ und „-O3“ gesetzt.

Startet die Übersetzung des von Ihnen erzeugten Projektes. Im Ausgabebereich wird das Ergebnis der Übersetzung angezeigt.

Lädt das übersetzte Modell auf das Zielsystem. Im Ausgabebereich wird das Ergebnis des Downloads angezeigt.

Der Standardwert für die minimale Rechenschrittweite *dtMin* ist nicht für Echtzeitanwendungen geeignet. Daher muss diese Schrittweite vor dem Code-Export in den allgemeinen Simulationseinstellungen angepasst werden.

Die minimale Ausgabeschrittweite *dtProtMin* sollte ein ganzzahliges Vielfaches der minimalen Rechenschrittweite *dtMin* sein.

### 9.4.3.8 NI VeriStand

Mit dem Projekttyp NI VeriStand wird die Echtzeit-Test- und Simulationssoftware NI VeriStand der Firma National Instruments direkt unterstützt. Damit ist es möglich SimulationX-Modelle in Echtzeit zu rechnen.

Die folgende Einschränkung muss berücksichtigt werden.

- Signale bzw. Variablen werden nicht unterstützt, d.h. nur die im Code-Export-Assistenten ausgewählten Parameter, Ein- und Ausgänge werden dem *Simulation Interface Toolkit* bereitgestellt.

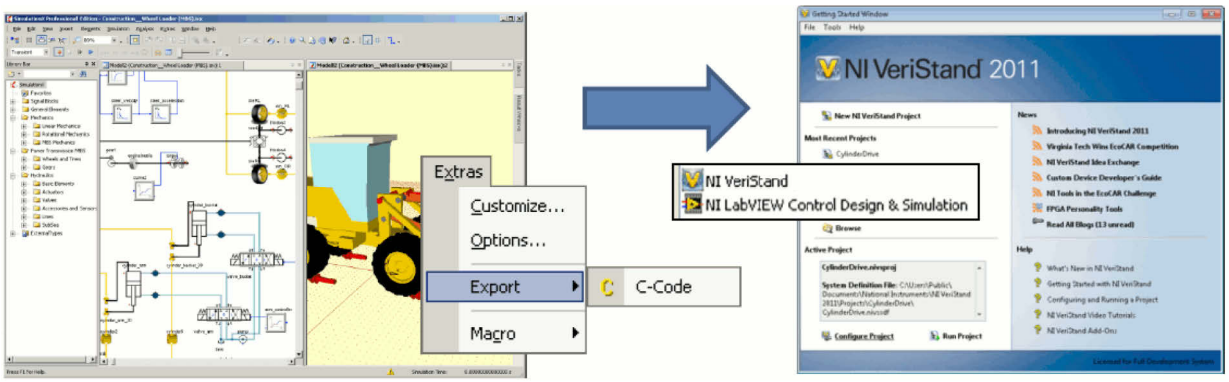



**Bild 9.55:** Code-Export nach NI VeriStand

Folgende Einstellungen müssen Sie im Code-Export-Assistenten vornehmen:

| Dialogseite | Beschreibung |
|---|---|
| Projekt | Bestimmen Sie hier Projektname und –verzeichnis. |
| Eingänge | Wählen Sie die Eingänge, die Sie im NI VeriStand Workspace manipulieren und beobachten möchten. |
| Ausgänge | Wählen Sie die Ausgänge, die Sie im NI VeriStand Workspace beobachten möchten. |
| Parameter | Wählen Sie die Parameter, die Sie im NI VeriStand Workspace beobachten möchten. |
| Nachbereitung | Auf dieser Dialogseite können Sie das generierte Projekt kompilieren und ausführen. |

Der Standardwert für die minimale Rechenschrittweite *dtMin* ist nicht für Echtzeitanwendungen geeignet. Daher muss diese Schrittweite vor dem Code-Export in den allgemeinen Simulationseinstellungen angepasst werden.

Die minimale Ausgabeschrittweite *dtProtMin* sollte ein ganzzahliges Vielfaches der minimalen Rechenschrittweite *dtMin* sein.

Für ein synchrones Zeitverhalten vom exportierten Modell und der Echtzeitumgebung müssen die minimale Ausgabeschrittweite *dtProtMin* und die *Primary Control Loop rate* von NI VeriStand übereinstimmen.

Der Standardwert für die *Primary Control Loop rate* von NI VeriStand ist 10000 µs und für PC-Tests geeignet. Für Tests auf Echtzeithardware sollte gegebenenfalls eine geringere Periode gewählt werden. Die *Primary Control Loop rate* (in Mikrosekunden) kann im NI System Explorer eingestellt werden, wenn der Controller-Knoten der Systemdefinition selektiert ist.

### 9.4.3.9 NI LabView Control Design & Simulation Module

Die Nutzung dieses Projekttyps ermöglicht die Integration von SimulationX-Modellen in NI LabVIEW. Dafür ist eine gültige Lizenz für das *NI LabView Control Design & Simulation Module* von National Instruments erforderlich.

Das exportierte Model wird mit Hilfe des NI External Model Interface in LabVIEW integriert. Dieses Interface hat derzeit folgende Einschränkungen:

- Nur eine diskrete Sample-Zeit wird unterstützt
- die Anzahl: Parameter + Eingänge + Ausgänge darf 25 nicht übersteigen

Wenn das exportierte Modell diese Einschränkungen überschreitet, erhalten Sie eine Fehlermeldung während der Initialisierung innerhalb LabVIEW.
LabVIEW unterstützt keine Ausgabe von Meldungen während der Simulation. Die Tracing-Einträge des Modells werden nur angezeigt, wenn ein Fehler auftritt und die Simulation stoppt.

LabVIEW ermöglicht skalare und vektorielle Parameter, Ein- und Ausgänge. Auch wenn SimulationX mehrdimensionale Werte erlaubt, werden diese Werte in Skalare umgewandelt.

Der Build-Prozess wurde unter Windows als dll getestet, nicht unter Mac OS, Linux oder auf Real-Time-Plattformen.

LabVIEW wird ab dem Release 2009 unterstützt.

Folgende Einstellungen müssen Sie im Code-Export-Assistenten vornehmen:

| Dialogseite | Beschreibung |
|---|---|
| Projekt | Bestimmen Sie hier Projektname und –verzeichnis. |
| Eingänge | Wählen Sie die Eingänge, die Sie im NI LabView VI manipulieren und beobachten möchten. |
| Ausgänge | Wählen Sie die Ausgänge, die Sie im NI LabView VI beobachten möchten. |
| Parameter | Wählen Sie die Parameter, auf die von NI LabView VI zugegriffen werden soll. Beachten Sie, dass die Parameter während der Simulation als konstant angenommen werden! Abhängig von der Nutzung der Parameter im SimulationX Modell ist es möglich, dass eine Parameteränderung nur während der Initialisierungsphase beachtet wird. Wenn Sie Parameter während der Laufzeit ändern wollen, sollten Sie stattdessen Eingänge verwenden. |
| Nachbereitung | Auf dieser Dialogseite können Sie das generierte Projekt kompilieren und ausführen. |
| | Erstellen — Startet die Übersetzung des von Ihnen erzeugten Projektes. Im Ausgabebereich wird das Ergebnis der Übersetzung angezeigt. |

Diese Schaltfläche ist nur aktiv, wenn *LabVIEW Control Design & Simulation Module*, sowie eines der Microsoft Developer Studios installiert sind.

Wenn das *LabVIEW Control Design & Simulation Module* nicht auf Ihrem Computer installiert sind, muss der generierte Source-Code manuell kompiliert werden. Die generierte Workspace-Datei (*.dsw) ist dann in das Microsoft Developer Studio zu laden. Diese Datei enthält alle notwendigen Projekteinstellungen. Das einzige, was sie noch tun müssen, ist die Datei „SIM_EMI_API.h“ dem Projekt bereitzustellen. Diese kommt mit den *LabVIEW Control Design & Simulation Module* und ist im LabVIEW-Verzeichnis unter „\CCodeGen\Simulation“ zu finden. Sie können dieses Verzeichnis als zusätzlichen Pfad zum Projekt hinzufügen oder die Datei in das Projektverzeichnis kopieren. Ergebnis eines erfolgreichen Erstellens ist die Datei <Projektname>.dll. Diese kann mittels *LabVIEW External Model function* in ein VI geladen werden. Diese Funktion muss in einer *Control & Simulation Loop* platziert werden.
**Bild 9.56** zeigt ein VI mit einem externen Modell aus SimulationX.
Bitte schlagen Sie für Details zum *Control Design & Simulation Module* in der Hilfe zu LabVIEW nach.

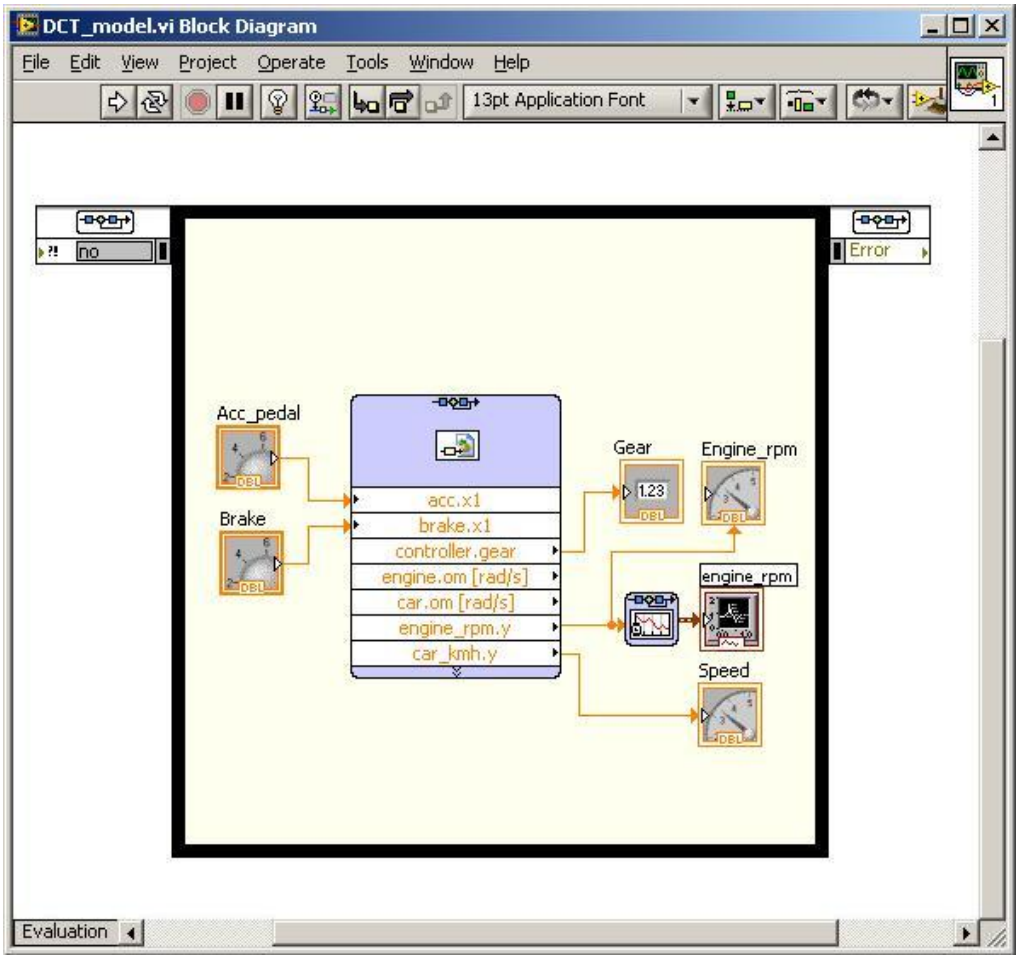



**Bild 9.56:** NI LabVIEW mit externem Model von SimulationX

### 9.4.3.10 Modelisar FMI für Model Exchange

FMI steht für „*Functional Mock-up Interface*“(http://www.functional-mockup-interface.org/index.html) und wurde innerhalb des MODELISAR Projektes spezifiziert. Ein wesentlicher Teil des FMI-Standards ist die Model Exchange – Spezifikation. Die Grundidee von *FMI für Model Exchange* ist es, Modellierungs-applikationen die Generierung von C-Code oder eines Binary auf Basis eines Modells zu ermöglichen, welches dann in anderen Simulationsumgebungen einfach integriert werden kann.

Die Verwendung dieses Projekttyps ermöglicht die Transferierung der Funktionalität eines kompletten SimulationX-Modells in eine FMU (*Functional Mock-up Unit*), die den FMI Standard implementiert. Andere Simulations-umgebungen können eine so erstellte FMU instanziieren und über die FMI Schnittstelle auf deren Funktionalität zugreifen.

Die folgende Abbildung zeigt schematisch den Ablauf eines *FMI* (*for Model Exchange*) Code-Exports.

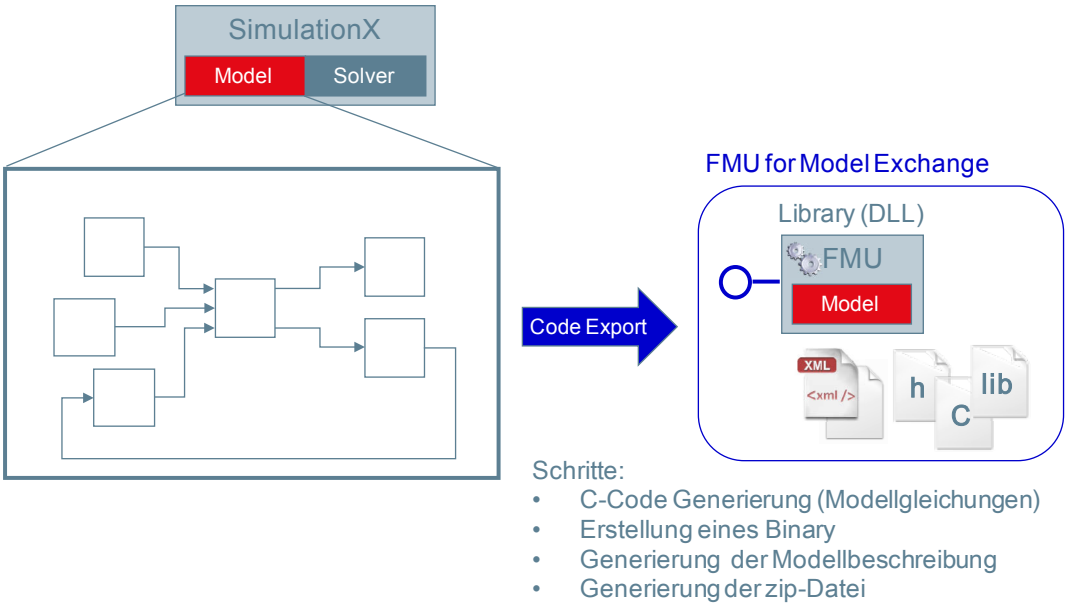


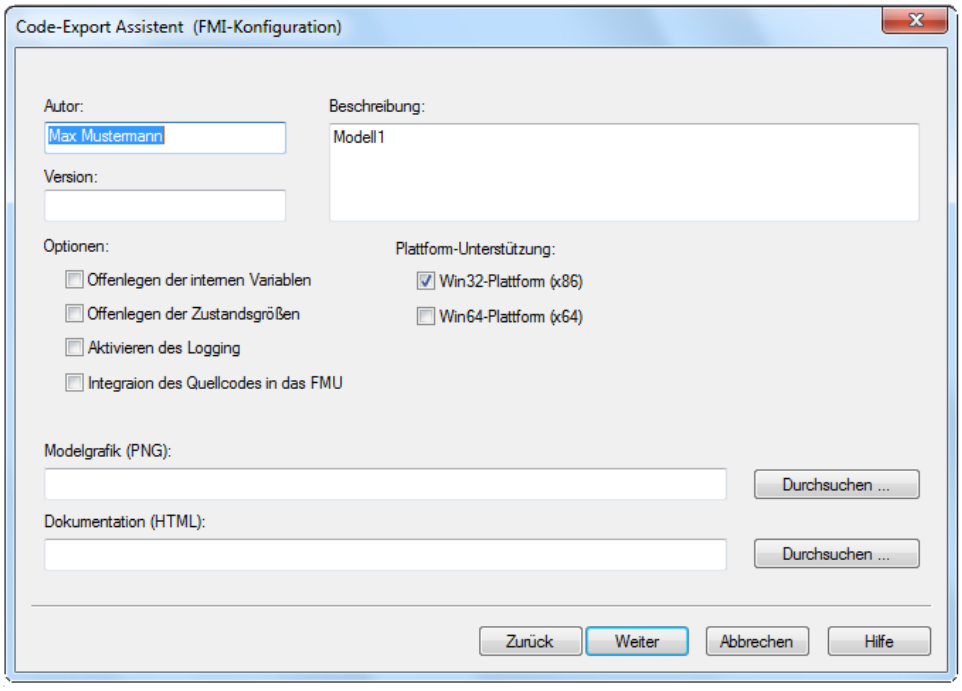

**Bild 9.57:** FMI Code-Export für Modelaustausch

Eine FMU wird in Form einer zip-Datei erstellt und besteht im Wesentlichen aus den folgenden Komponenten:

1. Exportiertes Modell + Schnittstelle
   Die exportierte Funktionalität des Modells ist über spezifizierte „C“ Funktionen (FMI) verfügbar. Durch die Verwendung der Programmier-sprache „C“ ist eine hohe Portierbarkeit garantiert. Diese Komponente kann als Quellcode und/oder als Binary z.B. DLL vorliegen.
2. *Model Description Schema*
   Dieses Schema liegt in Form einer XML-Datei vor, welche die Beschrei-bung aller nötigen Daten für den Informationsfluss zwischen der FMU und

des Simulationswerkzeugs beinhaltet. Durch die Modellbeschreibung in einer XML-Datei sind die Anbieter von Simulationswerkzeugen nicht auf eine bestimmte Datenrepräsentation festgelegt.

Folgende Einstellungen müssen Sie im Code-Export-Assistenten vornehmen:

| Dialogseite | Beschreibung |
|---|---|
| Projekt | Bestimmen Sie hier Projektname und –verzeichnis. Standardmäßig hat das Projekt den gleichen Namen wie das Modell. Während der Codegenerierung wird die Datei „generated.txt“ mit einer Liste der Namen aller automatisch generierten Dateien erzeugt. Wenn die Checkbox „Vorhandene Projektdateien löschen“ aktiviert ist, werden alle in dieser Datei aufgeführten Dateien vor der neuen Codegenerierung gelöscht. |
| FMI-Konfiguration |  |



| Option | Beschreibung |
|---|---|
| Autor | Name und/oder Organisation des Autors |
| Version | Version des Modells |
| Beschreibung | Kurze Beschreibung des Modells |
| Offenlegen der internen Variablen | Offenlegen der internen Variablen, welche nicht als Verbindungsvariablen in der XML-Beschreibungsdatei vorhanden sind. |

| | |
|---|---|
| Offenlegen der Zustandsgrößen | Offenlegen der Zustandsvariablen |
| Aktivieren des Logging | Kann zur Fehlersuche verwendet werden, wenn sich das Modell nicht wie gewünscht verhält |
| Integration des Quellcodes | Integration des für das Erstellen des exportierten Modells notwendigen C-Quellcodes in die FMU. |
| Plattform-Unterstützung | Integriert die erstellten DLLs für die ausgewählten Plattformen in die FMU |
| Modellgrafik | Grafik (PNG), die das Modell-Icon repräsentiert |
| Dokumentation | Verzeichnis, das die Dokumentation des Modells im HTML Format enthält. |

Eingänge — Wählen Sie auf dieser Seite die Eingangsvariablen. Die Werte dieser Variablen werden außerhalb des Modells definiert.

Ausgänge — Auf dieser Seite wählen Sie die Ausgangsvariablen. Diese Variablen werden zur Vernetzung von Modellen verwendet. Das bedeutet, dass Ausgangsvariablen in einer rufenden Funktion als Eingangsgrößen für andere FMUs oder Submodellen dienen können.

Parameter — Bestimmen Sie auf dieser Seite die Modellparameter. Die Werte dieser Variablen sind konstant nach der Initialisierung, können sich aber vor einer Simulation ändern.

Code-Generierung — Mit dem Aufruf dieser Seite des Assistenten startet die eigentliche Generierung der C-Quelltexte. Alle generierten Dateien, inklusive der Modellbeschreibungsdatei (XML), werden in dem angegebenen Projektverzeichnis gespeichert. Am Ende dieses Prozesses wird eine zip-Datei (FMU) erzeugt. Die Dateierweiterung dieser zip-Datei ist „.fmu“.

Nachbereitung

| | |
|---|---|
| Erweitert >> | Klicken Sie auf diese Schaltfläche um zusätzliche Einstellungen für die Nachbereitung festzulegen. Detaillierte Informationen zu diesen Einstellungen finden Sie in dem **Abschnitt 9.4.3.2**. |
| Erstellen | Kompiliert und erstellt Modell-DLLs entsprechend der von Ihnen auf der FMI-Konfigurationsseite eingestellten |

Plattformen. Die Ergebnisse dieses Prozesses werden im Ausgabebereich angezeigt. Anschließend werden die erzeugten DLLs in das FMU integriert.

### 9.4.3.11 Modelisar FMI für Co-Simulation

FMI steht für „*Functional Mock-up Interface*“(http://www.functional-mockup-interface.org/index.html) und wurde innerhalb des MODELISAR Projektes spezifiziert. Ein wesentlicher Teil des FMI-Standards ist die Co-Simulation – Spezifikation. *FMI für Co-Simulation* ist ein Schnittstellen-Standard für die Lösung von zeitabhängigen gekoppelten Systemen bestehend aus Teilsystemen, die kontinuierlich in der Zeit (Modellkomponenten, die durch instationäre Differentialgleichungen beschrieben werden) oder zeitdiskret (Modellkomponenten, die durch Differenzengleichungen beschrieben werden, wie z.B. diskrete Steuerungen) sind.
*FMI for Co-Simulation* definiert Interface-Routinen für die Kommunikation zwischen einem *Master* und den einzelnen Simulationswerkzeugen (*Slaves*) in der Co-Simulationsumgebung. Ein typischer Master-Algorithmus hält an jedem Kommunikationspunkt $tc_i$ die Integration über der Zeit aller Slaves an, sammelt die Ausgänge $y(tc_i)$ von allen Teilsystemen, evaluiert die Eingänge $u(tc_i)$ des Teilsystems, übergibt diese Eingänge an die Slaves und setzt die (Co-) Simulation mit dem nächsten Kommunikationsschritt $tc_i \rightarrow tc_{i+1} = tc_i + hc$ mit fester Kommunikationsschrittweite $hc$ fort. In jedem Slave wird ein geeigneter Solver verwendet, um eines der Teilsysteme für einen gegebenen Kommunikationsschritt $tc_i \rightarrow tc_{i+1}$ zu integrieren.

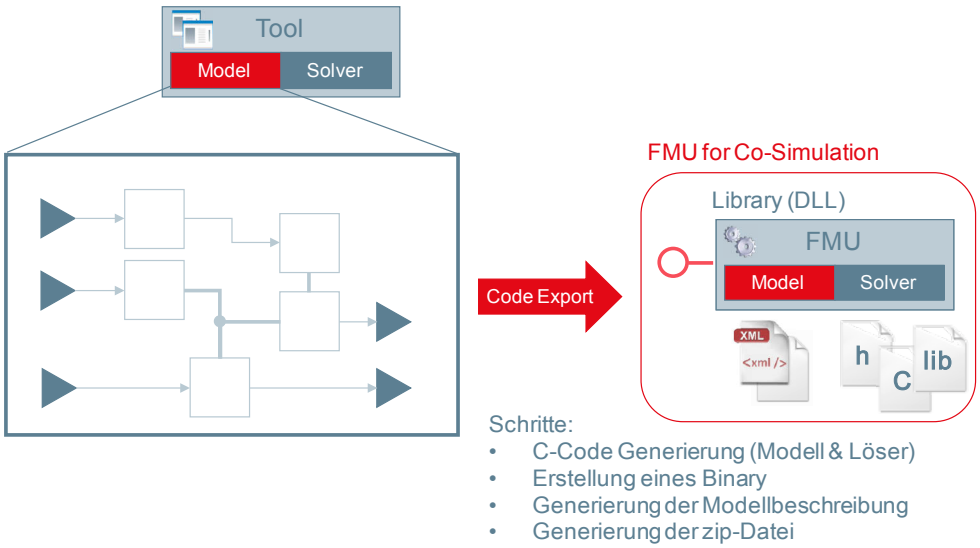



**Bild 9.58:** FMI Code-Export für Co-Simulation

Durch die Wahl des Projekttyps *Modelisar for Co-Simulation* im Code-Export Assistenten kann die komplette Modellfunktionalität in eine FMU (*Functional Mock-up Unit*) transferiert werden. Eine so erzeugte FMU enthält auch einen

Solver (CVODE) und kann als Slave-Simulator in einem Co-Simulation Umfeld verwendet werden.

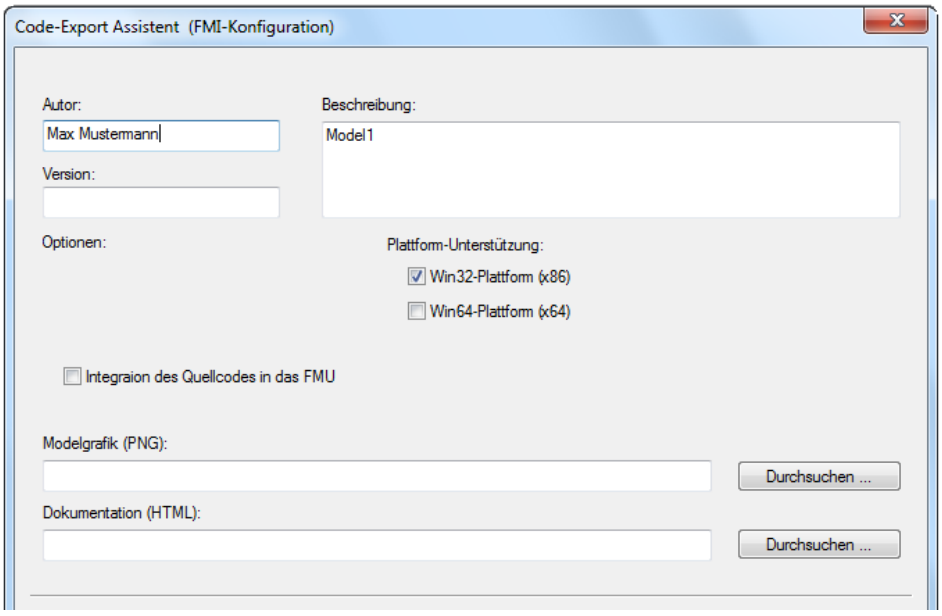
**Bild 9.58** zeigt schematisch den Ablauf eines FMI (for Co-Simulation) Code-Exports. Eine FMU wird in Form einer zip-Datei erstellt und besteht im Wesentlichen aus den folgenden Komponenten:

1. Exportiertes Modell + Löser (CVODE) + Schnittstelle

   Die exportierte Funktionalität des Modells ist über spezifizierte „C“ Funktionen (FMI) verfügbar. Durch die Verwendung der Programmiersprache „C“ ist eine hohe Portierbarkeit garantiert. Diese Komponente kann als Quellcode und/oder als Binary z.B. DLL vorliegen.

2. *Model Description Schema*

   Dieses Schema liegt in Form einer XML-Datei vor, welche die Beschreibung aller nötigen Daten für den Informationsfluss zwischen der FMU und des Simulationswerkzeugs beinhaltet. Durch die Modellbeschreibung in einer XML-Datei sind die Anbieter von Simulationswerkzeugen nicht auf eine bestimmte Datenrepräsentation festgelegt.

| Dialogseite | Beschreibung |
|---|---|
| Projekt | Bestimmen Sie hier Projektname und –verzeichnis. Standardmäßig hat das Projekt den gleichen Namen wie das Modell.<br>Während der Codegenerierung wird die Datei „generated.txt“ mit einer Liste der Namen aller automatisch generierten Dateien erzeugt. Wenn die Checkbox „Vorhandene Projektdateien löschen“ aktiviert ist, werden alle in dieser Datei aufgeführten Dateien vor der neuen Codegenerierung gelöscht. |
| FMI-Konfiguration |  |

| Option | Beschreibung |
|---|---|
| Autor | Name und/oder Organisation des Autors |
| Version | Version des Modells |
| Beschreibung | Kurze Beschreibung des Modells |
| Integration des Quellcodes | Integration des für das Erstellen des exportierten Modells notwendigen C-Quellcodes in die FMU. |
| Plattform-Unterstützung | Integriert die erstellten DLLs für die ausgewählten Plattformen in die FMU |
| Modellgrafik | Grafik (PNG), die das Modell-Icon repräsentiert |
| Dokumentation | Verzeichnis, das die Dokumentation des Modells im HTML Format enthält. |

Eingänge — Wählen Sie auf dieser Seite die Eingangsvariablen. Den Slaves werden die Werte dieser Eingangsgrößen vor der Simulation eines Teilintervalls übergeben.

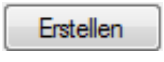
Ausgänge — Auf dieser Seite wählen Sie die Ausgangsvariablen. Nach der Simulation eines Teilintervalls werden die Werte dieser Ausgänge an den Master übertragen.

Parameter — Bestimmen Sie auf dieser Seite die Modellparameter. Die Werte dieser Variablen sind konstant nach der Initialisierung, können sich aber vor einer Simulation ändern.

Code-Generierung — Mit dem Aufruf dieser Seite des Assistenten startet die eigentliche Generierung der C-Quelltexte. Alle generierten Dateien, inklusive der Modellbeschreibungsdatei (XML), werden in dem angegebenen Projektverzeichnis gespeichert. Am Ende dieses Prozesses wird eine zip-Datei (FMU) erzeugt. Die Dateierweiterung dieser zip-Datei ist „.fmu“.

Nachbereitung

Klicken Sie auf diese Schaltfläche, um zusätzliche Einstellungen für die Nachbereitung festzulegen. Detaillierte Informationen zu diesen Einstellungen finden Sie in dem **Abschnitt 9.4.3.2**.




Kompiliert und erstellt Modell-DLLs entsprechend der von Ihnen auf der FMI-Konfigurationsseite eingestellten

Plattformen. Die Ergebnisse dieses Prozesses werden im Ausgabebereich angezeigt. Anschließend werden die erzeugten DLLs in das FMU integriert.

### 9.4.3.12 CarSim, BikeSim, TruckSim Component (VehicleSim)

VehicleSim ist eine Werkzeugsammlung für die effiziente Simulation von Autos, Trucks und Motorrädern der Firma Mechanical Simulation Corporation. Für die nahtlose Integration von SimulationX-Modellen in VehicleSim via Code Export wird das VehicleSim API (Application Programming Interface) verwendet.

SimulationX Modelle, die nach VehicleSim exportiert werden, enthalten einen Löser mit konstanter Schrittweite. Vor der Durchführung des Code-Exports ist es sinnvoll, in SimulationX zu testen, ob sich das Modell mit einem Löser konstanter Schrittweite berechnen lässt und welche Schrittweite verwendet werden soll. Die Schrittweite lässt sich in VehicleSim verändern.

Folgende Einstellungen müssen Sie im Code-Export-Assistenten vornehmen:

| Dialogseite | Beschreibung |
|---|---|
| Projekt | Selektieren Sie den Projekttyp „CarSim, BikeSim, TruckSim component“ und legen Sie den Projektnamen und das Projektverzeichnis fest. Standardmäßig hat das Projekt den gleichen Namen wie das Modell.<br><br>Während der Codegenerierung wird die Datei „generated.txt“ mit einer Liste der Namen aller automatisch generierten Dateien erzeugt. Wenn die Checkbox „**Vorhandene Projekt-dateien löschen**“ aktiviert ist, werden alle in dieser Datei aufgeführten Dateien vor der neuen Codegenerierung gelöscht. |
| Eingänge | Auf dieser Seite wählen Sie die Eingänge des SimulationX-Modells, die durch den einzugebenden VehicleSim Namen den entsprechenden VehicleSim Variablen zugeordnet werden können. Eingänge, die nicht einer VehicleSim Variable zugeordnet sind, werden während der Simulation auf NULL gesetzt. |
| Ausgänge | Auf dieser Seite wählen Sie die Ausgangsvariablen. Modellvariablen, die nur in VehicleSim beobachtet werden sollen, können ebenfalls ausgewählt werden. Wenn kein VehicleSim Name eingegeben wurde, wird der entsprechende SimulationX Name in die VehicleSim Datenbank eingefügt. VehicleSim unterstützt Zeichenketten mit einer maximalen |

Länge von 8 Zeichen mit ALL CAPS (Großbuchstaben). Daher wird der SimulationX Name abgeschnitten und zu ALL CAPS (Großbuchstaben) transformiert, bevor er der VehicleSim Datenbank hinzugefügt wird. Wenn ein eingegebener VehicleSim Name nicht in der VehicleSim Datenbank gefunden werden kann, wird die Variable mit diesem Namen hinzugefügt.

VehicleSim verwendet, genauso wie SimulationX, SI Einheiten für die Rechnung. Deshalb sind auch keinerlei Umrechnungen von Einheiten notwendig.

Parameter

Auf dieser Seite wählen Sie diejenigen Parameter aus, die in VehicleSim änderbar sein sollen. Auch hier kann ein VehicleSim Name eingegeben werden. Die Parameter behalten ihre ursprünglichen Werte aus SimulationX, wenn sie nicht in VehicleSim verändert werden.

Code-Generierung

Mit dem Aufruf dieser Seite des Assistenten startet nach der Globalen Symbolischen Analyse die eigentliche Generierung der C-Quelltexte. Alle generierten Dateien werden in dem angegebenen Projektverzeichnis gespeichert.

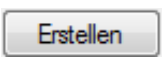
Nachbereitung

Wählen Sie hier die für das Erstellen zu verwendende Entwicklungsumgebung aus. SimulationX prüft automatisch, welche Microsoft C-Compiler auf dem Rechner installiert sind und zeigt diese an.

Informationen darüber, welche Entwicklungsumgebungen von SimulationX unterstützt werden, finden Sie im **Abschnitt 9.4.3.2.**

Das Ergebnis des Erstellens ist ein ausführbares Programm. Es enthält alle notwendigen Aufrufe der VehicleSim API-Funktionen, um die Ein-und Ausgänge zu verbinden und die erforderlichen Berechnungen durchzuführen.

Klicken Sie auf diese Schaltfläche, um zusätzliche Einstellungen für die Nachbereitung festzulegen. Detaillierte Informationen zu diesen Einstellungen finden Sie in dem **Abschnitt 9.4.3.2**.




Kompiliert und erstellt das von Ihnen erzeugte Projekt. Die Ergebnisse dieses Prozesses werden im Ausgabebereich angezeigt.

## Integration und Simulation in VehicleSim

Um das generierte ausführbare Programm in VehicleSim zu integrieren, muss es als „External Wrapper Programm“ in VehicleSim ausgewählt werden. Öffnen Sie den vorbereiteten VehicleSim Datensatz, indem Sie auf die SimulationX-Taste im Run-Control Fenster drücken.

Wählen Sie "External Wrapper Program" und die von SimulationX erzeugte ausführbare Datei, wie in **Bild 9.59** dargestellt, aus.

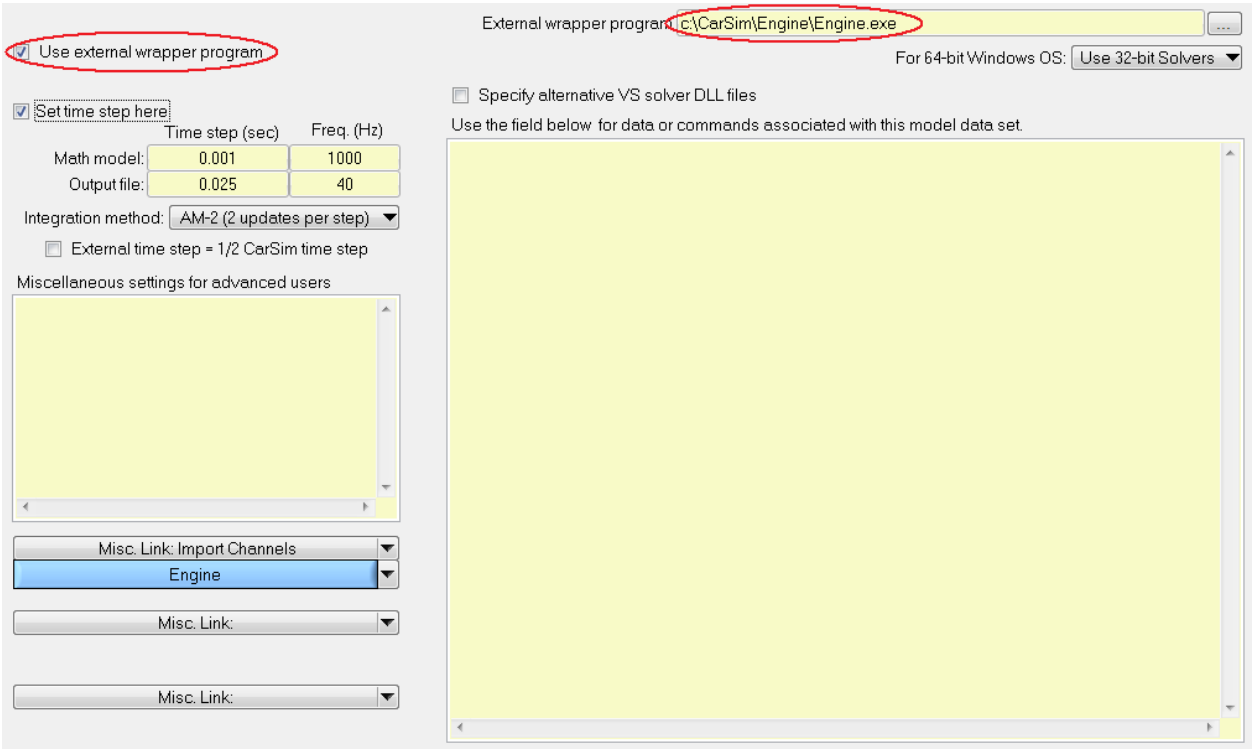



**Bild 9.59:** Auswählen der ausführbaren Datei als „External Wrapper Program“

Die Rechnung beginnt wie üblich mit einem Klick auf „***Run Math Model***“ auf der Hauptseite. Um einen Parameter zu ändern, geben Sie dessen Namen gefolgt von dem neuen Wert (in SI Einheit) im „Command Edit“ Feld ein.

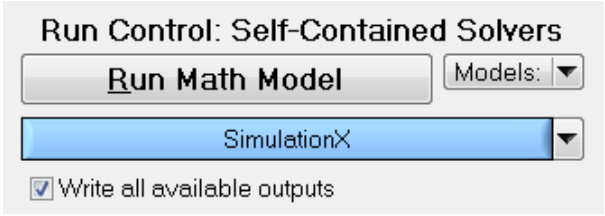



**Bild 9.60:** SimulationX Datensatz

Die SimulationX Solver-Parameter können hier ebenfalls geändert werden. Das generierte ausführbare Programm fügt die Integrations-Schrittweite (SIMX_DT) und die Integrationsmethode (SIMX_MODE) als Parameter der VehicleSim Datenbank hinzu. Sie können wie oben gezeigt geändert werden.

Die SimulationX Integrations-Schrittweite sollte kleiner oder gleich der VehicleSim Schrittweite sein. Die VehicleSim Schrittweite sollte ein Integer-Faktor der SimulationX Integrations-Schrittweite SIMX_DT sein. Dadurch kann das SimulationX Modell mit einer kleineren Schrittweite als der von VehicleSim verwendeten Schrittweite gerechnet werden.
Der Parameter SIMX_MODE bestimmt die verwendete Integrationsmethode wie folgt:

| SIMX_MODE | Integrationsmethode |
|---|---|
| 0 | Euler Forward |
| 1 | ITI Standard |
| 2 | Heun's Method |
| 3 | RKF23 |
| 4 | DOPRI5 |

Weitere Informationen finden Sie in dem ITI Tutorial 20 "Model Export to CarSim, BikeSim, TruckSim" (http://www.iti.de/fileadmin/Ressourcen/Tutorials/20_Code_Export_CarSim.pdf).

## 9.5 Code Import

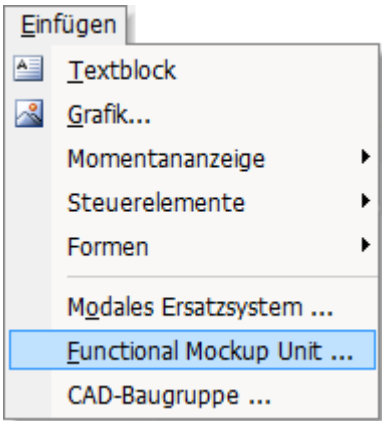
### 9.5.1 FMU Code Import

FMU - *Functional Mockup Unit*, ist eine ZIP-Datei mit der Endung ". FMU" entsprechend der FMI Definition, die alle Informationen zur Simulation eines Modells in einer Simulationsumgebung enthält (siehe **9.4.3.10)**.

Eine FMU besteht im Wesentlichen aus den folgenden Komponenten:

1. Exportiertes Modell + Schnittstelle + Löser
   Die exportierte Funktionalität des Modells ist über spezifizierte „C" Funktionen (FMI) verfügbar. Durch die Verwendung der Programmiersprache „C" ist eine hohe Portierbarkeit garantiert. Diese Komponente kann als Quellcode und/oder als Binary z.B. DLL vorliegen.
2. Model Description Schema
   Dieses Schema liegt in Form einer XML-Datei vor, welche die Beschreibung aller nötigen Daten für den Informationsfluss zwischen der FMU und des Simulationswerkzeugs beinhaltet. Durch die Modellbeschreibung in einer XML-Datei sind die Anbieter von Simulationswerkzeugen nicht auf eine bestimmte Datenrepräsentation festgelegt.

Um eine FMU in SimulationX zu importieren, wählen Sie über das Menü *Einfügen/Functional Mockup Unit…* aus.




Daraufhin öffnet sich folgender Dialog zum Importieren einer FMU.

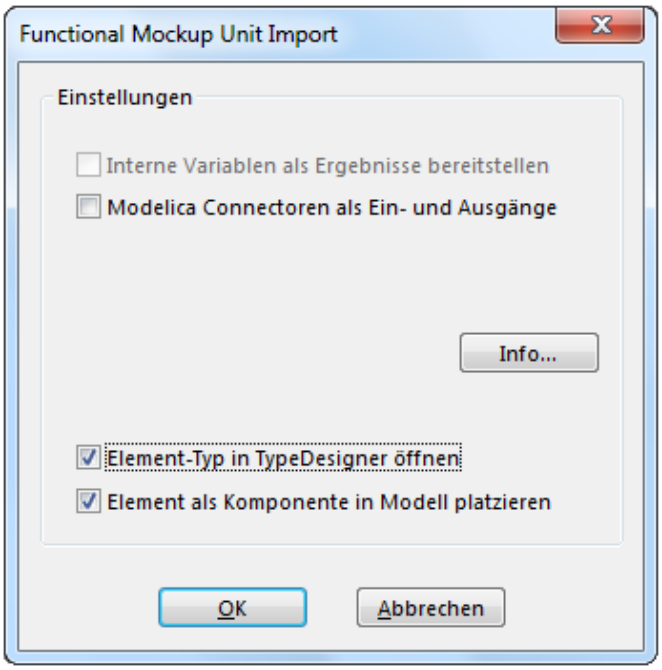



**Bild 9.61:** FMU Import Dialog

In diesem Dialog können Sie folgende Einstellungen vornehmen:

| Option | Beschreibung |
|---|---|
| Interne Variablen als Ergebnisse bereitstellen | Diese Option ist in dieser SimulationX Version noch nicht freigeschaltet. |
| Modelica Connectoren als Ein- und Ausgänge | Die in der FMU festgelegten Ein- und Ausgänge werden nicht als ITI-Connectoren (Signaleingänge bzw. Signalausgänge) sondern als Modelica-Connectoren angelegt. |

| | |
|---|---|
| Element-Typ in TypeDesigner öffnen | Der basierend auf der FMU und den vorgenommenen Einstellungen generierte Element-Typ wird automatisch im TypeDesigner geöffnet, um eventuelle Änderungen vorzunehmen. Diese Option ist standardmäßig ausgewählt. |
| Element als Komponente in Modell platzieren | Der erzeugte Element-Typ wird instanziiert und als Element im Modell platziert. Diese Option ist standardmäßig ausgewählt. |

Über die Schaltfläche ***Info…*** können Sie sich allgemeine Informationen der FMU anzeigen lassen.
Wenn Sie mit ***OK*** den Dialog verlassen, wird entsprechend der gewählten Einstellungen ein Element-Typ, basierend auf der importierten FMU, erzeugt und in das Modell eingefügt bzw. im TypeDesigner geöffnet.

## 9.5.2 Code Import aus Simulink/RTW

SimulationX kann DLLs einbinden, welche vom Real-Time Workshop® (RTW) aus Simulink-Modellen erzeugt wurden. Voraussetzung für den Import ist die Option *Code Import aus Simulink/RTW*. Im Lieferumfang ist das Element *RTWImport* in der Bibliothek *Simulatorkopplung*, ein angepasstes RTW Template und ein Beispielmodell enthalten.

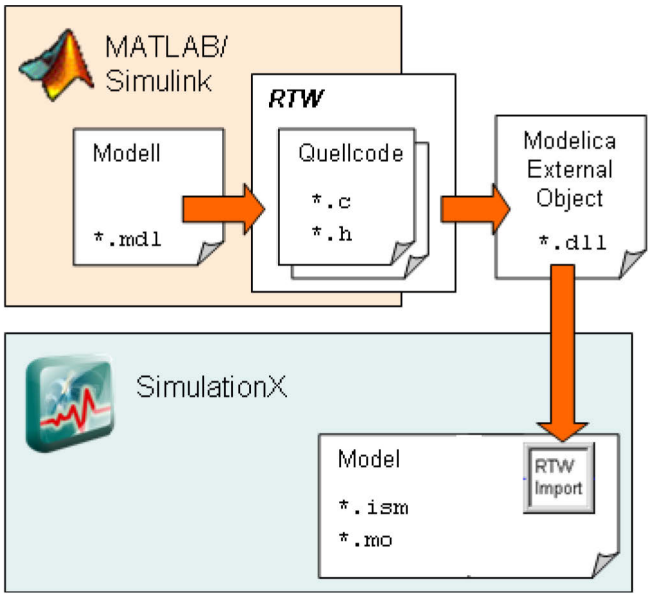



**Bild 9.62:** Ablauf Code Import aus RTW

Der Real-Time Workshop generiert aus dem Simulink-Modell eine DLL, die über ein *RTWImport* Element im SimulationX Modell verwendet werden kann. Die Simulationsrechnung läuft in SimulationX und der RTW DLL getrennt ab. Die

DLL enthält den beim RTW Codeexport eingestellten Löser mit fester Schrittweite. Der Datenaustausch erfolgt jeweils zu den Abtastzeitschritten.

### 9.5.2.1 Installation

Für den RTW-Export muss das RTW-SimulationX Target installiert werden. Die Installation erfolgt durch Kopieren des Verzeichnisses **simx** aus dem Samples Verzeichnis <Samples>**Co-Simulation\src\matlab\RTWImport** in das RTW Verzeichnis von Matlab (**<MATLABROOT>\rtw\c**). Danach ist SimulationX in der RTW-Target Auswahlliste verfügbar. Als Compiler wird der C-Compiler des Microsoft Visual Studio ab Version 2003 unterstützt.

### 9.5.2.2 Erzeugen und Verwenden der RTW-DLL

Die Kommunikation des Simulink-Modells mit SimulationX geschieht über externe Ein- und Ausgänge.
Alle Ein- und Ausgänge müssen skalare Größen von Typ Real sein.

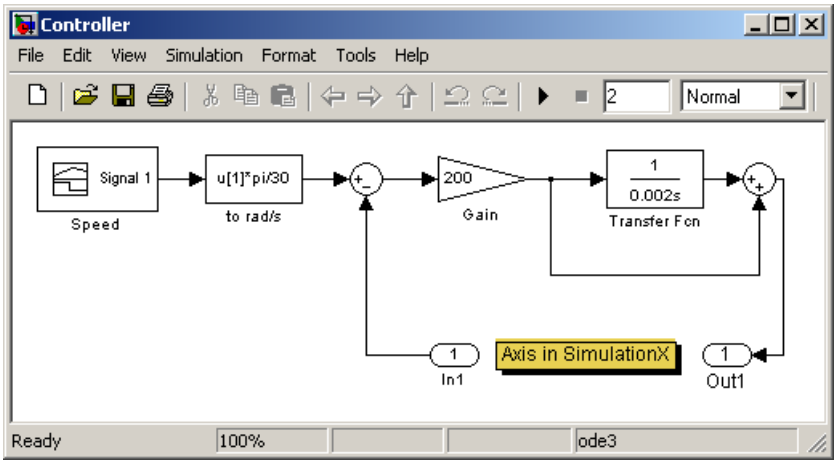



**Bild 9.63:** Simulink-Modell mit Schnittstelle zu SimulationX

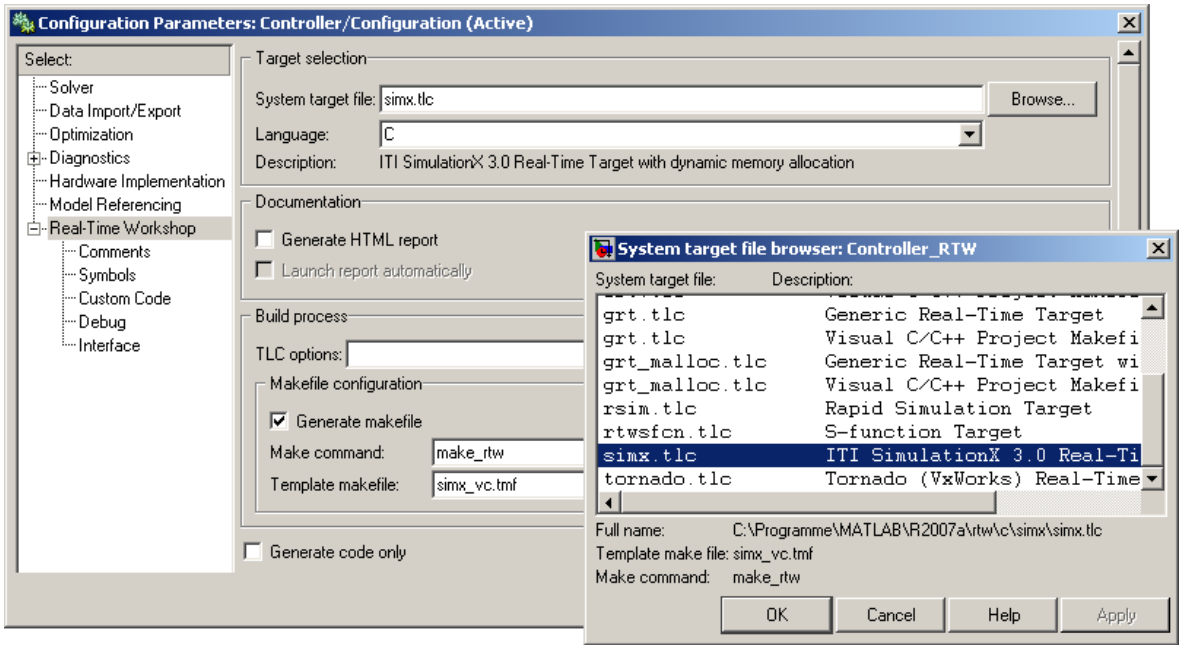



**Bild 9.64:** Auswahl des SimulationX Target File

Am Solver des Simulink-Modells müssen folgende Einstellungen vorgenommen werden:

| Schlüsselwort | Wert |
|---|---|
| StartTime | 0.0 (StopTime wird SimulationX festgelegt) |
| Type | Fixed-step |
| Fixed-step size | Kommunikationsschrittweite benötigt (Auto ist nicht erlaubt!) |
| Mode | Single Tasking |

Wenn im Realtime Workshop *simx* als System Target File ausgewählt wurde, dann können Sie die DLL erstellen.

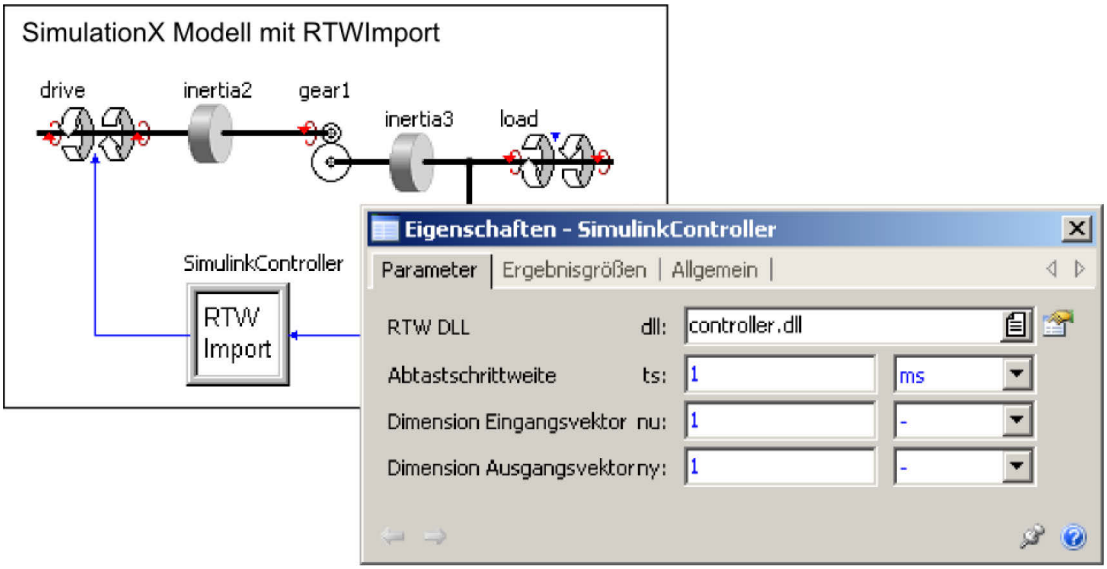



**Bild 9.65:** SimulationX Modell mit RTW Import

Das SimulationX Modell wird mit dem Modellobjekt *Realtime Workshop Import* aus der Bibliothek *Signalblöcke/Co-Simulation* aufgebaut. Unter RTW DLL kann die vom RTW generierte DLL ausgewählt werden. Die Dimension der Eingangs- und Ausgangsvektoren ist entsprechend der Ein- und Ausgänge im Simulink-Modell zu definieren.

## 9.6 Schnittstellen zu anderen Tools

### 9.6.1 Optimierung mit OptiY

Wenn das Optimierungsprogramm OptiY 4.0 von OptiY e.K. auf Ihrem PC installiert ist, so können Sie OptiY direkt über das Menü ***Extras/Tools/OptiY Optimierung…*** starten. Informationen zur Optimierung von SimulationX Modellen entnehmen Sie bitte der OptiY-Anwendungshilfe.

## 9.6.2 Optimierung mit Isight (optional)

Die SimulationX-Schnittstellenkomponete ermöglicht die Interaktion von Isight mit SimulationX. Die Hauptaufgaben dieser Komponente sind das Senden von Parametern zu SimulationX und das Empfangen skalarer Ergebnisgrößen von SimulationX für die Simulation im Zeitbereich.

Die folgenden Voraussetzungen müssen erfüllt sein:

- Isight ≥ Vs 3.5-1 muss installiert sein.
- SimulationX Vs 3.4 muss installiert sein.
- SimulationX Component Vs 2.0.4 muss installiert sein.

### 9.6.2.1 Komponenteneditor

1. Drag & Drop der SimulationX-Komponente in den Arbeitsbereich von Isight.
2. Ein Doppelklick auf das Komponentensymbol öffnet den Komponenteneditor. Die Dialogbox des Komponenteneditors erscheint (siehe **Bild 9.67**).

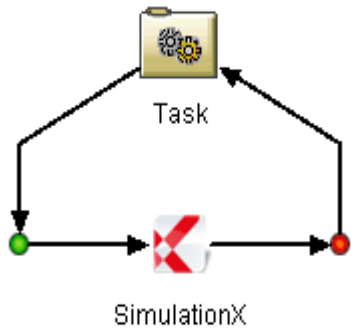



**Bild 9.66:** Isight-Arbeitsbereich mit SimulationX-Komponente

3. Durch klicken auf die Schaltfläche , gelangen Sie zum Datei-Öffnen Dialog.
4. Wählen Sie das SimulationX-Model oder eine -projektdatei aus, die Sie für den Datenaustausch nutzen wollen. Klicken Sie auf ***Öffnen***. Wenn SimulationX derzeit nicht geöffnet ist, wird eine unsichtbare Instanz von SimulationX erzeugt. Läuft SimulationX bereits, wird das Modell in SimulationX geöffnet.
5. Wählen Sie ein Element im Modellexplorer des Editordialoges der SimulationX-Komponente. Die Parameter und Ergebnisvariablen des gewählten Elements werden in Tabellen angezeigt.
6. Zum Wechsel des SimulationX-Modells können Sie die Schaltfläche erneut betätigen.
7. Klicken Sie auf die Schaltfläche , um die den Modellexplorer mit den Daten des aktuellen SimulationX-Modells zu aktualisieren.
8. Klick Sie auf OK, um alle Änderungen zu speichern und den Editor zu beenden.

### 9.6.2.2 Mapping von Parametern und Ergebnisgrößen

Das Mapping definiert, wie Isight Informationen an SimulationX leitet und von SimulationX empfängt.

1. Öffnen Sie den SimulationX-Komponenteneditor.
2. Wählen Sie den Wert eines Parameters oder einer Ergebnisgröße aus.
3. Der Name des Isight-Parameters kann in der gelb markierten Textbox editiert werden.
4. Ein Klick auf die Schaltfläche ➕ fügt das Mapping hinzu. Der entsprechende Isight-Parameter wird automatisch durch die SimulationX-Komponente erzeugt. Parameter aus SimulationX werden mit Isight-Eingabeparametern und Ergebnisgrößen aus SimulationX mit Isight-Ausgabeparametern identifiziert.
5. Sie können den Wert und den Datentyp der erzeugten Isight-Parameter in der Parameteranzeige ändern.
6. Sie können einzelne oder mehrere Mapping-Einträge durch Klicken der Schaltfläche ➖ von der Liste löschen.
7. Klicken Sie OK, um die Mapping-Informationen zu speichern.

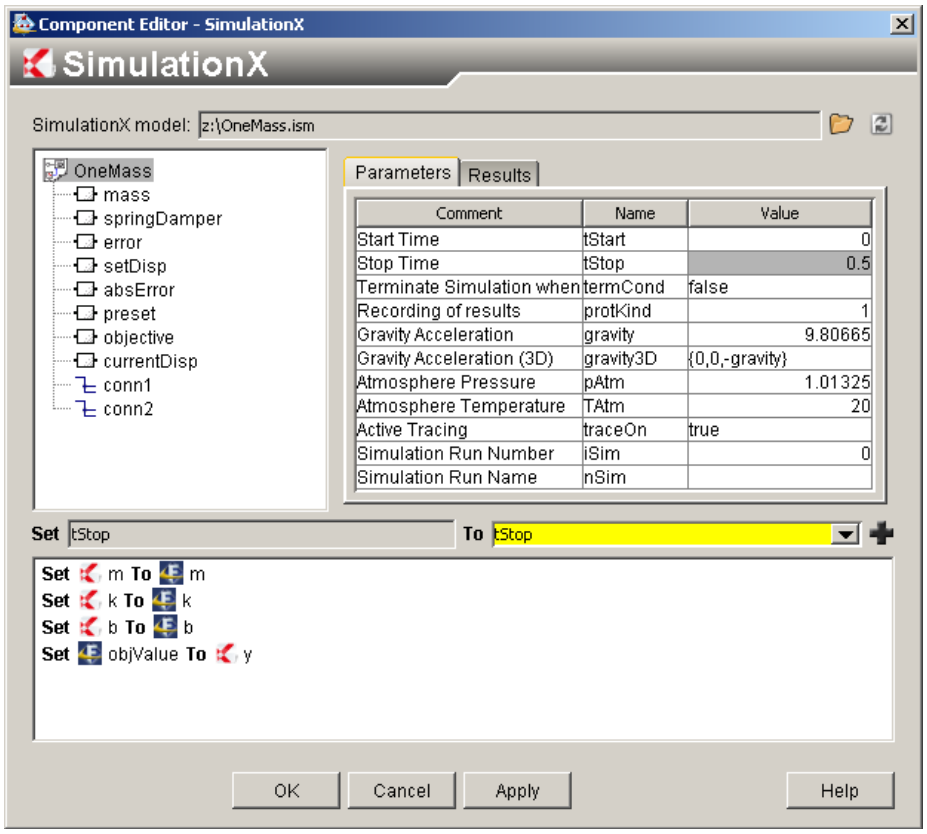



**Bild 9.67:** SimulationX-Komponenteneditordialog

## 9.6.3 Optimierung mit modeFRONTIER®

Ist das Optimierungsprogramm modeFRONTIER von Esteco s.r.l. auf Ihrem PC installiert ist, so können Sie von modeFRONTIER direkt auf SimulationX und das entsprechende Modell zugreifen. Alle notwendigen Einstellungen erfolgen in

modeFRONTIER. Informationen zur Optimierung von SimulationX Modellen entnehmen Sie bitte der modeFRONTIER-Onlinehilfe.

### 9.6.4 FEM-Import

Ist der SimulationX FEM-Import auf Ihrem Computer installiert, so können Sie diesen direkt über das Menü ***Extras/Tools/FEM-Import***... starten. Informationen zum FEM-Import finden Sie in der Dokumentation zum FEM-Import bzw. dem Bibliothekshandbuch zur Bibliothek Mechanik unter *Mechanics/Import Flexible Body*.

## 9.7 Datenbankanbindung

Dieses Kapitel beschreibt die Funktionsweise und Verwendung der Datenbankanbindung in SimulationX. Am Beispiel der Parametrierung eines Feder-Dämpfer-Elements werden die einzelnen Schritte erklärt.
Mit der Datenbankanbindung können OLEDB-Datenquellen (Object Linking and Embedding for Databases) zur Parametrierung von Modellen benutzt werden. Welche Datenquellen nutzbar sind, hängt von den auf dem Rechner installierten Daten-Treibern (z.B. Access, Excel, SQL) ab.

**Schema der Datenbankanbindung**

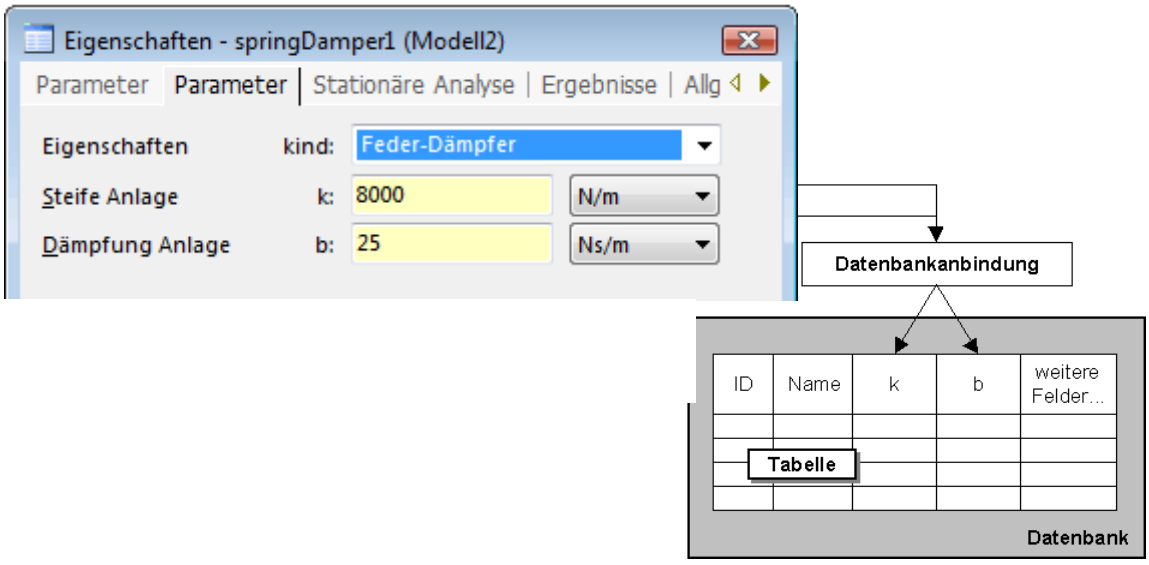



**Bild 9.68:** Schema der Datenbankanbindung

Die Datenbankanbindung kann an die Struktur der Datenquelle angepasst werden.

Die Tabellen enthalten die Daten der Objekte. Jede Spalte (Feld) entspricht einem Parameter, die Zeilen (Datensätze) enthalten den Wert der Parameter. Ein Objekt kann anhand der Werte in den Schlüsselfeldern eindeutig identifiziert werden.

Die Datenbankanbindung in SimulationX stellt die Zuordnung von Parametern in SimulationX und Spalten in der Tabelle her und kopiert die Daten aus der Datenbank in die Parameter des Elements.

Die Datenbankanbindung unterstützt folgende Komponenten:

- Skalare Parameter (Enumeration, Real, Integer, Boolean, String)
- Kennlinie
- Zeichenkette
- Kennliniensatz
- Kurvenschar
- Kennfeld (2D/3D)

### 9.7.1 Einfügen der Datenbankanbindung

Voraussetzung für die Definition der Datenbankanbindung ist eine Datenbankkomponente, die Sie im Typ Designer erzeugen (siehe **Bild 9.69**).

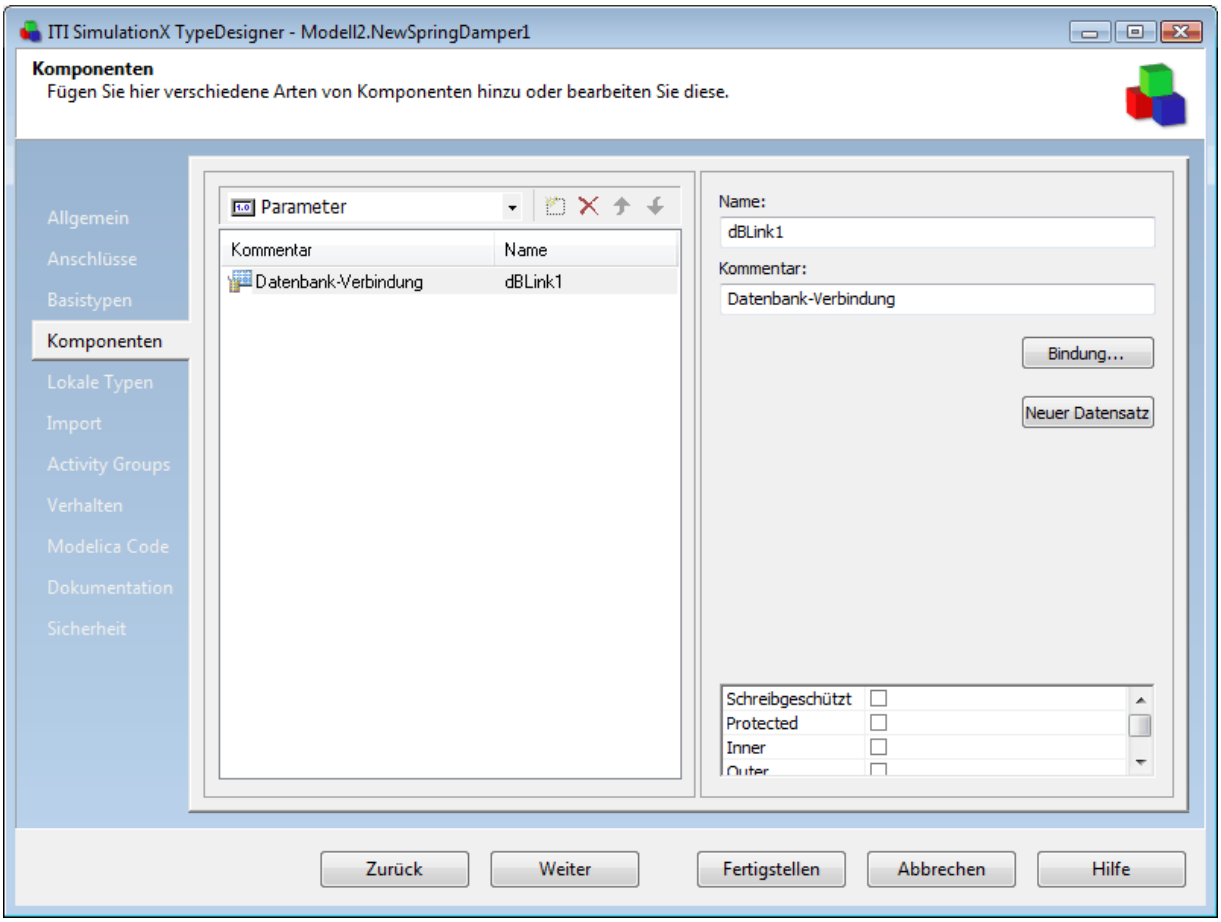



**Bild 9.69:** Erzeugen der Datenbankkomponente im TypeDesigner

**Beispiel:** Klicken Sie mit der rechten Maustaste auf das Symbol für ein Feder-Dämpfer-Spiel-Element und erstellen einen abgeleiteten Typ.

Der TypeDesigner wird geöffnet. Geben sie dem abgeleiteten Typ einen Namen (im Bsp. SpringDamperDB2). Anschließend gehen sie zur Seite Komponenten und fügen eine Datenbank-Verbindung hinzu (siehe auch **Kapitel 9.2.2.4** und **Bild 9.69**). Danach betätigen sie die Schaltfläche ***Fertigstellen***. Für die weitere Arbeit ziehen sie den abgeleiteten Typ in die Arbeitsfläche.
Die Definition der Datenbankanbindung wird in den nächsten Kapiteln anhand des Feder-Dämpfer Elements demonstriert.

Die Bindung mit der Datenbank kann im Typ Designer (Schaltfläche *Bindung...*) oder im Eigenschaftsfenster des Elements definiert werden.
Nach Öffnen des Eigenschaftsdialoges des abgeleiteten Typen ist unter Parameter ein Steuerelement für die Datenbank-Verbindung sichtbar. **Bild 9.71** zeigt den Eigenschaftsdialog, wobei noch keine Bindung  mit der Datenbank erfolgte.

Das Steuerelement zeigt Informationen über den ausgewählten Datensatz an. Sie können die Bindung definieren und die Parametrierung vornehmen .

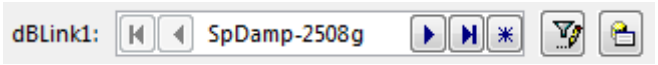



**Bild 9.70:** Steuerelement der Datenbankkomponente im Eigenschaftsdialog

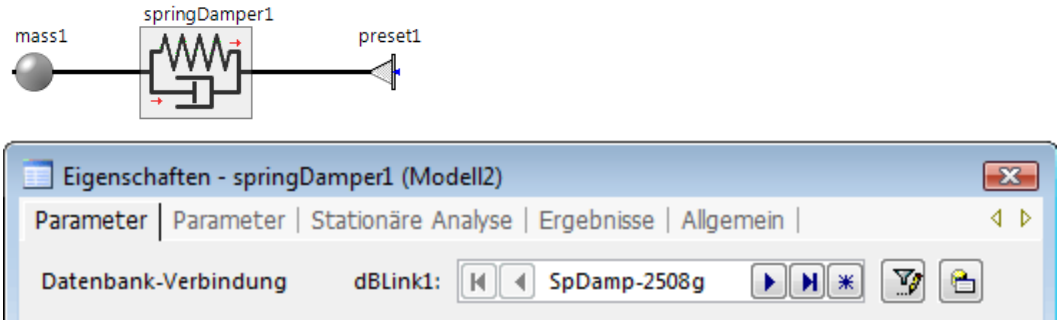



**Bild 9.71:** Feder-Dämpfer Element mit Datenbankanbindung

Die Definition der Datenbankanbindung wird in den nächsten Kapiteln anhand des Feder-Dämpfer Elements demonstriert. Sie können entweder eine passende Tabelle erzeugen lassen (siehe **Kapitel 9.7.2**) oder mit einer vorhandenen Datenbank verbinden (siehe **Kapitel 9.7.3**).

## 9.7.2 Automatische Erzeugung einer Datenbank

Der Assistent zur Datenbankanbindung bietet die Möglichkeit eine passende Datenquelle zu erzeugen und die entsprechenden Bindungen automatisch vorzunehmen.

Für die Datenbankerzeugung sind folgende Informationen notwendig

1. Typ der Datenquelle
   Standardmäßig werden die folgenden Datenquellen unterstützt:

- Access Datenbank
- Excel Arbeitsmappe
- SQL Server

Die Unterstützung weiterer Datenquellen kann auf Anfrage integriert werden.

2. Eigenschaften der Datenquelle

   Je nach ausgewählter Art der Datenquelle sind weitere Informationen für die Erzeugung notwendig (z.B. Name der Datenbank, Speicherort, Anmeldungsinformationen)

3. Tabellenname

   Geben Sie hier den Namen der Tabelle an welche die Daten enthalten soll. Wenn Sie den Namen einer existierenden Tabelle angeben, dann wird diese bei Bedarf erweitert.

4. Inhalt der Tabelle

   In der Liste am Ende des Dialogs können die zu speichernden Komponenten festlegen.

   Die erste Spalte enthält den Namen der Komponente und die zweite Spalte enthält den Namen der Spalte in der Tabelle. Standardmäßig ist der Spaltenname mit dem Komponentennamen identisch. Sie können den Spaltennamen aber verändern.

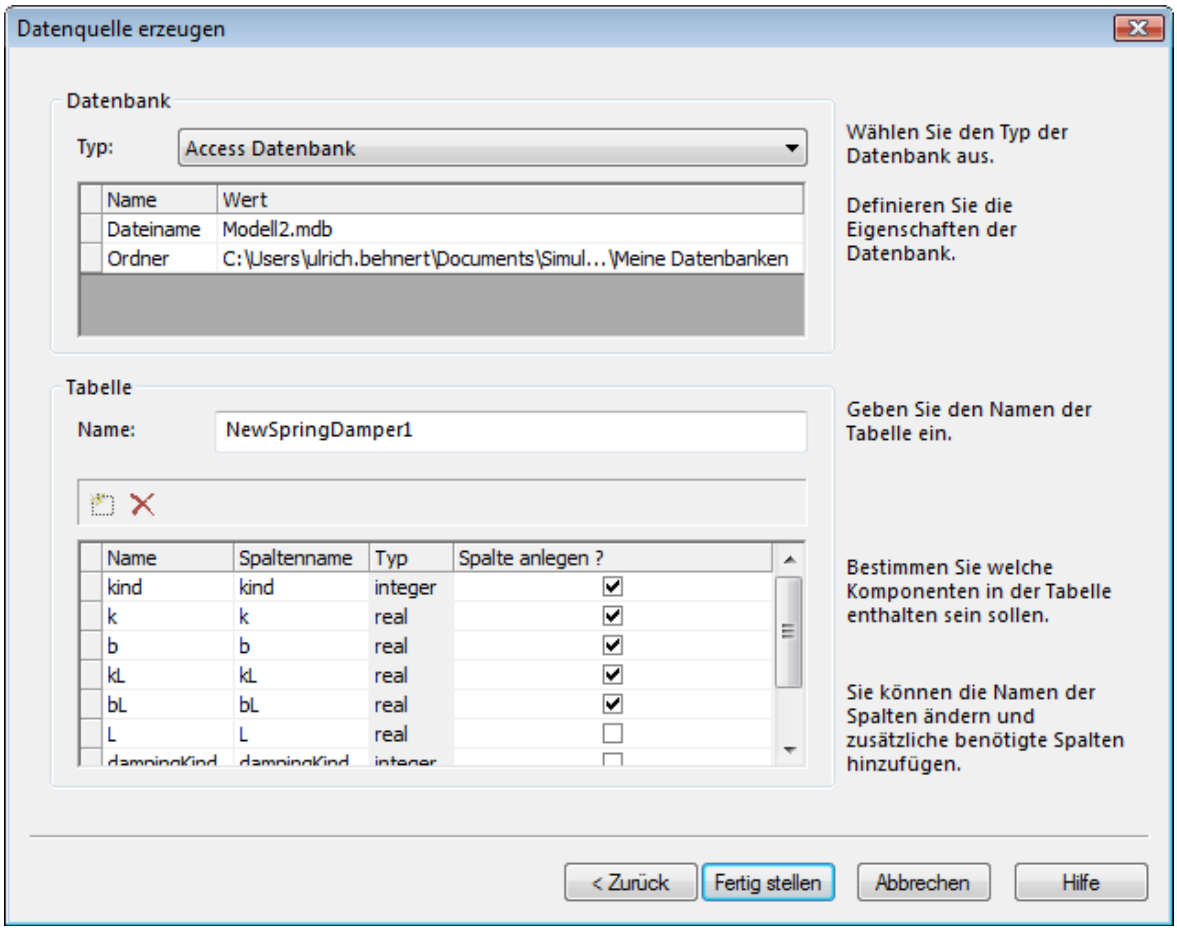



**Bild 9.72:** Datenquelle erzeugen

Der Datentyp der Spalte ergibt sich aus dem Typ der Komponente und kann nicht verändert werden. In der letzten Spalte können Sie entscheiden, ob Daten der Komponente in der Tabelle gespeichert werden sollen.

Mit Hilfe der Schaltflächen können Sie zusätzliche Spalten anlegen. Diese können Sie durch Eingabe eines Komponentennamens einer Komponente zuordnen. Der Typ von Spalten ohne Bindung kann beliebig gewählt werden. Zusätzlich zu den definieren Spalten wird eine Schlüsselspalte mit dem Namen ID angelegt. Durch die Schlüsselspalte kann jede Zeile der Tabelle eindeutig identifiziert werden (siehe auch **Kapitel 9.7.3.2**).
Mit der Schaltfläche ***Fertigstellen*** wird die Tabelle in der angegebenen Datenbank erzeugt (bzw. erweitert) und ein Datensatz mit den aktuellen Werten der Komponenten angelegt.

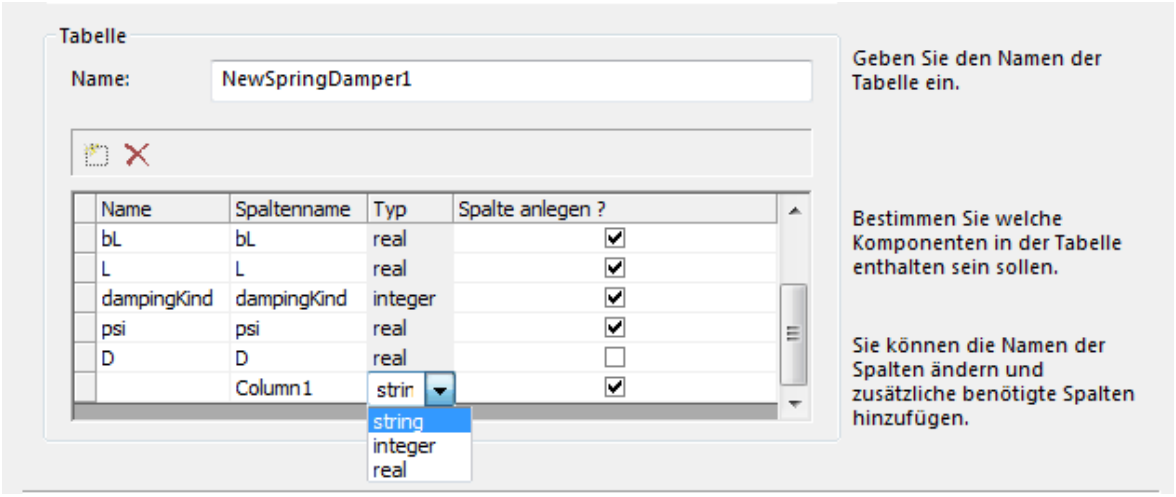



**Bild 9.73:** Inhalt der Tabelle

Die Datenbankanbindung speichert die Zuordnungen (Bindungen) der Spalten zu den Komponenten und der neue Datensatz wird ausgewählt. Die erzeugten Bindungen können nachträglich verändert werden.

Für Kennlinien, Kennliniensätze, Kurvenscharen und Kennfelder werden entsprechende untergeordnete Tabellen angelegt. Das entsprechende Schema wird in **Kapitel 9.7.6** beschrieben.

## 9.7.3 Verbinden mit vorhandener Datenbank

Das Verbinden der Datenbankanbindung mit einer existierenden Datenbank erfolgt mit Hilfe eines Assistenten und beinhaltet die folgenden Schritte:

- Datenbankauswahl
- Tabellenauswahl
- Binden der Parameter
- Parametrierung

Als Datenquelle soll eine Tabelle („SpringDampers“) in einer Access Datenbank (Components.mdb) dienen. Diese ist in ihrem SimulationX-Verzeichnis unter „...*Samples**Mechanics**Components.mdb*“ verfügbar.

### 9.7.3.1 Auswahl Datenbank

Die Verbindung zur Datenbank wird durch eine Zeichenkette beschrieben. Sie enthält Informationen über die Datenquelle und Anmeldung.
Diese Zeichenkette können Sie entweder mit Hilfe eines Assistenten erzeugen und bearbeiten oder bereits definierte Verbindungen aus einer UDL Datei laden.

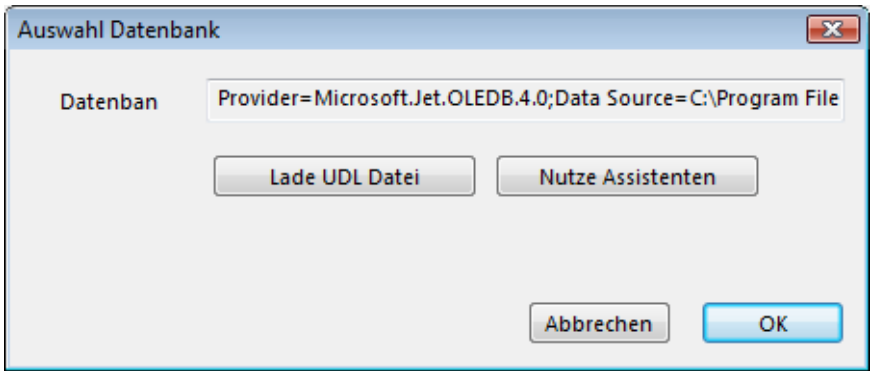



**Bild 9.74:** Verbindungszeichenkette einer Datenbank

Im Beispiel werden die Verbindungsinformationen mit Hilfe des Assistenten erzeugt.

Verwendung einer UDL Datei

Über die Schaltfläche ***Lade UDL Datei*** können sie die entsprechende Datei auswählen. Danach können Sie entscheiden, ob die Verbindungszeichenkette aus der UDL Datei in die Datenbankschnittstelle übertragen werden soll (siehe **Bild 9.74**) oder ob der Dateiname der UDL gespeichert werden soll (siehe **Bild 9.75**).

**Bild 9.75:** Verwendung UDL Datei

Wenn der Dateiname der UDL Datei verwendet wird, dann wird beim Öffnen der Datenbank immer der aktuelle Inhalt der UDL Datei als Verbindungszeichenkette benutzt. Durch Änderung der UDL Datei können Sie die Verbindungsinformation leicht nachträglich ändern.

Nutzung des Assistenten

Mit Hilfe des Assistenten können Sie eine neue Verbindungszeichenkette erzeugen bzw. die vorhandene Zeichenkette bearbeiten.

Auf der Registerkarte *Verbindung* können Sie Informationen über Datenbank und Anmeldung eingeben. Die Eingabemaske ist abhängig vom gewählten Provider. Die Richtigkeit Ihrer Eingaben können Sie durch Betätigen der Schaltfläche ***Verbindung testen*** prüfen (**Bild 9.77**).
Wenn Sie das Speichern des Kennworts nicht zulassen, dann werden Sie bei Verwendung der Datenbankanbindung zur Eingabe des Kennworts aufgefordert.
In den Registerkarten *Erweitert* und *Alle* können Sie je nach Provider zusätzliche Einstellungen vornehmen.

**Beispiel:** Öffnen sie den Dialog für die Bindung mit der Datenbank. Wählen sie den entsprechenden Provider (für das Bsp. siehe **Bild 9.76**).

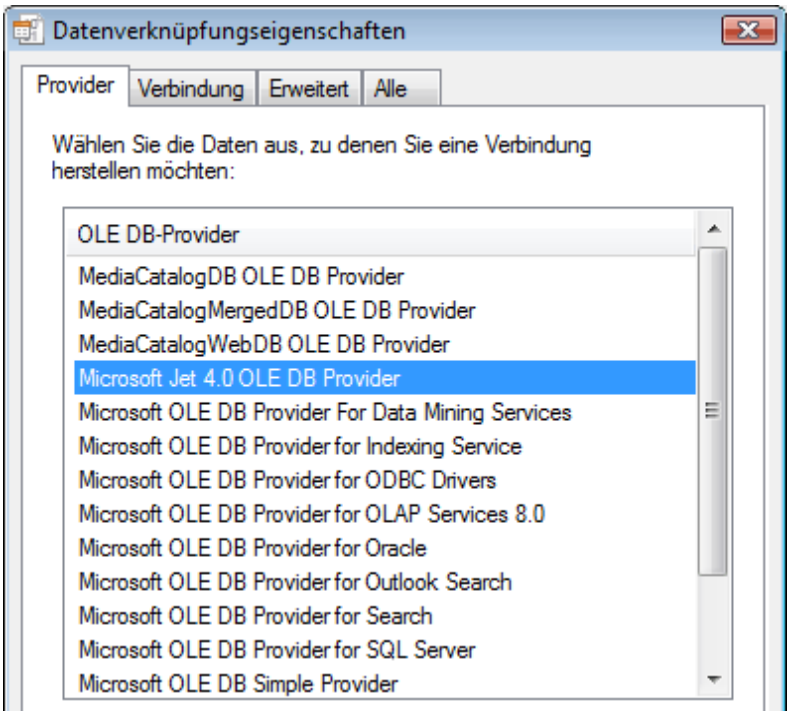



**Bild 9.76:** Auswahl OLEDB-Provider

Danach stellen sie unter Verbindung die vorhandene Datenbank „Components.mdb“ und die Anmeldedaten (Benutzer „Admin“ und kein Passwort) ein. Jetzt können sie die Verbindung testen (siehe **Bild 9.77**).

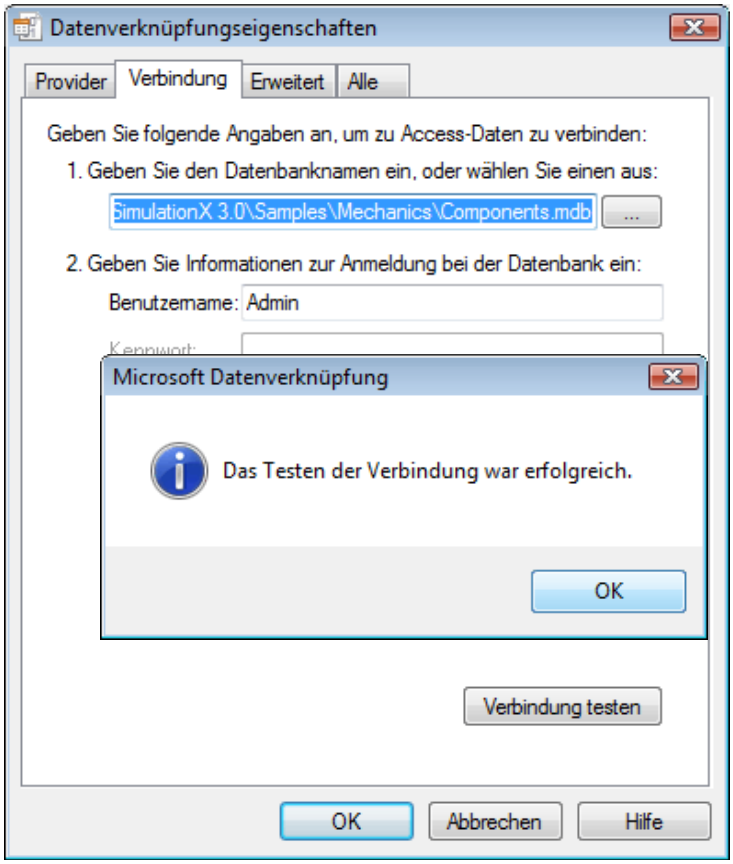



**Bild 9.77:** Test der Datenbankverbindung

### 9.7.3.2 Auswahl Tabelle

Es können Tabellen und Ansichten (Abfragen) der Datenbank als Tabelle verwendet werden.

Nachdem eine Verbindung zur Datenbank hergestellt wurde, kann die Tabelle ausgewählt werden.

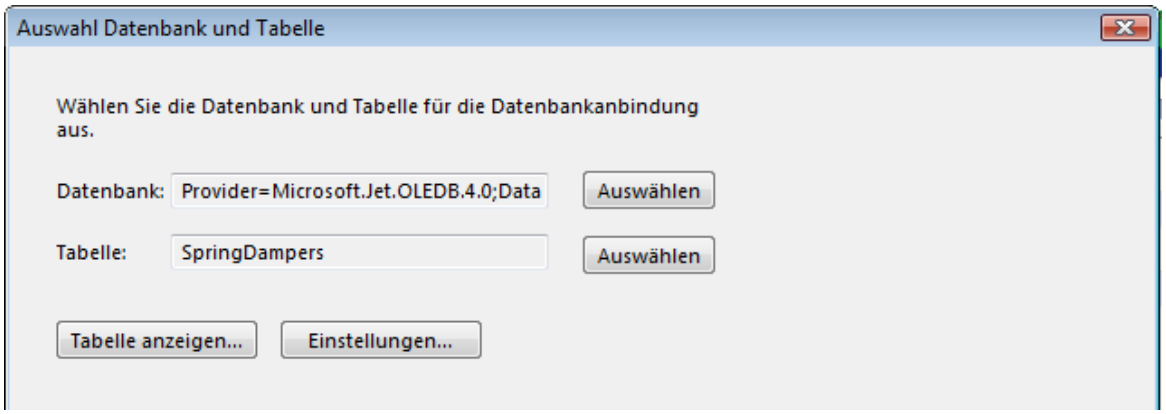



**Bild 9.78:** Auswahl Datenbank und Tabelle

**Beispiel:** Wählen sie die in der Datenbank vorhandene Tabelle „SpringDampers“ aus.

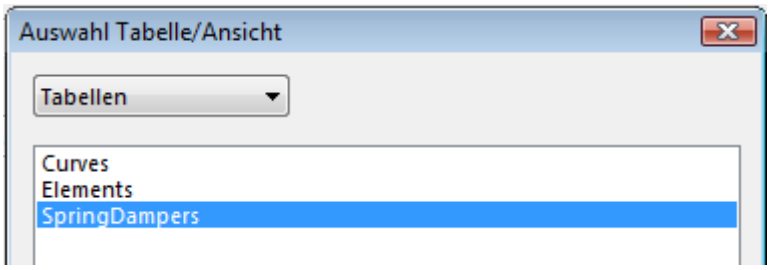



**Bild 9.79:** Auswahl Tabelle

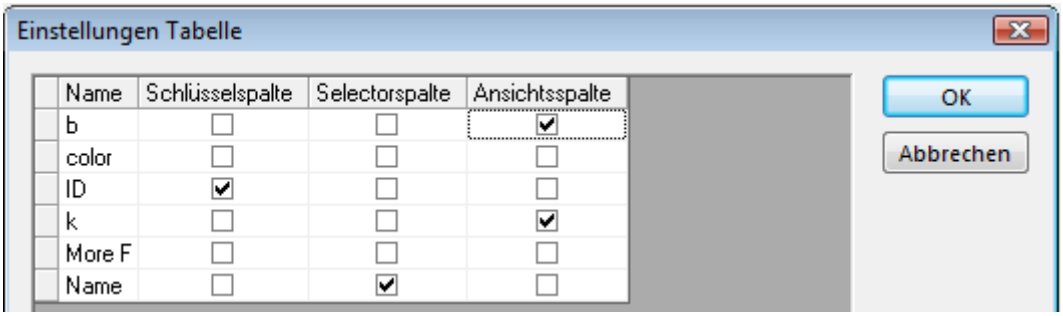



**Bild 9.80:** Einstellungen Tabelle

Nach der Auswahl der Tabelle versucht SimulationX die Schlüsselspalten zu ermitteln und automatisch als Schlüsselspalte zu definieren. Wenn das nicht möglich ist, definieren Sie über die Schaltfläche ***Einstellungen...***, mindestens eine Schlüsselspalte und eine Selektorspalte.

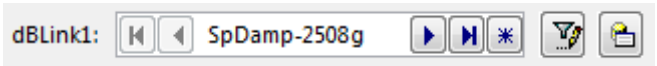
Die Spalten in der Tabellenansicht sind wie folgt angeordnet:

1. Schlüsselspalte
2. Selektorspalte
3. Ansichtsspalte

Erklärung der Spaltenarten:

Schlüsselspalte: Jede Zeile in der Tabelle wird durch die Werte in den Schlüsselspalten eindeutig identifiziert. Die Werte der Schlüsselspalte werden im Modell gespeichert, um den gewählten Datensatz später wieder zu finden.

Selektorspalte: Der Selektor präsentiert den ausgewählten Datensatz im Eigenschaftsdialog des Elements. Wenn der Inhalt der Schlüsselspalten nicht aussagekräftig ist, dann können abweichende Selektorspalten definiert werden.




Im Beispiel wurde der Produktname statt der ID als Selektorspalte definiert.

Ansichtsspalte: Die Daten der Ansichtsspalten werden während der Auswahl eines Datensatzes angezeigt und müssen nicht an einen Parameter gebunden werden. Im Beispiel wird die Farbe (Color) bei der Datenbankauswahl nur angezeigt, aber nicht in SimulationX verwendet.

Sie müssen mindestens eine Schlüssel- und eine Selektorspalte definieren.

### 9.7.3.3 Binden der Parameter

Nachdem die Tabelle ausgewählt wurde, ist das Binden von Parametern an Spalten der Tabelle möglich. In der Liste werden die Parameter des Elements angezeigt. Über ein Auswahlfeld kann jedem Parameter eine Spalte zugeordnet werden. Die Einheit der Werte in der Datenbank kann ebenfalls eingestellt werden.

**Beispiel:** Binden sie die Paramter `k` und `b` an die entsprechenden Spalten in der Tabelle.

Nach dem Binden der Parameter ist die Definition der Datenbankanbindung vollständig und die Einstellungen werden gespeichert. Die sich anschliessende Parametrierung legt einen Standarddatensatz für die Datenbankkomponente fest. Die Datenbankschnittstelle unterstützt auch die Bindung von Kennlinien und Kennfeldern. In diesem Fall wird statt der Auswahlbox ein neuer Dialog für die Definition der Bindung geöffnet. Die Bindung von Kennlinien wird im **Kapitel 9.7.6** beschrieben.

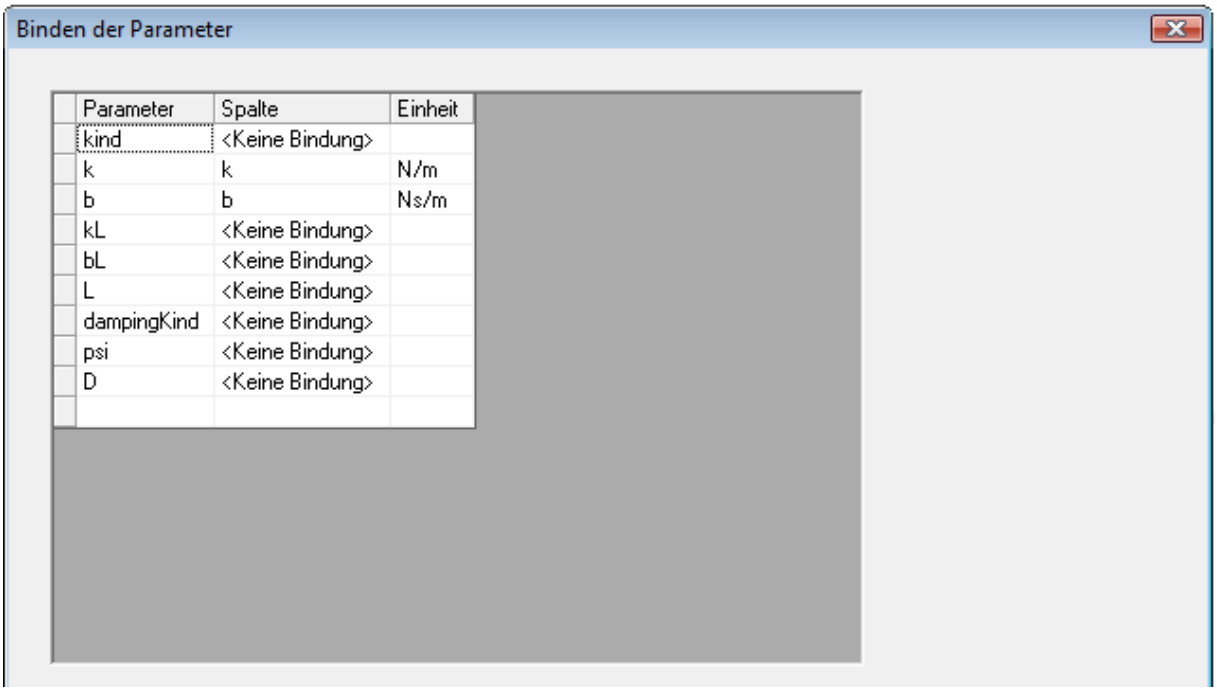



**Bild 9.81:** Binden von Parametern

### 9.7.4 Parametrieren

Sie können einen Datensatz mit Hilfe der Pfeil-Schaltflächen am Selektor auswählen oder die Tabellenansicht ( ) verwenden.

In der Tabellenansicht wird der Datensatz aus einem Datenblatt mit allen Datensätzen der Tabelle ausgewählt.

Die erste Zeile der Spaltenüberschrift enthält den Namen des Parameters und die Einheit der Werte in der Tabelle. In der zweiten Zeile wird der Name der Tabellenspalte angezeigt.

Nach der Parametrierung werden die Parameter mit Werten der Datenbank farblich hinterlegt (siehe **Bild 9.83**). Die Markierung wird gelöscht, wenn eine Änderung des Werts von Hand erfolgt.

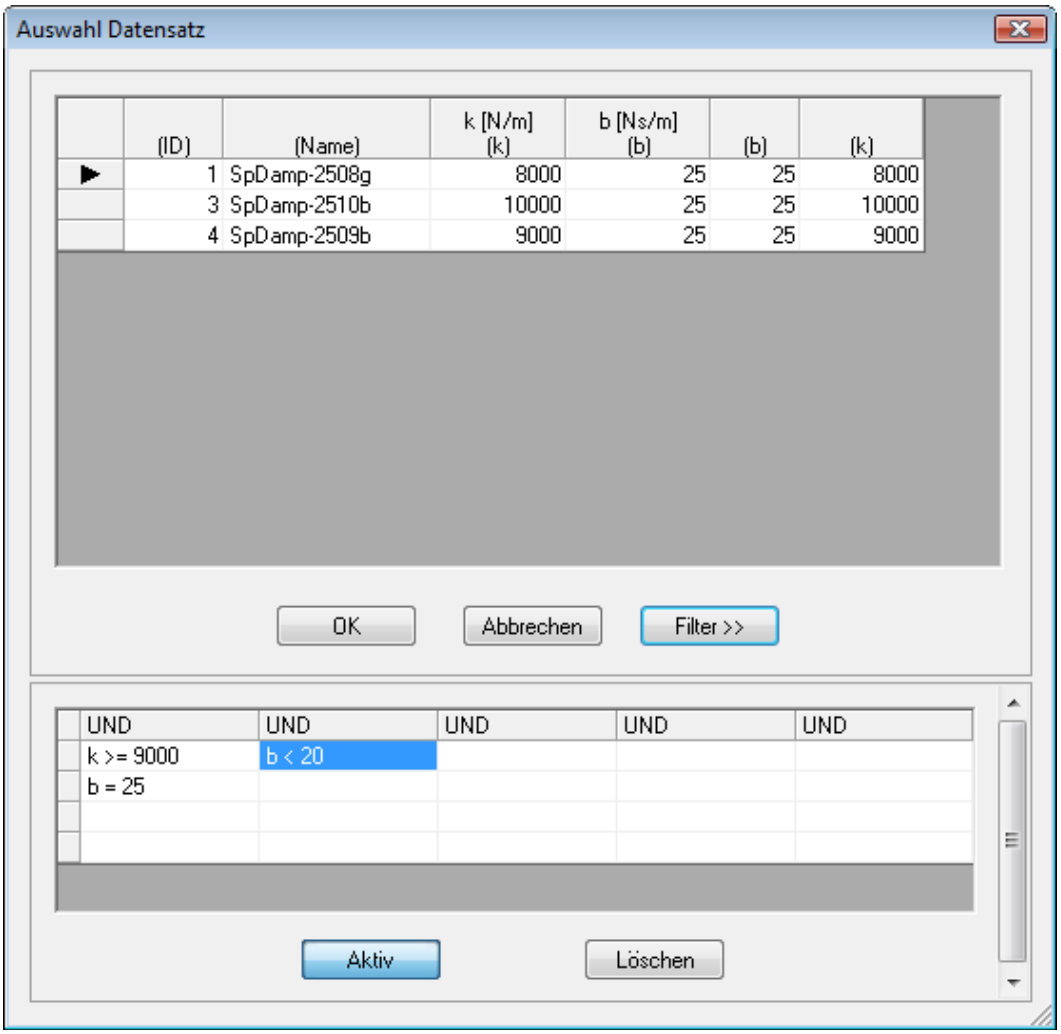



**Bild 9.82:** Parametrierung in der Tabellenansicht mit aktivem Filter

**Beispiel:** Führen sie mit den Paramtersätzen der Tabelle Simulationen mit einem Modell entsprechend **Bild 9.71** durch.

Parameter: `Mass1.m` 500g
`Mass1.x0` 5mm
`preset1.kind` Einspannung

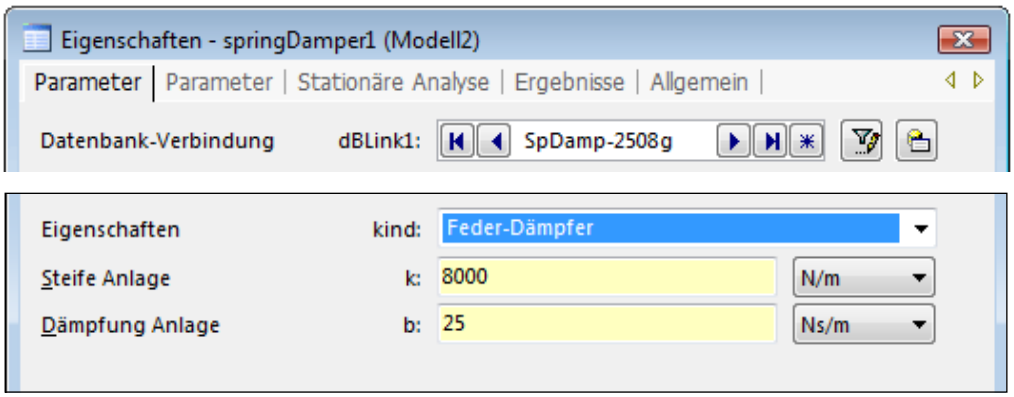



**Bild 9.83:** Parametriertes Element

## Filtern von Datensätzen

Während der Datensatzauswahl können Sie einen Filter definieren, welcher die anzuzeigenden Datensätze zusätzlich einschränkt. Die Bedingungen werden mit der Schaltfläche Filter >> ein- bzw. ausgeblendet.

Der Filter wird in einer Tabelle dargestellt, deren Zeilen und Spalten wie folgt miteinander verknüpft werden:

Die Bedingungen in einer Zeile werden durch ein logisches UND verbunden. Die Zeilen werden durch ein logisches ODER verbunden.

Mit der Schaltfläche Aktiv können Sie den Filter ein- bzw. ausschalten. Die Schaltfläche Löschen löscht alle Bedingungen des Filters.

Mit den Schaltflächen in den Bedingungen wird der folgende Dialog zur Eingabe bzw. Änderung der Bedingung geöffnet.

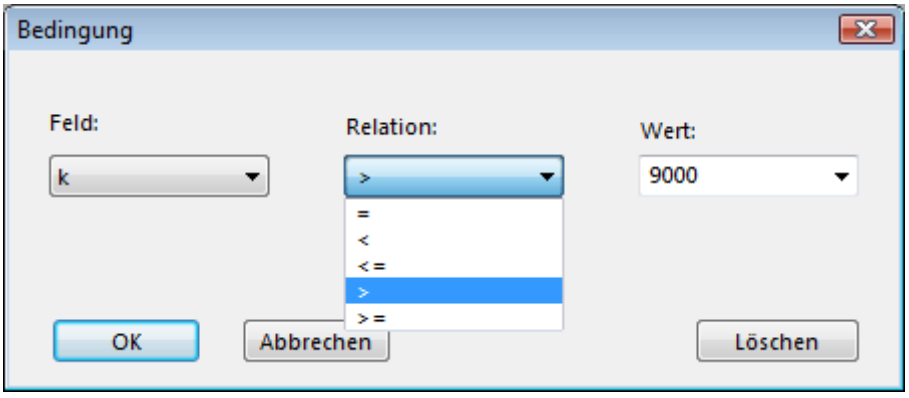



**Bild 9.84:** Eingabe von Bedingungen

Zuerst wählen Sie die Feld (Spalte) aus für welche die Bedingung definiert werden soll. Je nach ausgewähltem Feld ergeben sich die möglichen Relationen. Die Auswahlliste des Werts enthält alle Werte welche in der aktuellen Abfrage enthalten sind, sie können aber auch eigene Werte eingeben.
Filter sind nur in der Tabellenansicht wirksam und werden nicht gespeichert.

Beispiel:

| | |
|---|---|
| k>=9000 | b<20 |
| b=25 | |

→ `(k>=9000 AND b<20) OR (b=25)`

## 9.7.5 Löschen von Bindungen

Um die Datenbankanbindung zu löschen, können Sie entweder die Datenbankkomponente im Typ oder alle Bindungen der Parameter entfernen.

## 9.7.6 Binden von Kennlinien und Kennfeldern

Die Datenbankschnittstelle unterstützt auch die Bindung von Kennlinien und 2D- und 3D Kennfeldern. Sie können im Gegensatz zu Parametern nicht in einem Feld der Elementtabelle abgelegt werden, sondern werden in separaten Kennlinientabellen gespeichert.
Beim Binden von Kennlinien muss:

1. die Kennlinientabelle ausgewählt und
2. die Beziehung zwischen Element- und Kennlinientabelle über ein- oder mehrere ID Felder hergestellt werden.

Im Folgenden wird anhand eines Beispiels die Bindung einer Kennlinie und eines Kennfeldes demonstriert. Dazu verwenden wir wieder die Datenbank „...*Samples**Mechanics**Components.mdb*“. Sie enthält eine Tabelle *Elements* und eine Tabelle *Curves*.

### 9.7.6.1 Kennlinien

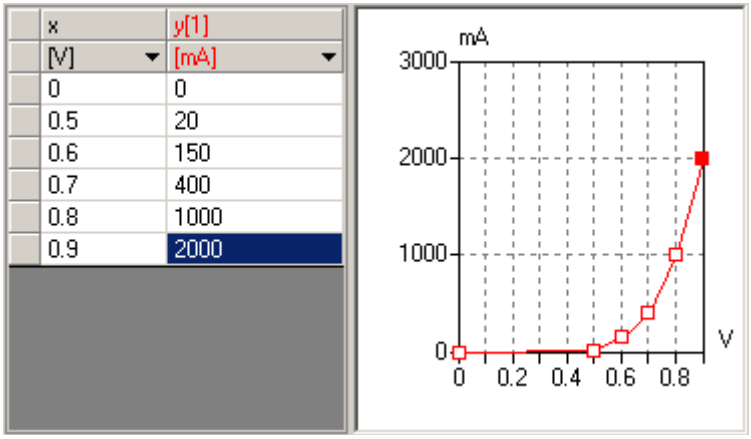
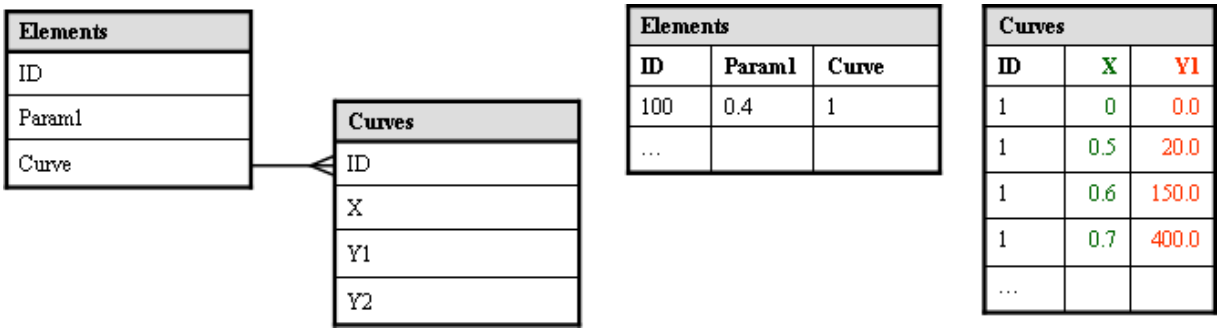



**Bild 9.85:** Beispiel Kennlinie

Kennlinie, Kennliniensatz und Kurvenschar bestehen aus zwei oder mehr Sequenzen (x, y1, y2,…).

Weil jede Sequenz der Kennlinie die gleiche Anzahl von Werten beinhaltet, kann die folgende Tabellenstruktur verwendet werden.




| Elements | | |
|---|---|---|
| ID | Param1 | Curve |
| 100 | 0.4 | 1 |
| … | | |

| Curves | | |
|---|---|---|
| ID | X | Y1 |
| 1 | 0 | 0.0 |
| 1 | 0.5 | 20.0 |
| 1 | 0.6 | 150.0 |
| 1 | 0.7 | 400.0 |
| … | | |

**Bild 9.86:** Beziehungen und Inhalt der Tabellen des Beispiels

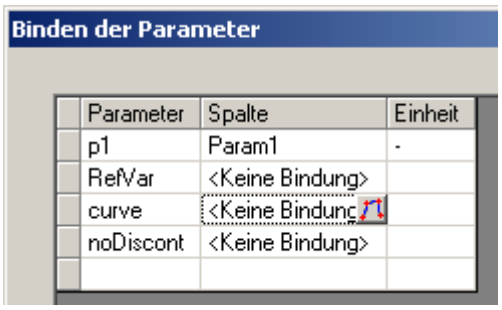
Die Kennlinientabelle (*Curves*) enthält die ID der Kurve und je eine Spalte für jede Sequenz. Die Spalte `Curve` der Elementtabelle (*Elements*) verweist auf die ID der Kennlinie. Wird für die Spalte `Curve` das Format Zahl verwendet, so muss die ID in der Tabelle *Curves* auch diesem Format entsprechen!

Dieses Vorgehen hat den Vorteil, dass die Kennlinie auch in der Datenbank einfach lesbar bleibt und deshalb problemlos verändert werden kann.

**Beispiel:**
Es wird von einem Element *Kennlinie* aus der Bibliothek Signalglieder ein neuer Typ abgeleitet. Dieser wird um eine Komponente Parameter und eine Komponente Datenbank-Anbindung erweitert. Danach wird die Erweiterung des abgeleiteten Typen fertig gestellt.

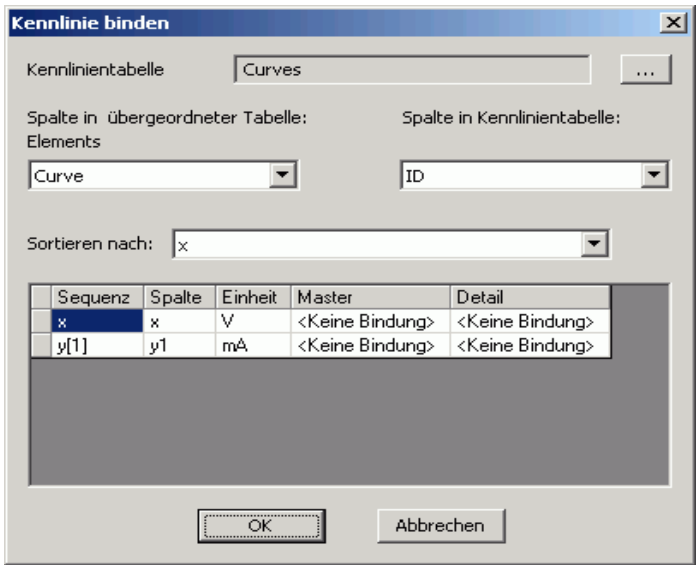
Öffnen Sie den Eigenschaftsdialog des abgeleiteten Elements. Jetzt wird zunächst die Datenbank, wie unter **9.7.3.1** beschrieben, eingebunden und die Tabelle *Elements* genutzt.

Unter *Einstellungen Tabelle* wählen Sie die ID als Schlüssel und Selektorspalte aus.




Beim Binden der Parameter wählen Sie in der Zeile *curve* die Bindung mit *Curves* aus. Durch Betätigen der Schaltfläche gelangen Sie zum Dialog für das Binden der Kennlinie.

Das Binden einer Kennlinie beinhaltet folgende Schritte:




1. Auswahl der Kennlinientabelle
2. Herstellen der Beziehung zwischen Element- und Kennlinientabelle
3. Auswahl Sortierspalte
4. Binden der Sequenzen an Spalten in Kennlinientabelle

Master- und Detailspalten werden für die Bindung von Kennfeldern verwendet.

Jetzt können Sie über das Steuerelement zwischen den Kennlinien der Tabelle wechseln.

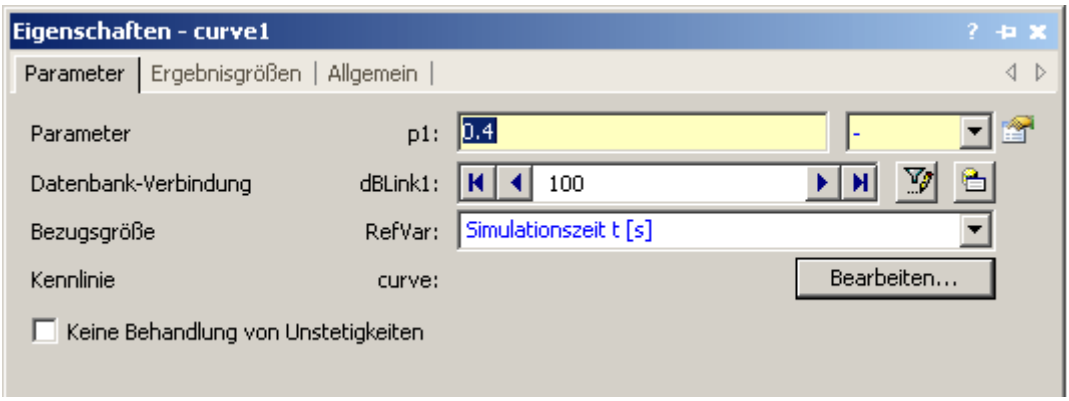



**Bild 9.87:** Element Kennlinie mit Steuerelement für die Kennlinienauswahl

### 9.7.6.2 Kennfelder

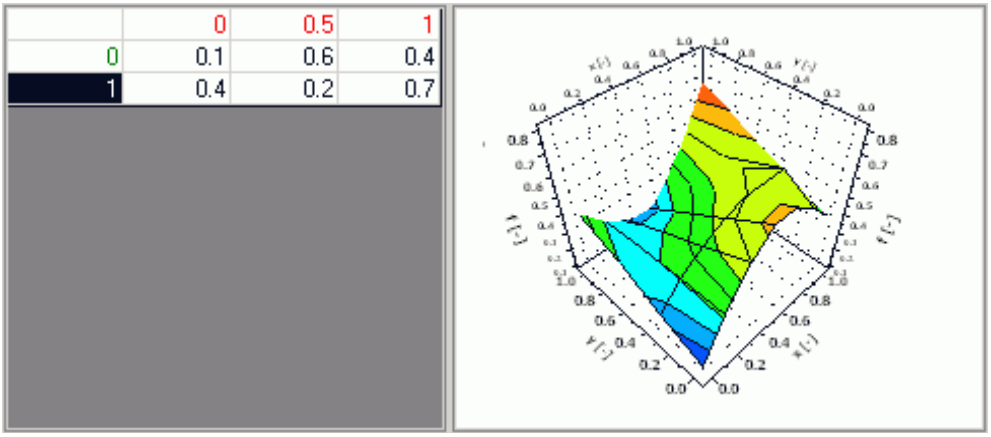


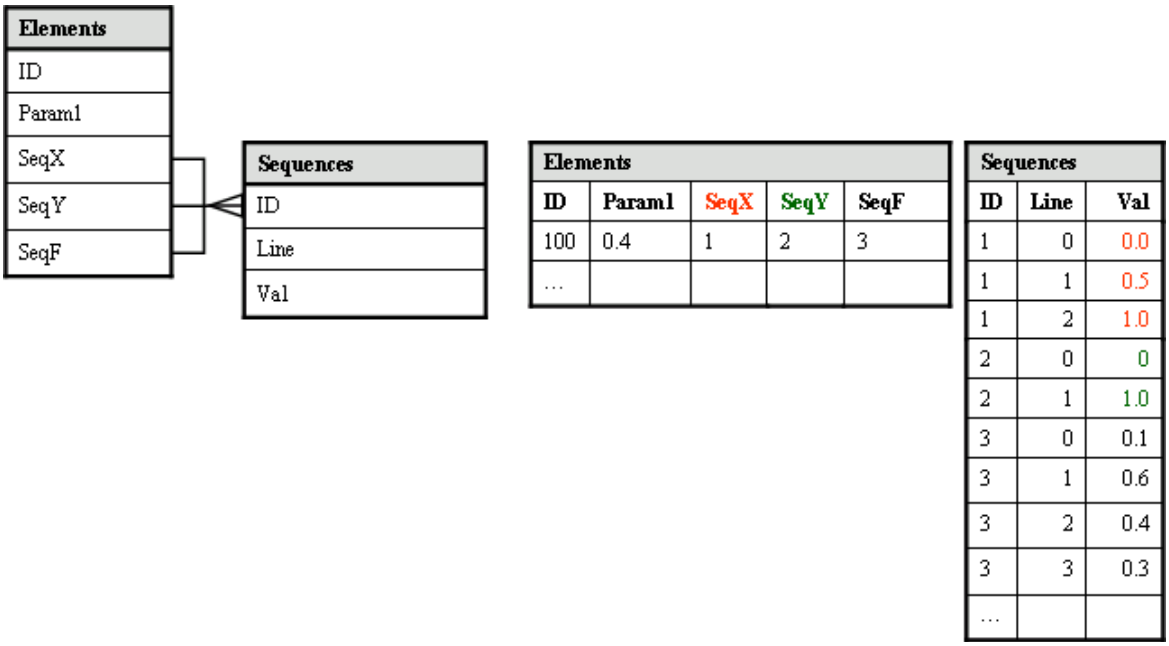

**Bild 9.88:** Beispiel 2D Kennfeld

Kennfelder sind ebenfalls aus Sequenzen aufgebaut:

- 2D-Kennfeld: x, y, f
- 3D-Kennfeld: x, y, z, f

Die Sequenzen innerhalb des Kennfelds können jedoch eine unterschiedliche Anzahl von Werten beinhalten.

Wegen der unterschiedlichen Länge wird für jede Sequenz jeweils ein Verweis in der Elementtabelle erzeugt und die Sequenz einzeln in der Sequenztabelle gespeichert. Jeder Datensatz der Sequenztabelle enthält die ID der Sequenz, einen Wert (Val) und die Position (Line) des Wertes in der Sequenz.




**Elements**

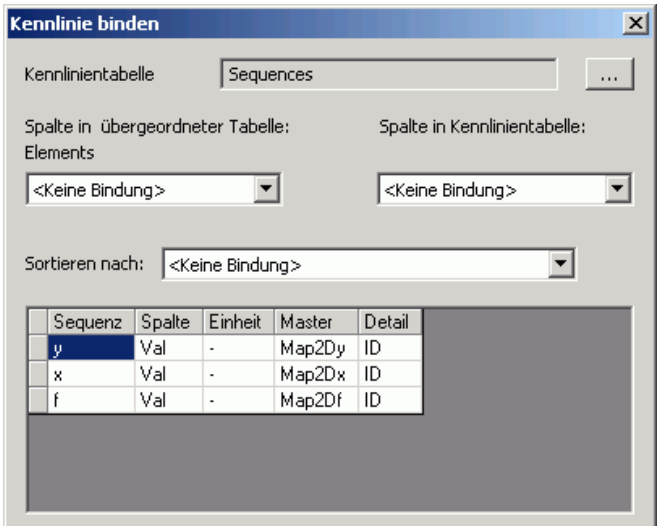
| ID | Param1 | SeqX | SeqY | SeqF |
|---|---|---|---|---|
| 100 | 0.4 | 1 | 2 | 3 |
| … | | | | |

**Sequences**

| ID | Line | Val |
|---|---|---|
| 1 | 0 | 0.0 |
| 1 | 1 | 0.5 |
| 1 | 2 | 1.0 |
| 2 | 0 | 0 |
| 2 | 1 | 1.0 |
| 3 | 0 | 0.1 |
| 3 | 1 | 0.6 |
| 3 | 2 | 0.4 |
| 3 | 3 | 0.3 |
| … | | |

**Bild 9.89:** Beziehungen und Inhalt der Tabellen des Beispiels

Das Binden eines Kennfeldes beinhaltet folgende Schritte:




1. Auswahl Sequenzentabelle
2. Auswahl Sortierspalte
3. Binden der Sequenzen an Wertspalten und Einstellung der Beziehung zwischen Elementtabelle (Master) und Sequenzentabelle (Detail)

### 9.7.7 Hierarchische Datenbankverbindungen

In zusammengesetzten Typen kann eine Datenbankschnittstelle die Datenbankschnittstellen der enthaltenen Elemente steuern. Solche hierarchischen Datenbankanbindungen erlauben die Berücksichtigung von Beziehungen zwischen den Tabellen der Datenbank. Ein weiterer Anwendungsfall ist die Auswahl identischer Datensätze für mehrere Datenbankschnittstellen.

Im Folgenden wird die Arbeitsweise anhand eines Getriebemodells demonstriert.

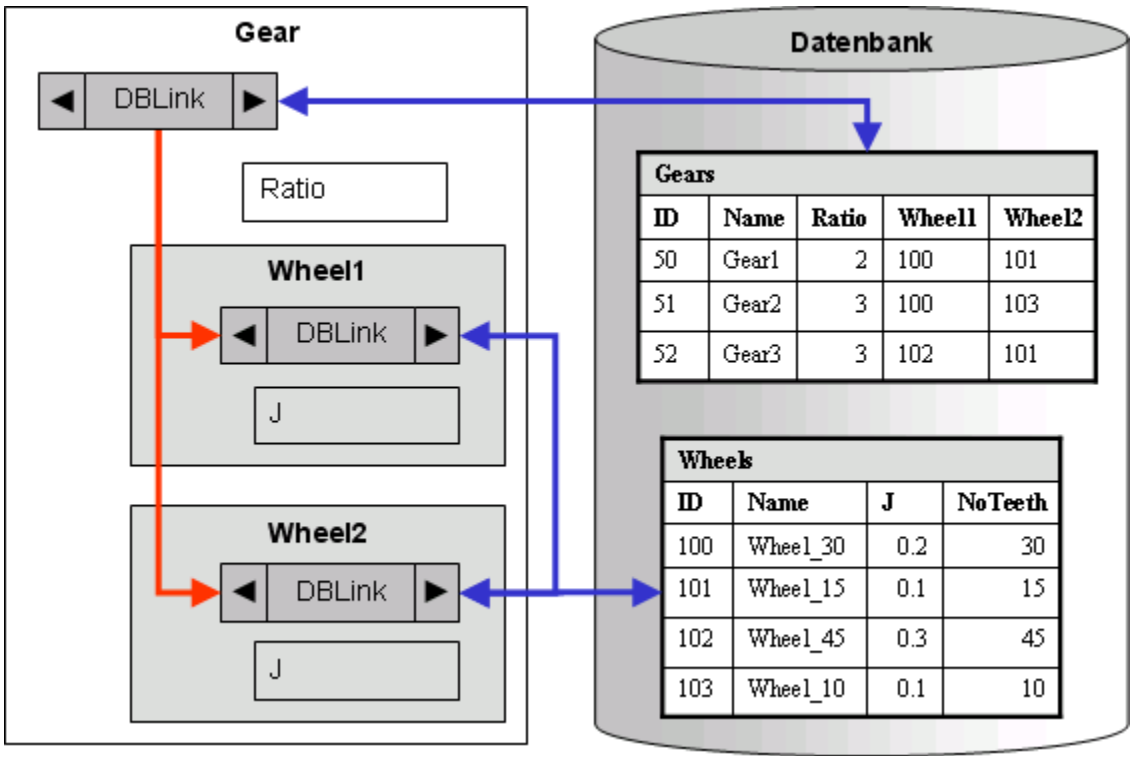



**Bild 9.90:** Beispiel Getriebedatenbank

Das Getriebemodell enthält zwei Zahnräder (Wheel1, Wheel2) deren Datenbankschnittstellen mit der Zahnradtabelle (*Wheels*) verbunden sind. Sie beinhaltet die Eigenschaften der Zahnräder. Die Datenbankschnittstelle des Getriebes verwendet die Getriebetabelle (*Gears*). Die Getriebetabelle definiert die zulässigen Kombinationen von Zahnrädern.

Ziel ist die Auswahl der Zahnräder in den Elementen Wheel1 und Wheel2 in Abhängigkeit vom ausgewählten Getriebedatensatz.

Voraussetzung für die Datenbankanbindung des Getriebes ist die vollständig definierte Datenbankanbindung beider Zahnräder. Dies kann entweder durch Definition am Zahnradtyp oder durch Definition an jedem einzelnen Zahnradelement erfolgen.

Die Parameter des Getriebes werden wie gewohnt an Spalten der Getriebetabelle gebunden. Die Bindung der untergeordneten Datenbankschnittstellen in den Zahnradelementen erfolgt mit Hilfe des folgenden Dialogs:

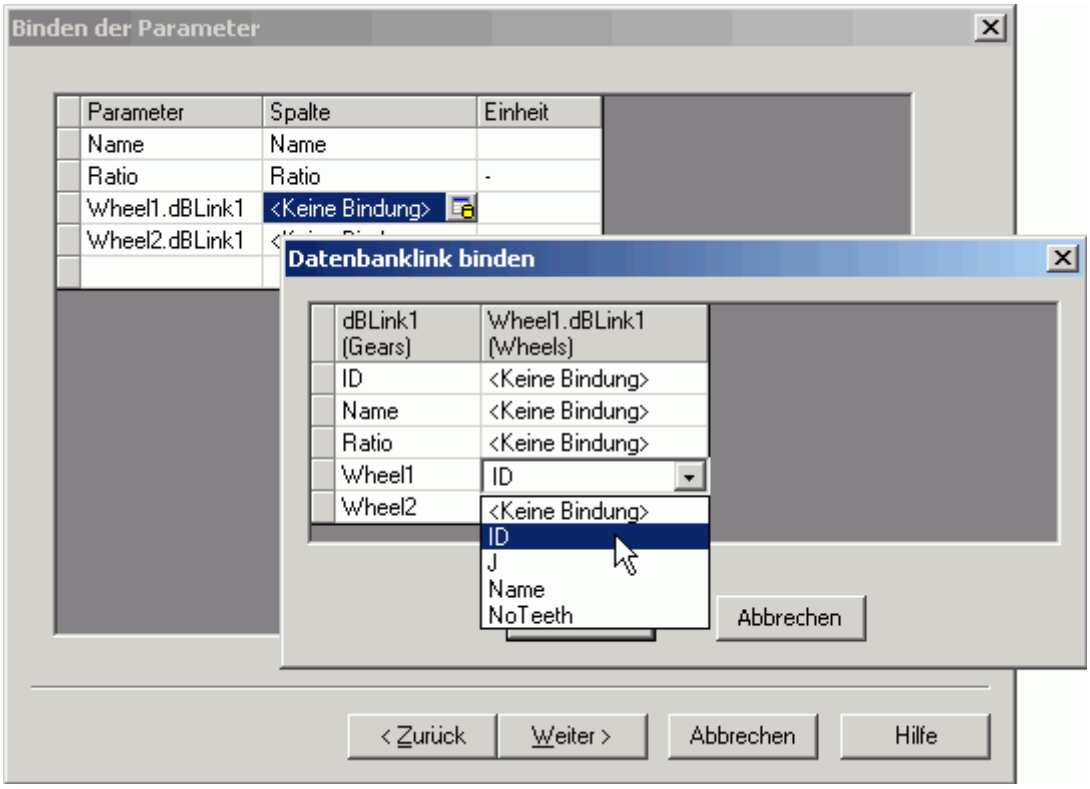



**Bild 9.91:** Bindungsdialog für untergeordneten Datenbankschnittstellen

Die Überschrift der ersten Spalte der Tabelle enthält den Namen der aktuell bearbeiteten Datenbankschnittstelle und den entsprechenden Tabellennamen. Die Überschrift der zweiten Spalte zeigt den Namen der untergeordneten Datenbankschnittstelle und deren Tabellennamen. Über die Auswahlbox wird die Beziehung zwischen beiden Tabellen hergestellt.

Für jedes Zahnrad wird eine Beziehung definiert:

- Wheel1.dbLink1: Gears.Wheel1 → Wheels.ID
- Wheel2.dbLink1: Gears.Wheel2 → Wheels.ID

Wird ein Getriebe ausgewählt, so verwendet Wheel1.dbLink1 den Datensatz mit der ID aus Spalte *Gears.Wheel1* und Wheel2.dbLink2 verwendet den Datensatz mit der ID aus Spalte *Gears.Wheel2*.

SIMULATION X®

Powered by ITI

Modelica®

# Modelica®

# 10 Modelica® in SimulationX

## 10.1 Allgemeine Hinweise

SimulationX integriert den Sprachstandard Modelica® (siehe **[4]**). Modelica ist eine objektorientierte Beschreibungssprache zur Modellierung technischer und physikalischer Systeme auf Basis von mathematischen Gleichungen. Die Modelle werden in Form eines lesbaren Quelltextes gespeichert, der sich mit üblichen Texteditoren öffnen und bearbeiten lässt.

In SimulationX ist es nicht unbedingt erforderlich, sich detailliert mit dem Sprachstandard auseinander zu setzen. Modelle können über die grafische Benutzeroberfläche wie in **Kapitel 3.2** beschrieben aus den Bibliotheken erstellt werden. Die Speicherung der Modelle erfolgt in Modelica Syntax, enthält aber einen für SimulationX spezifischen Basistyp für Simulationsdaten. Es ist jedoch auch möglich Modelle als Modelica Quellcode zu erstellen und mit SimulationX zu berechnen.

Ausdrücklich empfohlen ist die Kenntnis jener Teilmenge des Sprachstandards, die die mathematische und algorithmische Beschreibung von Systemen umfasst (Kapitel 3, 8, 10, 11 der Modelica-Sprachspezifikation). Zur Erstellung eigener Modelica Klassen empfiehlt sich die Verwendung des ***TypeDesigners*** (siehe **Kapitel 8**). Er erleichtert die Modellierung in der Modelica Sprache, indem er den Anwender unterstützt z.B. bei der Deklaration von Variablen oder der Ableitung von Basisklassen und deren Syntax automatisch erstellt. Gleichzeitig bietet er jedoch die Möglichkeit, den Quellcode unmittelbar zu editieren. Dabei werden Schlüsselwörter des Sprachstandards farblich hervorgehoben.

Es ist zu beachten, dass die mit SimulationX mitgelieferten oder zusätzlich erworbenen Bibliotheken wegen der Lizenzierungsmechanismen in der Regel in anderen Simulationsumgebungen nicht verwendet werden können. Des Weiteren geht das Konzept der SimulationX-Verbindungen über den Modelica-Sprachstandard hinaus. Andere Simulatoren unterstützen dieses Konzept der Verbindungen bisher nicht. Bitte berücksichtigen Sie dies, falls Sie Modelle für andere Zielsysteme als SimulationX entwickeln wollen. Entsprechendes gilt für die Verwendung der zugehörigen Elementanschlüsse in nutzerdefinierten Typen.

Weitere Informationen zu Modelica können **[4]** und *http://www.modelica.org* entnommen werden.

## 10.2 Modellierung mit der Modelica-Standard-Bibliothek

### 10.2.1 Installation

Die Modelica Standard Bibliothek wird bei der Installation von SimulationX mit installiert. Wahlweise kann sie auch frei von der Webseite der Modelica

Association heruntergeladen (*http://www.modelica.org*) werden. Wenn die Bibliothek mit installiert wurde, so ist der Modelica-Suchpfad (MODELICAPATH) schon richtig eingestellt. Ansonsten ist er noch anzupassen.

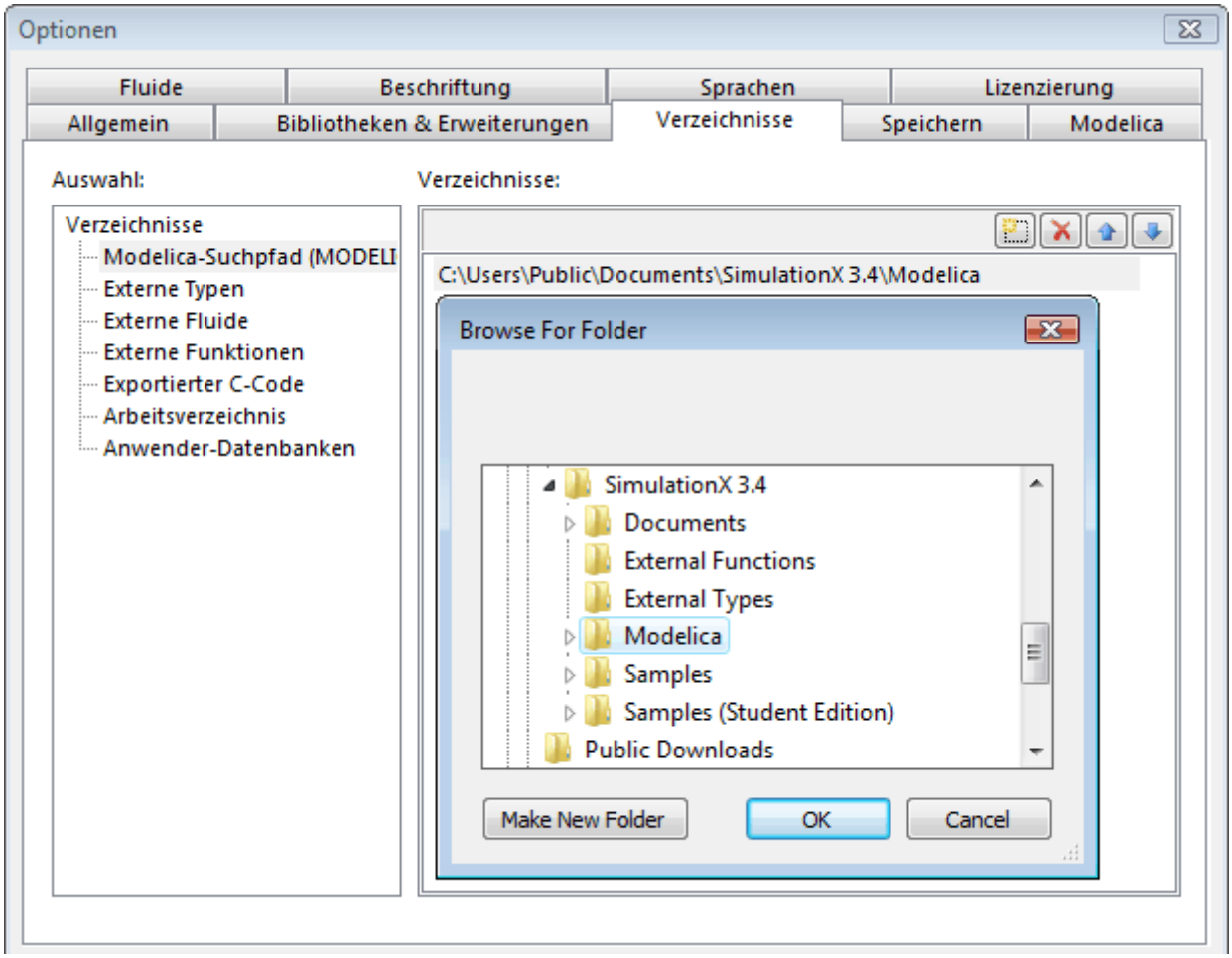



**Bild 10.1:** Einstellung des Modelica-Suchpfades (MODELICAPATH)

So passen Sie den Modelica-Suchpfad (MODELICAPATH) an:

1. Klicken Sie im Menü ***Extras*** auf ***Optionen***.
2. Wählen Sie im Dialogfeld ***Optionen*** die Registerkarte ***Verzeichnisse*** aus.
3. Klicken Sie unter ***Auswahl*** auf den Eintrag ***Modelica-Suchpfad* (MODELICAPATH)**.
4. Fügen Sie unter ***Verzeichnisse*** einen neuen Eintrag hinzu, oder bearbeiten Sie einen bereits vorhandenen Eintrag. Achten Sie darauf, dass das eigentliche Wurzelverzeichnis der Modelica-Standard-Bibliothek **in** dem ausgewählten Verzeichnis liegt.

Die Bibliothek kann in SimulationX geändert werden. Wenn das verhindert werden soll, so sind die Dateien im File-System auf „read only“ zu setzen. In diesem Fall sind im TypeDesigner die Bearbeitungswerkzeuge unsichtbar oder deaktiviert.

Die Verwendung der Typen (Klassen) der Modelica-Standard-Bibliothek erfolgt im Allgemeinen analog wie die Modellierung mit SimulationX Bibliotheks-elementen, das heißt sie werden durch Ziehen mit der Maus im Modell positioniert.

Über das Kontextmenü eines Typen und der Funktion „Öffnen“ kann er im TypeDesigner oder in verschiedenen Ansichten für eine Bearbeitung geöffnet werden (siehe auch **4.2.2.1)**. Beachten Sie auch die Einstellungen für Modelica-Dateien in **Kapitel 7.6.5**.

Das Speichern der Änderungen kann entsprechend **5.2.1** erfolgen.

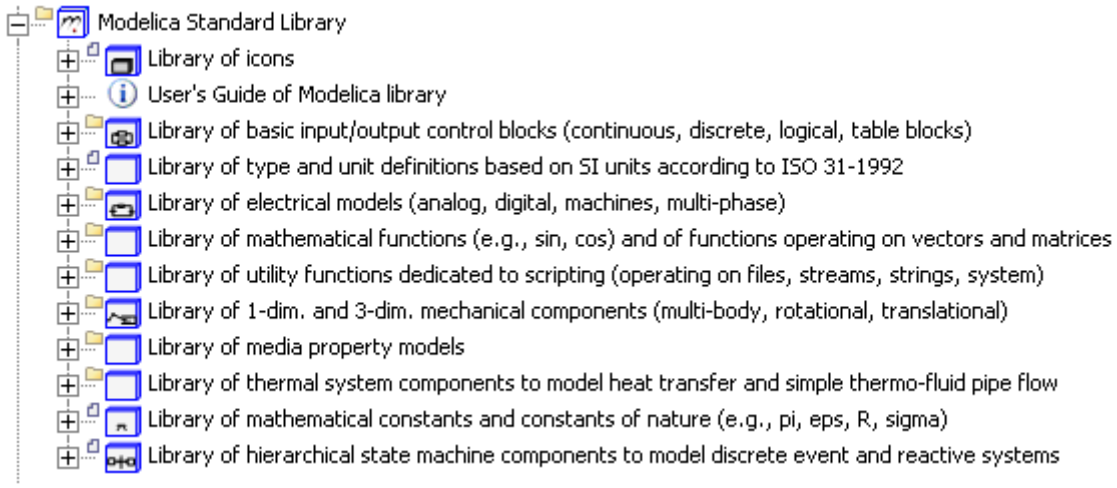



**Bild 10.2:** Ansicht der Modelica-Standard-Bibliothek in der Bibliotheksleiste

Es gibt jedoch einige Unterschiede, auf welche die folgenden Punkte näher eingehen.

1. Nicht alle in der ***Bibliotheksleiste*** (**Kapitel 4.2.2**) angezeigten Einträge der Modelica-Standard-Bibliothek sind Typen, die sich in einem Modell als Element verwenden lassen (d.h. per Drag & Drop in der Strukturansicht zu platzieren). Die Leiste zeigt alle enthaltenen Pakete (Packages, Libraries), Dokumentationen, Funktionen, Icons, Anschlüsse, Klassen (Typen) und auch komplette Beispielmodelle an. Pakete werden in der Bibliotheksleiste als blaue Boxen dargestellt (siehe **Bild 10.2**). Ein kleines Symbol in Form eines Dateiordners oder eines Papierblattes links vom Symbol weist zusätzlich auf den Speicherort des Paketes hin.
   Näheres hierzu entnehmen Sie bitte dem **Kapitel 4.2.2**.

2. Die Modelica-Standard-Bibliothek kennt keine Verbindungen wie SimulationX, d.h. die grafischen Verbindungen zwischen Modelica-Objekten sind keine Modellobjekte. Als Konsequenz besitzen Sie weder Namen noch Ergebnisgrößen oder Anfangswerte. Diese sind nur an den angeschlossenen Elementen änder- und/oder ablesbar. Modelica-Elemente werden direkt über ihre Anschlüsse verbunden.

3. Die Verwendung von Funktionen der Modelica-Bibliothek erfolgt entweder über die Angabe des vollständigen ***Idents*** (z.B. **`x=Modelica.Math.leastSquares(A,b)`**) oder durch Import von Paketen (siehe **Kapitel 8.2.2.6)** und einfache Nennung im Modell (`x=leastSquares(A,b)`). Der ***Ident*** ist dem unteren Teil der Bibliotheksleiste (**Bild 10.3**) zu entnehmen.

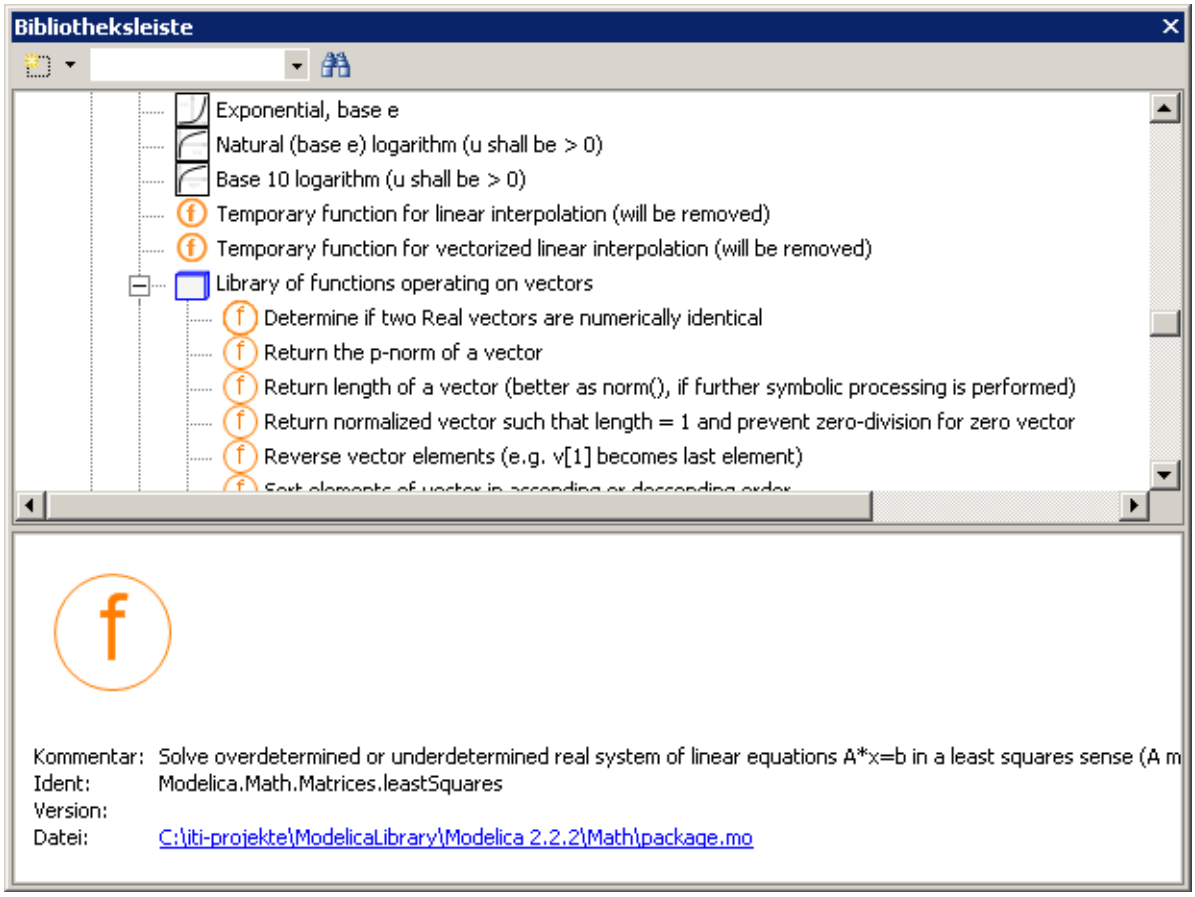



**Bild 10.3:** Informationen zu den Bibliothekelementen im unteren Fenster der Bibliotheksleiste

## 10.3 Kombinieren der SimulationX-Bibliotheken mit der Standard-Modelica-Bibliothek

Anschlüsse von Elementen der SimulationX-Bibliotheken und Anschlüsse von Modellen der Modelica-Standard-Bibliothek können zunächst nicht ohne weiteres miteinander verbunden werden. Es ist z.B. nicht möglich, eine translatorische Trägheit aus der Modelica-Standard-Bibliothek direkt mit einer Masse aus der Bibliothek der translatorischen Mechanik von SimulationX zu verknüpfen. Eine Schnittstelle kann als Modelica-Klasse ***model*** oder Externer Typ definiert werden. Der Element-Typ muss je einen der betreffenden Anschlusstypen besitzen und die physikalischen Größen durch Gleichungen verknüpfen. In der Regel erreicht man dass, indem die Summe der beiden Flussgrößen identisch Null ist, während die beiden Potentialgrößen äquivalent sein müssen.

## 10.4 Besonderheiten für SimulationX

SimulationX unterstützt Modelica®, beschrieben durch die ***Modelica® Language Specification Version 3.2*** (im Weiteren als **MLS** bezeichnet).

Im Folgenden werden einige Erläuterungen zu Notation und den Unterschieden (Erweiterungen bzw. Einschränkungen) beschrieben. Dabei orientiert sich die Gliederung an der Modelica® Spezifikation.

Modelle, die SimulationX-spezifische Erweiterungen verwenden, sind nicht konform mit der MLS und können von anderen ***Modelica®***-unterstützenden Werkzeugen ggf. nicht verarbeitet werden.

### 10.4.1 Lexikalische Struktur und Syntax

Die lexikalische Struktur ist im **Kapitel 2 der MLS** beschrieben.

Lexikalische Einheiten sind zusammenhängende Zeichenfolgen, die keine Trennzeichen, wie Leerzeichen, Tabulatoren und Zeilenwechsel, enthalten dürfen.

| | | |
|---|---|---|
| Name | ::= | Letter { Letter \| Digit \| **'_'** } **.** |
| Number | ::= | ( Digit { Digit } [ **'.'** { Digit } ] \| **'.'** Digit { Digit } ) [ **'e'** \| **'E'** [ **'+'** \| **'-'** ] Digit { Digit } ] **.** |
| Digit | ::= | **'0'** .. **'9'** **.** |
| Letter | ::= | **'A'** .. **'Z'** \| **'a'** .. **'z'** **.** |
| String | ::= | **'"'** ... beliebige Zeichen ... **'"'** **.** |
| UnitString | ::= | **''** [ Unit ] [ **'->'** Unit ] **''** |
| Unit | ::= | Zeichenkette, die eine physikalische Einheit beschreibt |

Modelica-Schlüsselworte:

| | | | |
|---|---|---|---|
| **algorithm** | **and** | **annotation** | **assert** |
| **block** | **break** | **class** | **connect** |
| **connector** | **constant** | **constrainedby** | **der** |
| **discrete** | **each** | **else** | **elseif** |
| **elsewhen** | **encapsulated** | **end** | **enumeration** |
| **equation** | **expandable** | **extends** | **external** |
| **false** | **final** | **flow** | **for** |
| **function** | **if** | **import** | **in** |
| **initial** | **inner** | **input** | **loop** |
| **model** | **not** | **or** | **outer** |

| | | | |
|---|---|---|---|
| **output** | **package** | **parameter** | **partial** |
| **protected** | **public** | **record** | **redeclare** |
| **replaceable** | **return** | **stream** | **then** |
| **true** | **type** | **when** | **while** |
| **within** | | | |

Diese Schlüsselwörter dürfen nicht für Namen und Bezeichner verwendet werden.

In SimulationX gibt es folgende zusätzliche Schlüsselworte:

| | |
|---|---|
| **version** | Definiert die Versionsnummer bei Typen. |
| **required** | Eine spezielle Version des Typs wird benötigt. |
| **nondiscrete** | Macht eine sonst diskrete Variable (integer, boolean) nicht-diskret. |
| **accept** | Markiert spezielle Algorithmussektion für die Behandlung von Impulsen (**impact**) |
| **_iti_special** | Markiert spezielle Boolean-Parameter |
| **_iti_parameter** | Parameter, die nicht konstant sein müssen |
| **_iti_startparam** | Parameter, die als Anfangswerte verwendet werden |

Die Syntax von Modelica® wird im **Anhang B der MLS** beschrieben.

## 10.4.2 Operatoren und Ausdrücke

Operatoren und Ausdrücke werden in **Kapitel 3 der MLS** beschrieben.

### 10.4.2.1 Arithmetik-Operatoren

| | |
|---|---|
| **+** | Addition |
| **-** | Subtraktion |
| ***** | Multiplikation |
| **/** | Division |
| **^** | Potenzieren |

Für diese Operatoren gelten bei Verwendung mit Vektoren und Matrizen besondere Regeln (z.B. Skalarprodukt) (siehe **MLS Kapitel 10.6**).

Für die elementweise Verknüpfung von mehrdimensionalen Werten stehen spezielle Operatoren zur Verfügung:

| | |
|---|---|
| `.+` | elementweise Addition |
| `.-` | elementweise Subtraktion |
| `.*` | elementweise Multiplikation |
| `./` | elementweise Division |
| `.^` | elementweise Potenzieren |

### 10.4.2.2 Vergleichs- und logische Operatoren

| | |
|---|---|
| `>` | größer als |
| `>=` | größer oder gleich als |
| `<` | kleiner als |
| `<=` | kleiner oder gleich als |
| `==` | Gleichheit innerhalb Ausdruck |
| `<>` | Ungleichheit |
| `not` | Negation |
| `and` | logisches UND |
| `or` | logisches ODER |

Die Argumente der Vergleichs- und logischen Operatoren müssen skalare Werte sein.

### 10.4.2.3 Einfache Mathematische Funktionen

| | |
|---|---|
| `sin(x)` | Sinus |
| `cos(x)` | Cosinus |
| `tan(x)` | Tangens |
| `sinh(x)` | Sinus Hyperbolicus |
| `cosh(x)` | Cosinus Hyperbolicus |
| `tanh(x)` | Tangens Hyperbolicus |
| `asin(x)` | Arcus Sinus ($-1 \leq x \leq 1$) |
| `acos(x)` | Arcus Cosinus ($-1 \leq x \leq 1$) |
| `atan(x)` | Arcus Tangens |
| `atan2(x,y)` | Arcus Tangens von x/y. |

| | |
|---|---|
| **exp(x)** | Exponential (Basis e; x>0) |
| **log(x)** | Logarithmus (Basis e; x>0) |
| **log10(x)** | Logarithmus (Basis 10; x>0) |
| **sqrt(x)** | Quadratwurzel |

In SimulationX zusätzlich verfügbar:

| | |
|---|---|
| **arcsin(x)** | Arcus Sinus (asin) |
| **arccos(x)** | Arcus Cosinus (acos) |
| **arctan(x)** | Arcus Tangens (atan) |
| **log(b,x)** | Logarithmus von x zu Basis b |
| **lg(x)** | Logarithmus von x zu Basis 10 |

Bei Anwendung der Funktionen auf mehrdimensionale Werte (Vektoren, Matrizen …) werden diese elementweise Angewendet und liefern einen Wert der gleichen Dimension zurück.

## 10.4.3 Klassen, Typen und Deklarationen

Klassen, Typen und Deklarationen sind in **Kapitel 4 der MLS** beschrieben.
In SimulationX werden diese mit Hilfe des TypeDesigners erstellt.

### 10.4.3.1 Komponentendeklaration

Die Syntax von **type_prefix** (**MLS 4.4.1**) wurde erweitert:

```
type_prefix :
     [ flow ]
[ discrete | nondiscrete | parameter | constant ]
[ input | output ]
[ _iti_parameter | _iti_parameter | _iti_special ]
```

Für die Bedeutung der neuen Schlüsselwörter siehe **Kapitel 10.4.1**.

### 10.4.3.2 Attribute vordefinierter Typen

Vordefinierte einfache Typen haben Attribute (**MLS 4.8**). Attribute halten zusätzliche Informationen, die von der SimulationX Oberfläche (GUI), der symbolischen Analyse (GSA) oder dem Löser ausgewertet werden und zur Steuerung benutzt werden.

Folgende zusätzliche Attribute werden unterstützt:

| Attribut | Typ | Standard-wert | Ver-wendet von | De-finiert an | Beschreibung |
|---|---|---|---|---|---|
| **min** | wie Variable | DBL_MAX | GSA, Löser | Real, Integer, Enume-ration | Untere Grenze des Wertes. Siehe Minima und Maxima. |
| **max** | wie Variable | DBL_MAX | GSA, Löser | Real, Integer, Enume-ration | Obere Grenze des Wertes. Siehe Minima und Maxima. |
| **start** | wie Variable | kein Wert | GSA, Löser | Real, Integer, Boolean | Der Startwert zu Beginn der Simulation |
| **fixed** | Boolean | true | Löser | Real, Integer, Boolean | true -> Der Startwert ist verbindlich false -> Der Startwert ist nur ein Richtwert für die Berechnung konsistenter Anfangswerte |
| **notFixed** | Boolean | false | Löser | Real | true -> Erlaubt dem Löser an Unstetigkeiten das Ändern des Zustandes, auch wenn es physikalisch nicht möglich ist (Ableitung existiert) |
| **nominal** | Real | kein Wert | Löser | Real | Wird zur Skalierung der Zustandsgröße im Löser benutzt. |
| **stateSelect** | enumeration StateSelect | StateSelect. default | GSA | Real | Enumerationwert der die Wahl non Zuständen in der GSY beeinflusst. Siehe StateSelect-Aufzählungstyp. |
| **minNotReached** | Boolean | false | GSA, Löser | Real | true -> Die untere Grenze wird nicht erreicht. Siehe Minima und Maxima. |

| | | | | | |
|---|---|---|---|---|---|
| **maxNotReached** | Boolean | false | GSA, Löser | Real | true -> Die obere Grenze wird nicht erreicht. Siehe Minima und Maxima. |
| **absTol** | Real | globale Einstellung | Löser | Real | Wenn gesetzt -> die absolute Toleranz für diese Zustands-größe, sonst wird der global eingestellte Wert verwendet |
| **relTol** | Real | globale Einstellung | Löser | Real | Wenn gesetzt -> die relative Toleranz für diese Zustandsgröße sonst wird der global eingestellte Wert verwendet |
| **discontChange** | Boolean | false | Löser | Real | Die Zustandsgröße darf während der Integration springen. |
| **notFixed** | Boolean | false | Löser | Real | Dir Zustandsgröße darf sich während der Anfangs-wertberechnung nach Events ändern, obwohl die Existenz einer Ableitung dagegen spricht |
| **quantity** | String | Basics. Unitless | GUI | Real, Integer | Die physikalische Größe. |
| **comment** | String | leer | GUI | alle | Der angezeigte Kommentar |
| **active** | Boolean | true | GSA | alle | (nur lesbar) Der Parameter/die Variable ist aktiv (nicht ausgeschalten durch ActivityGroups oder bedingte Deklaration) |

***Minima und Maxima***
In SimulationX werden die Angaben der Attribute `min`, `max`, `minNotReached` und `maxNotReached` genutzt, um zu bestimmen, ob eine Variable Null werden kann oder nicht. Diese Eigenschaft ist wichtig, wenn eine Gleichung, in der diese Variable vorkommt, nach einer anderen Variablen umgestellt werden soll und dafür durch diese Variable dividiert werden müsste.
Kann die Variable 0 werden, so wird nicht dividiert (um eine mögliche Division durch 0 zu verhindern), sondern die Gleichung wird implizit. D.h. die Variable, nach der diese Gleichung umgestellt werden soll, wird zur Zustandsgröße, und der numerische Löser muss die Gleichung lösen.

Da solche impliziten Gleichungen meist numerisch schwer lösbar sind, sollte man versuchen, sie durch geeignetes Setzen von Minimum und Maximum zu verhindern.

## 10.4.4 Namensräume und Bezeichner

Namensräume und Bezeichner sind in **Kapitel 5 der MLS** beschrieben.

### 10.4.4.1 Referenzen auf lokale Parameter

In Modelica beginnt die Suche nach referenzierten Bezeichnern in Modifikationen immer in dem umgebenen Typ. In Parametern ist es aber oft sinnvoll, auf andere Parameter desselben Objekts zu referenzieren. Um das zu vereinfachen und solche Objekte kopierbar zu machen, kann man in SimulationX der Referenz ein **`self.`** voranstellen. In diesem Fall beginnt die Suche im umgebenen Namensraum des Parameters. Bei der Eingabe von Ausdrücken in Parameter wird das **`self.`** wenn notwendig automatisch eingefügt.

## 10.4.5 Vererbung, Modifikation und Redeklaration

Vererbung, Modifikation und Redeklaration werden im **Kapitel 7 der MLS** beschrieben.

## 10.4.6 Gleichungen

Gleichungen werden im **Kapitel 8 der MLS** beschrieben.

**Syntax:**
Reference = Expression ;
**(**OutputExpressionList**)** = FunctionCall;

In Gleichungen wird die Gleichheit zweier Ausdrücke mit '=' beschrieben. Dies ist keine Zuweisung sondern bedeutet, dass diese Gleichheit während der Simulation gilt.
Die Auswertungsreihenfolge der Gleichungen ist **unbestimmt**. Sie werden auch ggf. nach anderen Variablen umgestellt.

Durch die Eingabe von unstetigen Ausdrücken (z.B if ... then ... else ...) können auch unstetige Gleichungen erzeugt werden.
Die zweite Form Gleichungen wird verwendet, wenn die gerufene Funktion mehrere Rückgabewerte besitzt. Dabei werden die Variablen auf der linken Seite mit den output-Variablen der Funktion über die Reihenfolge zugewiesen. In der Liste können die Variablen für nicht benötigte Rückgabewerte weggelassen werden. Es ist dann nur das trennende Komma zu schreiben. Am Ende der Liste können die Kommas weggelassen werden.
Beispiel:

```
when initial() then
  reinit(state,y0);
end when;
der(state) = G/Ti * x;
y = state + G*x;
xv = vector(x);
if size(B,1)>1 then
        p = B[1:n] / B[n1];
        if size(A,1) >= size(B,1) then
                q[1:n]=(A[1:n]-A[n1]* B[1:n]/B[n1])/B[n1];
                q[n1] = A[n1]/B[n1];
        else
                q[1:n]=A[1:n]/B[n1];
        end if;
        der(z[:,1])= - p[1] * z[:,n] + q[1] * xv;
        for k in 2:n loop
                der(z[:,k])=z[:,k-1]-p[k]*z[:,n]+q[k]*xv;
        end for;
        yv = z[:,n] + q[n1] *xv;
else
        yv = A[1]/B[1] * xv;
end if;
y = if ndims(x)>0 then yv else scalar(yv);
```

### 10.4.6.1 Die for-Anweisung

**Syntax:**
**for** Name **in** Expression, Name **in** Expression,... **loop**
  { Equation ";" }
**end for;**

Der Ausdruck in einer for-Anweisung muss ein Vektorausdruck sein. Er wird zu Beginn einmal ausgewertet. Den Schleifenvariablen werden nacheinander die Elemente des Vektors zugewiesen und für jedes Element wird der Schleifenkörper ausgeführt.

Beispiele:

```
for i in 1:10 loop ...  end for
// i bekommt die Werte 1,2,3,...,10

for i in 1.0 : 1.5 : 5.5 loop ... end for
// i bekommt die Werte 1.0,2.5,4.0,5.5

for i in { 1,3,6,7 } loop ... end for
// i bekommt die Werte 1,3,6,7

for i in { { 1,2 }, {3,4}} loop ... end for
// i bekommt die Werte {1,2} und {3,4}
```

Durch die for-Anweisung wird eine lokale Variable deklariert, die nur in dem Schleifenkörper sichtbar ist. Sie kann Variablen gleichen Namens in umgebenden Namensräumen verdecken.

In **for**-Anweisungen innerhalb von **Gleichungssektionen** müssen die Bereiche **konstant** sein.

**Ableitung des Bereichs**

In einer for-Anweisung kann der Bereich für eine Schleifenvariable auch weggelassen werden, wenn diese Variable innerhalb der Schleife als Array-Index verwendet wird. Der Bereich ergibt sich dann aus der Größe des Arrays in der indizierten Dimension als 1:size(array,indexpos). Wenn die Variable mehrfach als Index verwendet wird, müssen die entsprechenden Dimensionen übereinstimmen.

**Mehrere Schleifenvariablen**

```
for i in 1:2, j in {1,2,3} loop
    x[i+j]=i+j;
end loop;
ist equivalent zu

for i in 1:2 loop
    for j in {1,2,3} loop
        x[i+j]=i+j;
    end loop;
end loop;
```

Schleifen die einen konstanten Bereich besitzen werden von der symbolischen Analyse ggf. expandiert.

### 10.4.6.2 If-Anweisung

**Syntax:**
**if** Expression **then**
{ Equation ";" }
**elseif** Expression **then**
{ Equation ";" }
**else**
{ Equation ";" }
**end if;**

Die **if**-Anweisung besteht aus einem **if**-Abschnitt dem mehrere **elseif**-Abschnitte folgen können. Den Abschluss bildet ein optionaler End-Abschnitt.
Der Ausdruck in den **if**- und **elseif**-Abschnitten muß ein skalarer boolean-Wert sein. Diese Ausdrücke werden nacheinander evaluiert. Liefert ein Ausdruck **true**, so wird der zugehörige Anweisungsblock ausgewählt. Liefert keiner der Ausdrücke **true**,so wird der **else**-Zweig ausgewählt falls er existiert, ansonsten wird nichts ausgewählt.
In der Gleichungssektion wird der ausgewählte Anweisungsblock abgearbeitet, wenn die Gleichungen in dem selektierten Abschnitt während der Simulation erfüllt werden.

**if**-Anweisungen in Gleichungssektionen die **keine konstanten** Bedingungen haben, müssen einen **else**-Abschnitt besitzen **und die Anzahl der Gleichungen** muss in allen Abschnitten **gleich** sein.

Enthalten **if**- oder **elseif**-Abschnitte konstante Bedingungen, so werden die Abschnitte mit nichtzutreffenden Bedingungen eliminiert. Enthält die **if**-Anweisung nur noch eine konstante zutreffende Bedingung, so wird die **if**-Anweisung durch ihren Inhalt ersetzt.

### 10.4.6.3 Die when-Anweisung

**Syntax:**
**when** Expression **then**
{ Equation ";" }
**elsewhen** Expression **then**
{ Equation ";" }
**end when;**

Die **when**-Anweisung besteht aus einem **when**-Abschnitt dem mehrere **elsewhen**-Abschnitte folgen können.
Der Ausdruck der **when**- oder **elsewhen**-Abschnitte muss ein diskreter boolean Wert sein. Es kann ein Skalar oder ein Vektor sein. Die Anweisungen in einem Abschnitt werden nur ausgeführt, wenn der boolean Wert true **wird** (an einer

false ->true Flanke). Das kann nur an Unstetigkeitsstellen vorkommen (diskreter Ausdruck). Während der kontinuierlichen Integration wird der Körper niemals ausgeführt.
Wenn b ein Skalar ist, so ist

```
when b then ... end when;
```

equivalent zu

```
if edge(b) then ... end if;
```

Wenn b ein n-Vektor ist, so ist

```
when b then ... end when;
```

equivalent zu

```
if edge(b[1]) or edge(b[2]) or ... edge(b[n]) then ...
end if;
```

Für when-Anweisungen in Gleichungssektionen gelten folgende Einschränkungen:

- Sie müssen die Form Variable = Ausdruck; oder
- (var1, var2, ... ) = funktion(...); haben.
- assert(...), terminate(...), reinit(...) sind erlaubt.
- **for**- oder **if**-Anweisungen sind erlaubt, wobei für die enthaltenen Gleichungen dieselben Einschränkungen gelten.
- Die verschiedenen **when**-/**elsewhen**- Abschnitte müssen die gleichen Variablenreferenzen auf der linken Seite haben.
- In **if**-Anweisungen innerhalb von **when**-Anweisungen müssen die gleichen Variablenreferenzen auf der linken Seite stehen, wenn die Bedingungen nicht konstant sind.

**when**-Anweisungen dürfen nicht verschachtelt werden.

## 10.4.7 Konnektoren und connect-Gleichungen

Konnektoren und connect-Gleichungen werden im **Kapitel 9 der MLS** beschrieben.

In SimulationX können (mehrere) Konnektoren aus vordefinierten Bibliotheken mit (einem) passenden Verbindungsobjekt („Connection“) in beliebiger Anzahl mittels connect-Gleichungen verbunden werden. Die direkte Verbindung zweier SimulationX-Konnektoren durch eine connect-Gleichung wird nicht unterstützt.

## 10.4.8 Arrays

Arrays werden im **Kapitel 10 der MLS** beschrieben.

**Array** ist die allgemeine Bezeichnung für einen Wert höherer Dimension. **Skalare** haben die Dimension 0, **Vektoren** die Dimension 1 und **Matrizen** die Dimension 2.
Die **Elemente** eines Arrays sind die in ihm enthaltenen (skalaren) Werte. Funktionen, die auf Skalaren definiert sind, sind auf Arrays erweitert worden; sie werden elementweise angewendet. Ergebnis ist dann ein Array gleicher Dimension.

SimulationX unterstützt die Deklaration von Array beliebiger (unbekannter) Dimension vordefinierter Typen, z.B.

```
Real x[*]
```

Die Dimension von x wird durch den Kontext bestimmt, in dem x verwendet wird, dabei kann es auch zur skalaren Größe werden.

## 10.4.9 Algorithmen

Algorithmen werden im **Kapitel 11 der MLS** beschrieben.

### 10.4.9.1 Zuweisungen

**Syntax:**
Reference:= Expression ;
**(**OutputExpressionList**) :=** FunctionCall;

In Algorithmen ist der Zuweisungsoperator ":=", das Zeichen '=' kann dort nicht verwendet werden. Zuweisungen werden in der Reihenfolge abgearbeitet, in der sie aufgeschrieben wurden. Dabei können während der Abarbeitung zwischen zwei aufeinander folgenden Zuweisungen, Anweisungen aus anderen Elementen eingefügt werden.

Beispiel:

```
Y = if(t<0.2) then 0 else 1
```

oder

```
when sample(start,interval)then

   y:=x;

end when;

z:=sign(sin(pi*x/I));

when{change(z),initial()}then
```

```
    if(x/I)<0 then
        y:=I*ceil(x/I);
    else
      y:=I*floor(x/I);
    end if;
  end when;
```

### 10.4.9.2 Die for-Anweisung

**Syntax:**
**for** Name **in** Expression, Name **in** Expression,... **loop**
{ Algorithm ";" }
**end for;**

Der Ausdruck in einer for-Anweisung muss ein Vektorausdruck sein. Er wird zu Beginn einmal ausgewertet. Den Schleifenvariablen werden nacheinander die Elemente des Vektors zugewiesen und für jedes Element wird der Schleifenkörper ausgeführt.

### 10.4.9.3 If-Anweisung

**Syntax:**
**if** Expression **then**
{ Algorithm ";" }
**elseif** Expression **then**
{ Algorithm ";" }
**else**
{ Algorithm ";" }
**end if;**

Die **if**-Anweisung besteht aus einem **if**-Abschnitt dem mehrere **elseif**-Abschnitte folgen können. Den Abschluss bildet ein optionaler End-Abschnitt.
Der Ausdruck in den **if**- und **elseif**-Abschnitten muß ein skalarer boolean-Wert sein. Diese Ausdrücke werden nacheinander evaluiert. Liefert ein Ausdruck **true**, so wird der zugehörige Anweisungsblock ausgewählt. Liefert keiner der Ausdrücke **true**,so wird der **else**-Zweig ausgewählt falls er existiert, ansonsten wird nichts ausgewählt.
In einem Algorithmus wird dann der ausgewählte Anweisungsblock abgearbeitet.

### 10.4.9.4 Die when-Anweisung

**Syntax:**
**when** Expression **then**
{ Algorithm ";" }
**elsewhen** Expression **then**
{ Algorithm ";" }
**end when;**

Die **when**-Anweisung besteht aus einem **when**-Abschnitt dem mehrere **elsewhen**-Abschnitte folgen können.
Der Ausdruck der **when**- oder **elsewhen**-Abschnitte muss ein diskreter boolean Wert sein. Es kann ein Skalar oder ein Vektor sein. Die Anweisungen in einem Abschnitt werden nur ausgeführt, wenn der boolean Wert true **wird** (an einer false ->true Flanke). Das kann nur an Unstetigkeitsstellen vorkommen (diskreter Ausdruck). Während der kontinuierlichen Integration wird der Körper niemals ausgeführt.
Wenn b ein Skalar ist, so ist

```
when b then ... end when;
```

equivalent zu

```
if edge(b) then ... end if;
```

Wenn b ein n-Vektor ist, so ist

```
when b then ... end when;
```

equivalent zu

```
if edge(b[1]) or edge(b[2]) or ... edge(b[n]) then ...
 end if;
```

**when**-Anweisungen dürfen nicht verschachtelt werden.

**when**-Anweisungen dürfen nicht in Funktionen verwendet werden.

#### 10.4.9.5 Die while-Anweisung

**Syntax:**
**while** Expression **loop**
  { Algorithm }
**end while;**

In Ergänzung zur for-Anweisung gibt es die while-Anweisung. Diese kann nur innerhalb von Algorithmen eingesetzt werden. Der Ausdruck der while-Anweisung muss ein skalarer boolean-Wert sein. Der Schleifenkörper wird solange ausgeführt, solange der Ausdruck den Wert true liefert.

### 10.4.10 Funktionen

Funktionen werden in **Kapitel 12 der MLS** beschrieben.

### 10.4.11 Bibliotheken (Packages)

Packages werden in Kapitel **13 der MLS** beschrieben.

### 10.4.12 Annotationen

**`Annotationen`** werden in **Kapitel 17 der MLS** beschrieben.

In SimulationX werden Annotationen zur Versionsbehandlung nicht unterstützt.

SIMULATION X®

Powered by ITI

FAQ

# FAQ

# 11 FAQ – Häufig gestellte Fragen

## 11.1 Allgemein

- **Wie kann ich zur Beantwortung technischer Fragen mit ITI in Kontakt treten?**

  Wann immer Fragen und Probleme auftreten, können Sie sich auf eine schnelle und professionelle Antwort des ITI-Support-Teams verlassen. Der einfachste und schnellste Weg ist die Nutzung des ITI-HelpDesk Online, der für jeden offen ist. Alternativ senden Sie eine Email an support@itisim.com. Egal für welchen Weg Sie sich entscheiden, wir werden Ihnen so schnell, wie möglich helfen.

- **Wie registriere ich mich für den ITI-HelpDesk?**

  Um den ITI-HelpDesk zu nutzen, registrieren Sie sich unter folgendem Link http://helpdesk.itisim.com. Sobald Ihre Registrierung abgeschlossen ist, erhalten Sie umgehend einen Benutzernamen und ein Passwort, dass Sie für die Anmeldung am HelpDesk benötigen.

- **Ist es möglich jemanden im Support-Center direkt zu kontaktieren?**

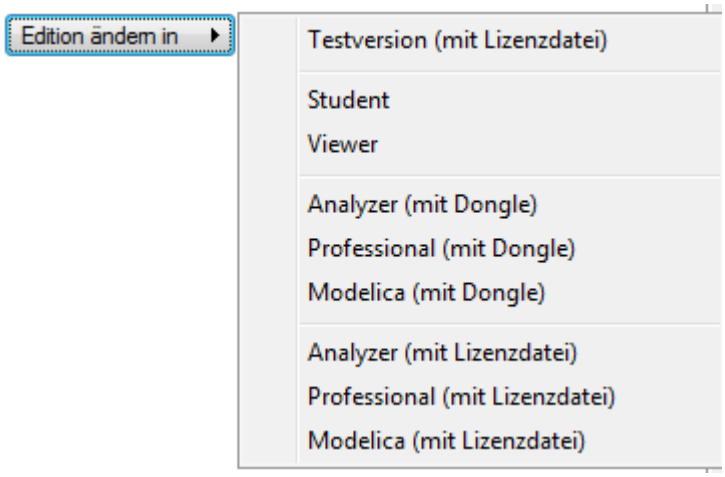
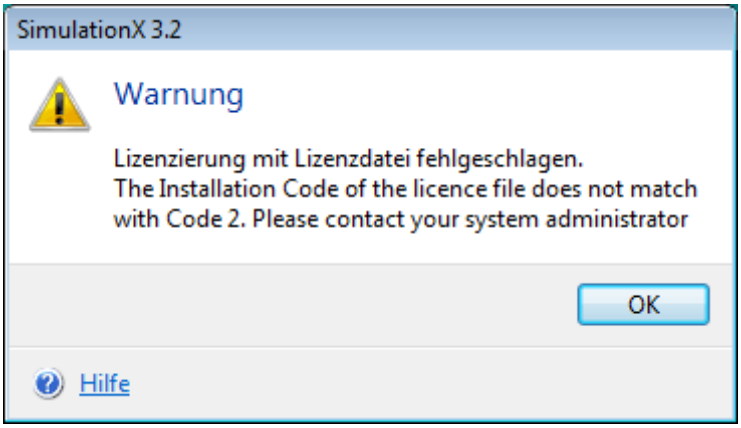
  Ja, es gibt die eine oder andere Frage, die eine schnelle Antwort oder eine individuelle Diskussion erfordert. Für Nutzer innerhalb Europas stehen die IT-Ingenieure von Montag bis Freitag in der Zeit von 8:00 bis 17:00 Uhr MEZ zur Verfügung.

  - Europe:
    Für Support-Fragen, rufen Sie bitte unter +49 (0)351/260 50 – 151 an.
  - America:
    Für Support-Fragen, rufen sie bitte unter ++1 (734)620-9991 an.

- **Kann ich Unterstützung in meinem Land bekommen?**

  Ja. Wir haben eine große Zahl von autorisierten Partnern auf der ganzen Welt, die technische Unterstützung anbieten. Um Ihren Partner vor Ort zu erfahren, schauen Sie bitte in unserer Partner-Liste unter http://www.iti.de/kontakt/iti-global/distributoren.html nach.

## 11.2 Zur Installation/Lizenzierung

| Problem | Ursache/Hilfe |
|---|---|
| ***Ich habe die Student Edition genutzt und möchte jetzt eine andere Edition (Test, Professional, etc.) nutzen.*** | Starten Sie SimulationX in der Student Edition. Gehen Sie in das Menü ***Extras/Optionen*** und wählen Sie das Register ***Lizenzierung***. Betätigen Sie die Schaltfläche ***Edition ändern in…*** und wählen Sie die gewünschte Edition aus.<br>  <br>Schließen Sie SimulationX. Nach einem erneuten Start werden Sie entsprechend der gewünschten Edition zur Auswahl eines Lizenzmediums – Key/Dongle oder Lizenzdatei – aufgefordert. |
| ***Ich erhalte beim Starten von SimulationX folgende Fehlermeldung.*** |  <br>Bitte prüfen Sie zuerst, ob Sie die Lizenzdatei auf dem Rechner nutzen, von dem aus sie angefordert wurde.<br><br>Des Weiteren sollten Sie immer die Konfiguration des Rechners nutzen, wie sie zum Zeitpunkt der Lizenzanforderung vorhanden war. Das WLAN und andere netzabhängige Geräte, wie PDA können die vom Programm ausgelesene MAC-Adresse beeinflussen. Meist hilft ein ab- oder anschalten der entsprechenden Geräte, um das Problem zu lösen. Sollte dies nicht zum Erfolg führen, fordern Sie bitte eine neue Lizenzdatei bei ITI an. |

| | |
|---|---|
| ***Ich erhalte beim Starten von SimulationX folgende Fehlermeldung:***<br>***"Keine freie Lizenz vorhanden."*** | Alle Lizenzen dieser Datei werden aktuell genutzt. Die Nutzer werden wie folgt angezeigt (bei Einzelplatzlizenz - der Computername, bei Netzwerklizenz - Nutzer+++Computername). Wählen Sie eine andere Lizenzdatei aus.<br>Wenn kein Nutzer vorhanden ist und Sie diese Meldung trotzdem erhalten, so kann das Lizenzfile beschädigt sein. Überschreiben Sie das vorhandene Lizenzfile mit der vorher erstellten Sicherheitskopie. |
| ***Ich sehe neu erworbene Bibliotheken nicht.*** | Überprüfen Sie unter „***Extras/Optionen***“ im Register „***Bibliotheken und Module***“, ob die entsprechenden Bibliotheken für eine Nutzung aktiviert sind. Setzen Sie bei Bedarf den entsprechenden Haken. |
| ***Beim Starten von SimulationX wird kein Dongle erkannt, obwohl er angesteckt ist.*** | Für dieses Problem gibt es mehrere Ursachen.<br>1. Es wird kein Dongle erkannt weil die Schutzsoftware **nicht installiert** wurde. Rufen Sie das Setup von SimulationX erneut auf und installieren die Schutzsoftware.<br>2. Der Dienst für die Schutzsoftware läuft nicht. Das können Sie unter „***Start***“ -> „***Systemsteuerung***“ -> „***Verwaltung***“ -> „***Dienste***“ überprüfen. Hier muss der Dienst „**Sentinel Protection Server**“ gestartet sein. Ist dies nicht der Fall starten Sie diesen Dienst.<br><br>Lässt sich dieser Dienst nicht starten und auf dem Computer läuft andere Software, welche auch den Sentinel Softwareschutz nutzt (z.B. NI Software), so hilft der Workaround, den Sie auf der Installations-CD im Verzeichnis Dongle finden. Doppelklick auf die Datei NIWorkAround.reg nimmt die entsprechenden Einträge in der Registry vor. Durch einen Neustart des Computers wird der Dienst aktiviert. |

## 11.3 Zur Simulation/Berechnung

Alle Handlungen zur Beseitigung von Fehlern, die zur entsprechenden Meldung führten, sind mit dem Zeichen ➢ markiert. Da in vielen Fällen die Ursachen für die Meldung (Fehler, Warnung, Problem) sehr komplex sind, sollte der komplette Text zur entsprechenden Fehlermeldung gelesen werden, bevor mit der Eingrenzung der Ursachen begonnen wird.

### 11.3.1 Interne Löser (BDF und MEBDF)

Das folgende Kapitel beschreibt den Umgang mit Fehlermeldungen, Warnungen und bestimmten Problemen in den internen Lösern (BDF und MEBDF). Es ist eine Erweiterung von **Kapitel 6.9.**

Für die Fehlersuche und Problembehebung ist es hilfreich, im **Kapitel 6** die folgenden Abschnitte

- **6.1.2 Ablauf der Berechnung im Zeitbereich,**
- **6.1.3 Verfügbare Solver,**
- **6.4 Gleichgewichtszustand (statisch/stationär) oder**
- **6.9 Troubleshooting: Was tue ich, wenn...**

nachzulesen sowie die verwendeten Begriffe gemäß **Kapitel 12** zu kennen.

Für dieses Kapitel gelten folgende Festlegungen:

- Werden Tracingausgaben aktiviert, so geschieht das entsprechend **Kapitel 6.1.1.4**. Es wird nur noch die jeweilige Rubrik angegeben, z.B. ***Solver/ Zustandsgrößen***.
- Der im **Kapitel 6.1.1.4** beschriebene Umgang mit der Variable **`traceOn`** wird vorausgesetzt.
- Im Menü ***Simulation/Einstellungen Transient…*** auf der Dialogseite ***Solver*** gibt es die Option ***Weitere Parameter…*** für erweiterte Einstellungen (siehe **6.1.1.5**). Die hier einstellbaren Variablen sind im Text hervorgehoben (z.B. **`dblResRelError)`** .

| **Fehlermeldung / Verhalten** | **Ursache/Hilfe** |
|---|---|
| ***Warnung: Es gibt Parameter, die keinen Wert haben.*** | Es gibt Parameter, deren Eingabefelder leer sind. Diese werden mit dem Wert null belegt.<br>➢ Diese Warnung können Sie ignorieren, wenn Sie sicher sind, dass für alle unbelegten Parameter der Wert null eingesetzt werden darf. Wenn Sie die Meldung expandieren, werden alle betreffenden Parameter angezeigt. |

| | |
|---|---|
| ***Warnung: Das System ist singulär.*** | Diese Warnung deutet oft auf einen Modellierungsfehler oder ungünstigen Modellaufbau hin. Sie wird angezeigt, wenn die (quadratische) Jacobimatrix (ebenso viele Gleichungen wie Zustandsgrößen) des Modells strukturell singulär ist.<br>Das ist der Fall, wenn:<br>1. Nullzeilen auftreten<br>In die Gleichung gehen keine Zustandsgrößen und keine Ableitungen ein.<br>Dieser Fall ist gegeben, wenn elektrische Verbindungen über ideale Schalter abgeschaltet werden, sie also „in der Luft“ hängen.<br>2. Nullspalten auftreten<br>Es gibt für die betreffende Zustandsgröße und deren Ableitung keine Gleichung.<br>3. Spalten mit nur einem Nichtnulleintrag doppelt auftreten, d.h. für zwei Zustandsgrößen bzw. deren Ableitungen gibt es nur eine Gleichung.<br>Dieser Fall tritt am häufigsten auf. Gründe dafür sind:<br>- parallel geschaltete Zwangsbedingungen (stattdessen nur eine Zwangsbedingung verwenden),<br>- parallel geschaltete Reibstellen (nur eine Reibstelle mit der Summe der Haft- bzw. Gleitreibkräfte verwenden),<br>- starrer Anschlag parallel zur starren Reibstelle (stattdessen die Reibung im Anschlag berücksichtigen)<br>- überbestimmte Mehrkörpermodelle,<br>- parallel geschaltete Spannungsquellen<br>- in Reihe geschaltete Stromquellen.<br>Für die Analyse:<br>In der expandierten Trace-Nachricht wird zu jedem singulären Anteil folgendes angezeigt:<br>Eine Anzahl von Zustandsgrößen und eine kleinere Anzahl von Gleichungen, wobei die Zustandsgrößen und deren Ableitungen nicht in die anderen Gleichungen eingehen.<br>oder<br>Eine Anzahl von Gleichungen und eine kleinere Anzahl von Zustandsgrößen, wobei alle übrigen Zustandsgrößen |

| | |
|---|---|
| | und deren Ableitungen nicht in diese Gleichungen eingehen.<br>oder<br>Bei Gleichheit wird beides, gekennzeichnet mit (a) und (b), angezeigt.<br><br>Bei Nullspalten werden keine Gleichungen, bei Nullzeilen keine Zustandsgrößen angezeigt.<br>➢ Dies können Sie nachvollziehen, wenn Sie sich mit ***Solver/Jacobi-Matrix*** die beiden Jacobimatrizen dF/dY und dF/dYP unmittelbar vor der Warnung anzeigen lassen. Dabei sind die beiden Matrizen als „übereinandergelegt“ zu betrachten. Das bedeutet, eine Nullzeile bzw. Nullspalte enthält in beiden Matrizen nur Nullelemente. (Ausgabe mit **`traceOn`** einschränken)<br>Außerdem können Sie die beteiligten Zustandsgrößen im Modell suchen (siehe **Kapitel 6.1.1.4**).<br>➢ Welche Gleichungen sich hinter dem jeweiligen Namen der Gleichung verbergen, kann durch Anzeige der symbolischen Jacobimatrix bei der symbolischen Analyse ermittelt werden (***Gleichungen/Symbolische Jacobimatrix***).<br>Prinzipiell kann SimulationX singuläre Modelle berechnen, jedoch kann keine Indexberechnung erfolgen. Dadurch können die Anfangswertberechnung sowie die Berechnung über der Zeit insbesondere bei DAEs fehlschlagen.<br>Außerdem treten höhere Rechenzeiten auf.<br>➢ Verändern Sie das Modell, bis die Meldung nicht mehr erscheint! |
| ***Es konnten keine konsistenten Anfangswerte berechnet werden!*** | Dieses Problem tritt auf, wenn<br>1. keine konsistenten Anfangswerte für das Modell existieren,<br>2. bei einer DAE Anfangswerte vorgegeben und fixiert wurden, die nicht konsistent sind,<br>3. der Algorithmus fehlschlägt.<br><br>Beispiele für Fall 1 sind einander widersprechende Gleichungen oder nichtlösbare Gleichungen wie $x = x + 1$ oder $x^2 + 1 = 0$. |

Fall 2 liegt vor, wenn für (über Zwangsbedingungen; z.B. starres Getriebe) verknüpfte Connections beidseitig fixierte Anfangswerte vorliegen, die sich widersprechen (nicht der Getriebeübersetzung entsprechen). Ein anderes Beispiel ist $x^2 - 1 = 0$ mit fixiertem Anfangswert ungleich 1 oder -1.

Fall 3 tritt bei $x^2 - 1 = 0$ mit einem nichtfixierten Startwert 0 auf, solange die symbolische Analyse dies nicht auflöst oder die Jacobimatrix nicht numerisch berechnet wird. In diesem Fall lautet das linearisierte Gleichungssystem $2 \cdot 0 \cdot (x - 0) - 1 = 0$. Dieses ist nicht lösbar.

In den Details der Trace-Meldungen werden die Gleichungen angezeigt, die bei der Anfangswertberechnung nicht erfüllt werden bzw. die Größen, die nicht konvergieren. Das Problem liegt zumeist in der näheren Umgebung.

Zuerst sollte überprüft werden, ob ein Modellierungsfehler vorliegt! Danach prüft man, ob es sich bei dem Modell um eine DAE handelt. Dazu lässt man sich die Indizes der Zustandsgrößen anzeigen (***Solver/Indizes***).

- Hat eine Zustandsgröße einen Index größer null, handelt es sich um eine DAE. Prüfen Sie, ob in der Umgebung der nichterfüllten Gleichung Anfangswerte fixiert sind, die der Gleichung widersprechen. Tragen Sie dort entweder konsistente Werte ein oder schalten Sie die Anfangswerte auf frei.

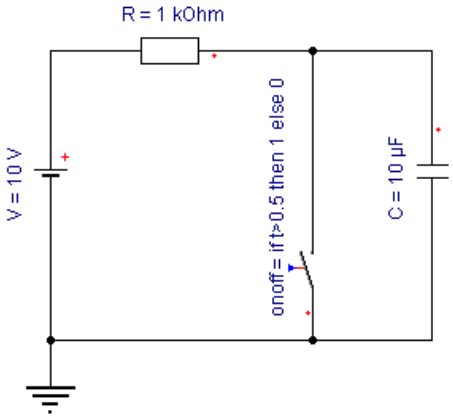
Führt dies nicht zum Erfolg, kann es sein, dass der Algorithmus zur Berechnung konsistenter Anfangswerte fehlschlägt. Das tritt bei zu kleinen oder zu großen Schrittweiten auf.

- Versuchen Sie zuerst, die minimale Rechenschrittweite **`dtMin`** zu verkleinern. Hilft das nicht, vergrößern Sie sie wieder und weichen stattdessen die Fehlerschranken auf. Weitere Hinweise finden Sie unter **6.8.2.**

Um zu sehen, ob zu große Residuen die Ursache für die fehlgeschlagene Anfangswertberechnung sind, expandieren Sie die Fehlermeldung. Wenn zu große Residuen die Ursache für die fehlgeschlagene Anfangswertberechnung sind, kann durch Aktivieren von ***Solver/Residuen*** angezeigt werden, ob die Residuen sich regelmäßig oder unregelmäßig sprunghaft ändern oder ob sie sich nur wenig oder überhaupt nicht ändern.

| | |
|---|---|
| | 1. Im letzteren Fall kann durch Aktivieren von ***Solver/ Fehler in den Residuen durch die Lösung des linearen Systems*** festgestellt werden, ob dies an der Nichtlösbarkeit des Gleichungssystems liegt.<br>Eine Erklärung dafür kann durch Anzeige der Jacobi-Matrizen (***Solver/Jacobi-Matrix***) gewonnen werden. Nichtlösbarkeit durch fixierte nicht konsistente Anfangswerte lässt sich häufig daran erkennen, dass es mindestens ein Residuum außerhalb des Toleranzbereiches gibt, in das keine veränderliche Zustandsgrößen oder deren Ableitungen eingehen ("()" wird angezeigt und es erscheint die Meldung "50.Schritt der Anfangswertberechnung der DAE: Lineares Gleichungssystem nicht lösbar."). Das gleiche Verhalten tritt auf, wenn nichtfixierte Größen zwar eingehen (in symbolischer Jacobimatrix (***Gleichungen/Symbolische Jacobimatrix*** und ***Gleichungen/Symbolische Jacobimatrix nach den Ableitungen***) steht nicht null), aber die Ableitung nach ihnen (***Solver/Jacobi-Matrix***) aktuell 0 ist.<br>➢ Beseitigen Sie die falschen Fixierungen bzw. wählen Sie andere Startwerte.<br>➢ Wenn das lineare Gleichungssystem nicht lösbar ist, aber die Werte akzeptiert werden sollen, können die Residuentoleranzen (Menü ***Simulation/ Einstellungen Transient…*** Dialogseite ***Solver***, Schaltfläche ***Weitere Parameter…***, **`dblEquaAbsTolFacForDer`** und **`dblEquTol)`** oder der Faktor zur Bestimmung des unvermeidlichen Fehlers bei der Berechnung der Residuen und damit der Mindesttoleranz der Residuen **`dblResRelError`** vergrößert werden.<br>➢ Wenn nichtfixierte Größen zwar eingehen, aber die Ableitung nach ihnen aktuell 0 ist, kann die Verwendung der numerisch berechneten Jacobi-Matrix helfen. Es kann auch die Einbettung (siehe **6.1.2.3**) verwendet werden, um den singulären Punkt verlassen zu können.<br>2. Bei nur geringer Änderung in Richtung null kann eine Dämpfung die Ursache sein. Dies lässt sich durch Anzeige der Newtonkorrekturen klären (***Solver/Newton-Korrektur***). In der letzten Zeile stehen die Dämpfungsfaktoren. Wenn sie kleiner als 1 sind, liegt die Ursache im versuchten Verlassen des zulässigen Bereichs (***Solver/ Überschreitung von Minimum/Maximum der*** |

| | |
|---|---|
| | ***Zustandsgrößen***).<br>➢ In diesem Fall ist analog zu ***Überschreitung von Minimum/ Maximum der Zustandsgrößen*** zu überprüfen, ob dies das korrekte Verhalten des Modells ist, durch einen Fehler in den Elementen verursacht wird oder es an der Berechnung der Iterationsschritte liegt. Für die Analyse können im Ausgabebereich die Zustandsgrößen (***Solver/Zustandgrößen***) und Newtonkorrekturen (***Solver/Newton-Korrektur***) angezeigt werden.<br>➢ Wenn das Verhalten dem Modellinhalt entspricht, so muss das Modell geändert werden (z. B. Hinzufügen oder Entfernen von Elementen, Ändern von Startwerten oder Modellparametern).<br>➢ Ist das Verhalten des Modells korrekt, sind im Eigenschaftsdialog (siehe **Kapitel 5.3.1**) dieser Zustandsgröße **`min`**, **`minNotReached`**, **`max`** bzw. **`maxNotReached`** entsprechend zu verändern, damit die Lösung im zulässigen Bereich bleibt.<br>3. Bei regelmäßiger sprunghafter Änderung sind Sprünge in der Residuenberechnung oder Auslöschungsfehler möglich. Diese können durch Aktivierung von ***Gleichungen/ Berechnung der Gleichungen*** nachvollzogen werden.<br>4. Unregelmäßige sprunghafte Änderungen weisen dagegen auf Startwerte außerhalb des Konvergenzbereichs hin.<br>➢ Welche Gleichungen sich hinter dem jeweiligen Residuum verbergen, kann durch Anzeige der symbolischen Jacobimatrix bei der symbolischen Analyse ermittelt werden (***Gleichungen/Symbolische Jacobimatrix***).<br>➢ Möglicherweise hilft in diesem Fall die Verwendung einer geeigneten Einbettung (siehe **6.1.2.3**) zur Lösungsfindung.<br>➢ Es ist auch möglich, dass die Werte zu Beginn der Eventiteration zum Startzeitpunkt nicht sinnvoll sind. In diesem Fall können im Menü ***Simulation/ Einstellungen Transient…*** auf der Dialogseite ***Solver***, Schaltfläche ***Weitere Parameter…***, mit **`iSetValuesToNominal`** Zustandsgrößen und Ableitungen ohne fixierte Startwerte auf ihre Nominalwerte gesetzt werden. |

| | |
|---|---|
| ***Kann nach einer Unstetigkeit keinen konsistenten Zustand herstellen.*** | Ein häufiger Modellierungsfehler ist, dass sich zu Ereignissen Zustandsgrößen sprunghaft ändern sollen, die nicht sprungfähig sind. Deren Werte werden bei der Reinitialisierung vom Algorithmus als fixiert erkannt. Beispiele dafür sind sprunghafte Änderungen der Spannung über Kondensatoren (**Bild 11.1**), die nur mit unendlichen Strömen realisiert werden können.<br>  <br><br>**Bild 11.1:** Fehlerhafte Modellierung: Ein Kondensator wird mittels idealem Schalter ohne Widerstand kurzgeschlossen (wenn die Spannung in der oberen rechten Verbindung sich sprunghaft ändern darf, verläuft die Rechnung erfolgreich)<br><br>Sprunghafte Änderungen des Übersetzungsverhältnisses starrer Übersetzungen führen zu Sprüngen im Winkel oder der Winkelgeschwindigkeit. Das ist physikalisch unmöglich!<br><br>Die Aussagen zu ***Es konnten keine konsistenten Anfangswerte berechnet werden!*** gelten auch hier. Da das Problem meist während der Rechung auftritt, sollte die Ausgabe durch Verkleinerung des Zeitintervalls mittels **`traceOn`** eingeschränkt werden (z. B. time>0.56). Man wählt dazu am besten einen Zeitpunkt kurz vor dem Auftreten des Fehlers.<br>➢ Die Beseitigung des Problems hängt von der Ursache ab. Wenn sich Zustandsgrößen sprunghaft ändern dürfen, kann dies durch **`notFixed=true`** im Attributdialog (siehe **Kapitel 5.3.1**) ermöglicht werden. |
| ***Anfangswertberechnung abgebrochen*** | Diese Meldung wird ausgegeben, wenn die Rechnung während einer Anfangswertberechnung angehalten wurde (Menü ***Simulation/Stopp*** bzw. betätigt). |

| | |
|---|---|
| ***Der berechnete Anfangszustand entspricht nicht meinen Erwartungen…*** | ➢ Überprüfen Sie, ob Sie Ihre gewünschten Anfangswerte inklusive der Maßeinheiten eingegeben und fixiert haben (siehe Abschnitt **6.1.2.2,** Bild 6.5).<br>Freie Anfangswerte dienen lediglich als Startwerte für den iterativen Berechnungsalgorithmus. Sie werden vom Programm im Allgemeinen geändert. |
| ***Fehler: Zyklus in der Event-Iteration gefunden.*** | ➢ Expandieren Sie die Fehlermeldung, um Hinweise auf die beteiligten diskreten Veränderlichen, einschließlich der Vorzeichen der Zerofunctions, zu erhalten.<br>➢ Wenn Ihnen das nicht ausreicht und Sie deshalb weitere Informationen benötigen und vor dieser Event-Iteration sehr viele Unstetigkeiten oder Zeitschritte existieren, sollte die Ausgabe durch Verkleinerung des Zeitintervalls im `traceOn` eingeschränkt werden.<br>➢ Wenn Sie sich alle ändernden diskreten Variablen und die Zerofunctions mit Vorzeichenänderung anzeigen lassen wollen, aktivieren Sie in den Einstellungen für das Tracing die Punkte ***Variable/Änderung von Werten diskreter Variablen*** und ***Unstetigkeiten/ Vorzeichenänderung von Zerofunctions.***<br>➢ Es sollten die Umschaltbedingungen für die diskreten Variablen, einschließlich der Vorzeichen der Zerofunctions, untersucht werden, um die Ursache für den Zyklus zu finden. Dazu ist ***Gleichungen/Berechnung der Event-Schritte*** zu aktivieren. In den so erhaltenen Ausgaben, können die schaltenden diskreten Veränderlichen, einschließlich der Zerofunctions, insbesondere der am Zyklus beteiligten, gesucht werden. So können auch die dabei verwendeten Veränderlichen und rückwärts die eingehenden Zustandsgrößen gesucht und die Änderung ihrer Werte zwischen den einzelnen Event-Schritten betrachtet werden.<br>➢ Wenn das zu kompliziert ist, können die Zustandsgrößen und Ableitungen durch Aktivieren von ***Solver/ Zustandsgrößen*** und die Newtonkorrekturen von ***Solver/Newton-Korrektur*** angezeigt werden. So ist erkennbar, welche Größen sich während der Anfangswertberechnung, also durch die Änderung des diskreten Zustands, ändern. Nach jeder Anfangswertberechnung stehen die Zustandsgrößen und Ableitungen auf den Werten, die zum Umschalten führen. |

| | |
|---|---|
| | ➢ Die Änderungen der Zustandsgrößen und Ableitungen ergeben sich aus den durch die Umschaltung des diskreten Zustands ergebenden Änderungen der Residuen. Um zu sehen, welche Residuen sich ändern und deshalb durch den Löser bei der Anfangswert-berechnung korrigiert werden müssen, ist ***Solver/ Residuen*** zu aktivieren.<br>➢ Wenn dann die Fehlerursache und -beseitigung nicht klar ist, ist die Genauigkeit (der entsprechenden Zustandsgrößen) zu erhöhen. Dazu sind deren Toleranzen (und bei Bedarf die minimale Rechen-schrittweite **`dtMin`**) zu verringern. Dadurch verkleinern sich die Beträge der Residuen nach der Anfangswertberechung und die Veränderlichen liegen näher an ihren wahren Werten dran. Solange das nicht der Fall ist, ist dies fortzusetzen. Deshalb kann mit der Verkleinerung der Toleranzen um eine Zehnerpotenz angefangen werden.<br>➢ Wenn die Schaltbedingungen der am Zyklus betei-ligten schaltenden diskreten Veränderlichen, ein-schließlich der Zerofunctions, nicht oder nicht entsprechend stark in Richtung Nichtschalten während eines Event-Schritts innerhalb des Zyklus geht, weist das darauf hin, dass der Zyklus auch bei einem exakt gelösten Gleichungssystem auftritt. In diesem Fall gibt es Probleme bei den Umschaltbedingungen für die diskreten Variablen bzw. bei den Zerofunctions oder durch ungünstig gewählte Gleichungen im Gleichungs-system. Dann ist das Modell an diesen Stellen zu ändern. |
| ***Die geforderte Genauigkeit konnte nicht erreicht werden…*** | Diese Meldung weist meist darauf hin, dass im Modell vor-handene hohe Eigenfrequenzen oder kleine Zeitkonstanten nicht genau genug berechnet werden können. Wenn man diese Eigenschaften im Modell nicht benötigt, führt eine Eliminierung meist zu einer schnelleren Berechnung.<br>➢ In der Fehlermeldung wird auch ausgegeben, welche Zustandsgröße das Problem verursacht. Sie können diese Information nutzen, um eventuell das Modell zu verändern.<br>➢ Verringern Sie zuerst die minimale Rechenschrittweite **`dtMin`** (Menü ***Simulation/Einstellungen Transient…/Allgemein***). Falls das nicht zum Erfolg führt, ändern Sie die Fehlerkriterien leicht. Zu große |

| | |
|---|---|
| | Fehlerkriterien können aber zu unexakten Ergebnissen führen.<br>➢ In bestimmten Fällen hilft auch die Einstellung eines anderen Solvers (***Simulation/Einstellungen Transient…/Solver***, unter ***Verfahren*** entweder ***BDF-Verfahren*** oder ***MEBDF- Verfahren*** auswählen).<br>Wenn sehr viele nichtgeklammerte Zustandsgrößen in der Liste stehen, können zuerst die mit den größten relativen Änderungen untersucht werden. Sie können aber sich auch von allen beteiligten Größen die Ergebnisverläufe in Ergebnisfenstern anzeigen lassen (dazu die Protokollierungsschrittweite (Menü ***Simulation/Einstellungen Transient…/Allgemein***, **`dtProtMin`**) genügend klein wählen).<br>➢ Wenn eine Verkleinerung der Zeitschrittweite nicht zu einer Verringerung des Fehlers führt, kann dies bedeuten, dass die entsprechende Zustandsgröße an der Stelle einen Sprung hat und die entsprechende Unstetigkeit mit noEvent unterdrückt wurde. Wenn die Unstetigkeitsbehandlung notwendig ist, entfernen Sie das noEvent. Wenn sie nicht notwendig ist, können Sie bei an der Oberfläche liegenden Größen diese aus dem Genauigkeitstest herausnehmen. Wählen Sie dazu diese Zustandsgröße im Modell (im Eigenschaftsdialog oder im Modellexplorer). Öffnen Sie den dazugehörigen Attributdialog (siehe **Kapitel 5.3.1**). Setzen Sie **`discontChange`** auf true. |
| ***Fehler: Das Verfahren konvergiert nicht!*** | ➢ Verfahren Sie, wie im vorhergehenden Fall beschrieben.<br>➢ Eine Reduktion der maximalen Verfahrensordnung hilft bei stark nichtlinearen Modellen.<br>➢ Die Vorgehensweise bei zu großen Residuen ist bei ***Es konnten keine konsistenten Anfangswerte berechnet werden!*** und ***Der stationäre Zustand (statisches Gleichgewicht) konnte nicht gefunden werden!*** beschrieben.<br>Für weitere Informationen schauen Sie sich die Gründe für die letzten ungültigen Schritte an (Menü ***Simulation/ Einstellungen Transient…/Tracing*** bei ***Solver/Ungültige Schritte/Details*** aktivieren, **`traceOn`** bei Bedarf ändern, Trace-Meldung expandieren). Wenn Sie sehen wollen, ob es |

| | |
|---|---|
| | dazwischen gültige Schritte gab, aktivieren Sie zusätzlich ***Solver/Gültige Schritte/Details***.<br>Folgende Meldungen können auftreten:<br>– *die geschätzte Konvergenzrate ist …*>**`dblMaxConvRate`** (diese ist über die Schaltfläche ***Weitere Parameter…*** im Menü ***Simulation/ Einstellungen Transient…/Solver*** erreichbar)<br>Durch Aktivieren von ***Newton-Korrektur***, ***Zustands-größen*** und ***Residuen*** im Tracing zum ***Solver*** ist zu ermitteln, welche Newtonkorrekturen wachsen oder beinahe gleich und wesentlich beteiligt (Zustandsgrößen und nichtgeklammerte Ausdrücke bei den Newton-korrekturen betrachten) sind oder welche Residuen wachsen. Um die Ursache für dieses Verhalten zu finden, sind Kenntnisse zum jeweiligen Element notwendig. Wenn die Newtonkorrekturen sich wie die Residuen verhalten, weist dies auf Probleme bei der Residuen-berechnung oder auf einen sehr kleinen Konvergenz-bereich hin. Die Residuenberechnung lässt sich durch Aktivieren von ***Gleichungen/Berechnung der Gleichungen*** nachvollziehen.<br>➢ Bei zu kleinem Konvergenzbereich hilft üblicherweise die Verkleinerung der minimalen Rechenschrittweite **`dtMin`**. Die Residuen im Prädiktorschritt müssen in diesem Fall bei kleinerer Zeitschrittweite (annähernd linear bezüglich der Zeitschrittweite) kleiner werden. Wenn diese klein genug sind, müssten die Residuen in den folgenden Korrektorschritten immer kleiner werden. Wenn dies nicht der Fall ist, kann dies an Problemen mit der Residuenberechnung liegen. Eine häufige Ursache sind Sprünge in der Residuenbe-rechnung, d.h. eine kleine Änderung der eingehenden Größen führt zu großen Änderungen der berechneten Größen. Dies ist bei mit **`noEvent`** ausgeschalteten Unstetigkeiten der Fall. In diesem Fall ist das **`noEvent`** zu entfernen oder ein stetiger Übergang zwischen den Zweigen der Zuweisung / Gleichung zu sichern. Eine andere Ursache für Fehler in der Residuenberechnung sind Auslöschungsfehler. Diese treten z.B. bei der Subtraktion etwa gleichgroßer Zahlen auf. Diese sind durch Änderungen im Gleichungssystem zu vermeiden. |

- *(zu viele) Korrektorschritte, Schritte mit zu großen Residuen, Schritte mit zu großer Newtonkorrektur*
  Die Entwicklung der Korrektor-Iteration kann durch Aktivieren von ***Solver/Newton-Korrektur*** *und* ***Solver/ Residuen*** im Tracing zum ***Solver*** verfolgt werden. Eventuell sollte die Ausgabemenge durch eine Verkleinerung des Zeitintervalls mittels **`traceOn`** eingeschränkt werden. Zum Nachvollziehen des Verhaltens kann die Ausgabe von ***Zustandsgrößen***, ***Ableitungen*** und ***Jacobi-Matrix*** sinnvoll sein. Die Ermittlung der Ursache erfordert jedoch Kenntnisse zum Modell.

  - Wenn nur *Schritte mit zu großen Residuen* und keine *Schritte mit zu großer Newtonkorrektur* vorhanden sind, werden für jeden Iterationsschritt die zu großen Residuen in der Form:
    "Name des Residuums (Korrektorschritt) (Wert des Residuums)" ausgegeben.
    Anschließend erfolgt die Ausgabe, welche Residuen am Ende der Korrektoriteration einen zu großen Betrag haben in der Form:
    "Name des Residuums" "Wert des Residuums" "(eingehende Zustandsgrößen und deren Ableitungen)".
    Dabei erfolgt der Vergleich des Betrags diesen Wertes mit ***Weitere Parameter…***, **`dblEquTol`** (falls positiv) oder der absoluten Toleranz der Gleichung (falls gesetzt) oder **`absTol`** (sonst). Wenn die zur Gleichung zugeordnete Zustandsgröße eine Ableitung hat, die benötigt wird (z.B. Ergebnisvariable ist), aber nicht in das Gleichungssystem eingeht, wird diese Toleranz noch mit ***Weitere Parameter…***, **`dblEquaAbsTolFacForDer`** multipliziert. Sollte allerdings der unvermeidliche Fehler bei der Berechnung dieser Gleichung relativ groß sein, wird die Fehlerschranke unter Berücksichtigung von **`dblResRelError`** vergrößert. Dann untersucht man durch Aktivieren von ***Gleichungen/ Berechnung der Gleichungen***, ***Gleichungen/ Berechnung der partiellen Ableitungen***, ***Solver/ Newton-Korrektur*** und ***Solver/Residuen,*** warum die Residuen nicht klein genug werden. Hier kann dann korrigiert werden. Wenn dagegen die Werte akzeptiert werden sollen, sind die überschrittenen Toleranzen bzw. **`dblResRelError`** so zu vergrößern, dass die

| | |
|---|---|
| | Gleichungstoleranzen größer als die Beträge der Werte der entsprechenden Residuen sind.<br>– *zu große Änderung der Zustandsgrößen*<br>➢ tritt beim MEBDF-Verfahren auf und ist analog dem vorherigen Anstrich zu behandeln. Die Residuen-toleranzen werden hier nicht verwendet.<br><br>Manchmal kommt es vor, dass sich der Wert einer Zustands-größe während des Newton-Schrittes nicht ändert, obwohl der Betrag der zugehörigen Newtonkorrektur (***Solver/ Newton-Korrektur***) größer als 1e-15 * Betrag der Zustands-größe (***Solver/ Zustandsgrößen***) ist. In diesem Fall ist die zugehörige Ableitung (***Solver/Ableitungen***) gleich null und steht außerdem auf der linken Seite einer Gleichung (z. B. ***Gleichungen/Symbolische Jacobimatrix***).<br>➢ Hier wirkt eine Sonderbehandlung für gewöhnliche Differentialgleichungen: Wenn der Betrag des Wertes auf der rechten Seite kleiner als der Solverparameter **`dblExplicitOdeIs0`** (Standardwert 1e-20, Schalt-fläche ***Weitere Parameter…***, siehe **6.1.1.5**) ist, wird die linke Seite gleich null gesetzt.<br>➢ Schwankt die rechte Seite um den Wert null, obwohl sie eigentlich exakt null wäre (z. B. als numerische Differenz zweier gleich großer Werte), sollte der Parameter **`dblExplicitOdeIs0`** nicht geändert werden. Dann verhindert er unnötig kleine Zeitschritte.<br>➢ Wenn allerdings die rechte Seite eine Zeitlang null war und dann ihr Betrag langsam größer wird (wie z.B. die Geschwindigkeit bei einem Anfahrprozess), ist **`dblExplicitOdeIs0`** auf eine negative Zahl zu setzen. Sonst kann diese Sonderbehandlung zu ungül-tigen Schritten führen. |
| ***Lineares Gleichungssystem nicht lösbar und Verfahrensordnung und Schrittweite nicht verkleinerbar*** | Auch hier sollte die Ausgabe durch eine Verkleinerung des Zeitintervalls mittels **`traceOn`** eingeschränkt werden. Genauere Informationen über die Ursache erhält man durch Aktivierung der entsprechenden Ausgaben.<br>Bei ***Solver/Ungültige Schritte/Details*** wird eine Gleichung angezeigt, die nicht lösbar ist. Bei ***Solver/Fehler in den Residuen durch die Lösung des linearen Systems*** ist zu erkennen, welche linearisierte Gleichungen durch den Gleichungslöser nicht richtig gelöst werden können. Mit ***Solver/Residuen*** und ***Solver/Jacobi-Matrix*** lässt sich das zu |

| | |
|---|---|
| | lösende linearisierte Gleichungssystem in den einzelnen Iterationsschritten anzeigen. Welche Gleichung sich hinter dem Gleichungsnamen verbirgt und welche Einträge in der Jacobimatrix immer null sind, ist mit ***Gleichungen/ Symbolische Jacobimatrix*** und ***Gleichungen/Symbolische Jacobimatrix nach den Ableitungen*** ermittelbar. So ist feststellbar, warum das Gleichungssystem singulär und nicht lösbar ist.<br>➢ Bei kleinen Einträgen in der Jacobimatrix oder schlechter Kondition ist bei Bedarf die Pivotschranke für den entsprechenden linearen Gleichungslöser (Schaltfläche ***Weitere Parameter…***, **`dblPivGaussNotZero`**, siehe **6.1.1.5** bzw. **`dblPivSparseNotZero`**, siehe **6.1.1.5**) zu verkleinern.<br>➢ Es kann auch die Schranke für die rechte Seite (Schaltfläche ***Weitere Parameter…***, **`dblRightSideGaussNotZero`**, siehe **6.1.1.5**) vergrößert werden, falls die Lösung akzeptiert werden soll.<br>➢ Eine Modelländerung ist jedoch der bessere Weg. |
| ***Überschreitung von Minimum/ Maximum der Zustandsgrößen*** | *Prädiktor, Maximum der Zustandsgröße … überschritten* bzw. *Minimum der Zustandsgröße … unterschritten*<br>Es wird angezeigt, welche Größe im Prädiktorschritt welchen Wert annimmt und wie groß die überschrittene Grenze ist.<br>Zur Beurteilung, ob dies das korrekte Verhalten des Modells ist, durch einen Fehler in den Elementen verursacht wird oder es an der Berechnung des Prädiktorschrittes liegt, ist eine genaue Kenntnis des Modells notwendig.<br>➢ Für die Analyse sollte die entsprechende Größe in einem Ergebnisfenster dargestellt werden. Eventuell ist die Auflösung der Protokollierung hoch zu wählen (**`dtProtMin = dtMin`**, unbedingt nach der Analyse wieder zurücksetzen).<br>➢ Wenn das Verhalten dem Modellinhalt entspricht, so muss das Modell geändert werden. Das ist z.B. der Fall, wenn in einem Hydraulikkreis eine Pumpe in ein geschlossenes Volumen fördert.<br>➢ Ist das Verhalten des Modells korrekt, sind im Attributdialog (siehe **Kapitel 5.3.1**) dieser Zustandsgröße **`min`**, **`minNotReached`**, **`max`** bzw. |

| | |
|---|---|
| | **maxNotReached** entsprechend zu verändern, damit diese Lösung im zulässigen Bereich bleibt.<br>➢ Wenn dieser Wert nicht erreicht werden darf, ist das Modell entsprechend zu ändern (z.B. Änderung von Startwerten oder Modellparametern).<br><br>Wenn der Prädiktorschritt zu weit geht, kann eine Verkleinerung der minimalen Rechenschrittweite **dtMin** (Menü ***Simulation/Einstellungen Transient…/Allgemein***) hilfreich sein. |
| ***Fehler: Ausnahme während der Berechnung…*** | Während der Rechnung (im Regelfall bei der Residuenberechnung, manchmal bei der Berechnung der partiellen Ableitung oder im DAE-Löser) trat eine Ausnahme auf, die zum Rechnungsabbruch führte. In den ersten beiden Fällen ist die Ursache relativ leicht ermittelbar. Dazu ist die Ausgabe durch **traceOn** einzuschränken. Dann können durch Aktivierung der entsprechenden Ausgaben die Residuenberechnungen (***Gleichungen/ Berechnung der Gleichungen***) bzw. die Berechnungen der partiellen Ableitungen (***Gleichungen/ Berechnung der partiellen Ableitungen***) angezeigt werden. Wenn die letzte Ausgabe geöffnet (expandiert) wird, steht ganz unten die Fehlerursache. Diese lässt sich dann rückwärts verfolgen.<br>Zur Gliederung der Ausgabe können auch einige Informationen des Lösers (z.B. ***Solver/Ungültige Schritte/ Details***, ***Solver/Gültige Schritte/Details***, ***Solver/Zustandgrößen*** oder ***Solver/Residuen***) mit ausgegeben werden.<br>➢ Meist ist eine Änderung der Modellgleichungen erforderlich. In einigen Fällen liegt dieser Fehler daran, dass bestimmte Zustandsgrößen unsinnige Werte bekommen. In diesem Fall hilft die Einschränkung des zulässigen Bereichs. Dazu ist im Attributdialog (siehe **Kapitel 5.3.1**) dieser Zustandsgröße **min**, **minNotReached**, **max** bzw. **maxNotReached** entsprechend zu verändern. |
| ***Fehler bei der Berechnung des Dämpfungsfaktors für die Newton-Korrektur*** | Das ist ein Folgefehler, wenn während der Korrektoriteration eine Zustandsgröße versucht ihren zulässigen Bereich zu verlassen und dies nicht abgefangen werden kann.<br>➢ Durch Aktivieren ***Simulation/Einstellungen Transient…/Tracing/Solver/Überschreitung von Minimum/Maximum der Zustandsgrößen*** lassen sich die entsprechenden Zustandsgrößen anzeigen. Deshalb |

| | |
|---|---|
| | ist zuerst zu untersuchen, warum diese Zustandsgrößen versuchen, ihren zulässigen Bereich zu verlassen. |
| ***Der stationäre Zustand (statisches Gleichgewicht) konnte nicht gefunden werden!*** | Es ist möglich, dass das Gleichungssystem überhaupt keinen Gleichgewichtszustand hat. Sie können das überprüfen, indem Sie das Modell über einen größeren Zeitraum rechnen lassen. Gehen die Werte aller höchsten Ableitungen (in der Mechanik z.B. die Beschleunigungen, in der Hydraulik die Druckableitungen) gegen null und klingen somit **alle** Schwingungen ab, gibt es wahrscheinlich einen Gleichgewichtszustand.<br>Es gibt aber auch periodische schwingende Systeme, die Gleichgewichtszustände besitzen (z.B. ungedämpfter Ein-Massen-Schwinger).<br>In Abschnitt **6.4** (Bild 6.35) sind einfache Modelle dargestellt, die keinen Gleichgewichtszustand haben.<br>➢ Wenn Sie sicher sind, dass es einen Gleichgewichtszustand gibt, kann es sein, dass das Berechnungsverfahren nicht in der Lage ist, ihn zu finden. Bei der Berechnung muss ein nichtlineares Gleichungssystem mit einem speziellen iterativen Verfahren gelöst werden. Als Startwerte für die Iteration werden die eingetragenen Startwerte verwendet, wenn vorher keine Rechnung erfolgte oder das Modell zurückgesetzt wurde. Bei vorher erfolgter Rechnung ohne Abbruch mit Fehlermeldung werden die zuletzt berechneten Werte genutzt.<br>Wenn der gesuchte Zustand zu weit von den Startwerten entfernt liegt, ist es möglich, dass dieser nicht gefunden werden kann. Sie können dann von Hand Startwerte eintragen, die näher an der Lösung liegen. Mit einer kleinen Nebenrechnung oder einer Vorsimulation über einen gewissen Zeitraum gelangen Sie zu brauchbaren Werten.<br>Nach dieser Vorsimulation können Sie die Gleichgewichtsberechnung starten. In diesem Fall sind alle eingehenden zeitabhängigen Größen (z.B. zeitabhängige Kennlinien und Funktionen, sonstige zeitabhängige Vorgaben) auf den Wert zu setzen, der sich aus dem gewünschten Zeitpunkt ergibt, zu dem die Gleichgewichtsberechnung erfolgen soll.<br>➢ Wenn zu große Residuen die Ursache für die fehlgeschlagene Gleichgewichtsberechnung sind, kann durch Aktivieren von ***Solver/Residuen*** angezeigt werden, ob die Residuen sich überhaupt nicht, nur |

| | |
|---|---|
| | wenig, regelmäßig oder unregelmäßig sprunghaft ändern.<br>➢ Um zu sehen, ob zu große Residuen die Ursache für die fehlgeschlagene Gleichgewichtsberechnung sind, expandieren Sie die Fehlermeldung.<br>➢ Welche Gleichungen sich hinter dem jeweiligen Residuum verbergen, kann durch Anzeige der symbolischen Jacobimatrix bei der symbolischen Analyse ermittelt werden (***Gleichungen/Symbolische Jacobimatrix***).<br>➢ Im ersten Fall kann durch Aktivieren von ***Solver/Fehler in den Residuen durch die Lösung des linearen Systems*** festgestellt werden, ob dies an der Nichtlösbarkeit des Gleichungssystems liegt. Eine Erklärung dafür kann durch Anzeige der Jacobi-Matrizen (***Solver/Jacobi-Matrix***) gewonnen werden.<br>➢ Bei nur geringer Änderung in Richtung null kann Dämpfung die Ursache sein. Dies lässt sich durch Anzeige der Newtonkorrekturen (***Solver/Newton-Korrektur***) klären. In der letzten Zeile stehen die Dämpfungsfaktoren. Wenn sie kleiner als 1 sind, kann die Ursache das versuchte Verlassen des zulässigen Bereichs sein (***Solver/Überschreitung von Minimum/ Maximum der Zustandsgrößen***). Wenn das nicht der Fall ist, ist bei der Gleichgewichtsberechnung auch die Sicherung des hinreichenden Abstiegs möglich. Diese kann praktisch ausgeschaltet werden. Dazu sind im Menü ***Simulation/Einstellungen Transient…*** Dialogseite ***Solver***, Schaltfläche ***Weitere Parameter…***, **`dblMinAlpha`** und **`rho`** auf einen Wert etwas kleiner als 1 (z.B. 0.999999) zu setzen.<br>➢ Wenn versucht wird, den zulässigen Bereich zu verlassen, ist analog zu ***Überschreitung von Minimum/ Maximum der Zustandsgrößen*** zu überprüfen, ob dies das korrekte Verhalten des Modells ist, durch einen Fehler in den Elementen verursacht wird oder es an der Berechnung der Iterationsschritte liegt. Für die Analyse können im Ausgabebereich die Zustandsgrößen (***Solver/ Zustandgrößen***) und Newtonkorrekturen (***Solver/ Newton-Korrektur***) angezeigt werden. |

| | |
|---|---|
| | ➢ Wenn das Verhalten dem Modellinhalt entspricht, so muss das Modell geändert werden (z.B. Hinzufügen oder Entfernen von Elementen, Ändern von Startwerten oder Modellparametern).<br>➢ Ist das Verhalten des Modells korrekt, sind im Attributdialog (siehe **Kapitel 5.3.1**) dieser Zustandsgröße **`min`**, **`minNotReached`**, **`max`** bzw. **`maxNotReached`** entsprechend zu verändern, damit die Iteration im zulässigen Bereich bleibt.<br>➢ Bei regelmäßiger sprunghafter Änderung sind Sprünge in der Residuenberechnung oder Auslöschungsfehler möglich. Diese können durch Aktivierung von ***Gleichungen/Berechnung der Gleichungen*** nachvollzogen werden.<br>Unregelmäßige sprunghafte Änderungen weisen dagegen auf Startwerte außerhalb des Konvergenzbereichs hin (siehe oben). Möglicherweise hilft in diesem Fall die Verwendung einer geeigneten Einbettung (siehe **6.1.2.3**) zur Lösungsfindung. |
| ***Berechnung des statischen Gleichgewichts abgebrochen*** | Diese Meldung wird ausgegeben, wenn durch den Nutzer die Rechnung während einer Gleichgewichtsberechnung angehalten wurde (Menü ***Simulation/Stopp*** bzw. betätigt). |
| ***Fehler: Die absolute Toleranz wurde kleiner Null gewählt!*** | Der Wert für die absolute Toleranz **`absTol`** (Menü ***Simulation/Einstellungen Transient…/Allgemein***) darf nicht negativ sein.<br>➢ Setzen Sie **`absTol`** auf einen nichtnegativen, möglichst auf einen positiven Wert.<br>Sollte **`absTol`** bereits nichtnegativ sein, wurde **`absTol`** für die angezeigte Zustandsgröße oder ein Synonym (***Gleichungen/Variablen***) geändert. Lassen Sie sich dazu die "Gleichungen nach der symbolischen Analyse" (***Gleichungen/Gleichungen nach der symbolischen Analyse***) anzeigen. Expandieren Sie die Ausgabe, suchen Sie dort nach **`absTol`** mit negativem Wert, suchen Sie die zugehörige Variable im Modell und ändern Sie **`absTol`** im Attributdialog (siehe **Kapitel 5.3.1**) dieser Veränderlichen. |

| | |
|---|---|
| ***Fehler: Die relative Toleranz wurde kleiner Null gewählt!*** | Der Wert für die relative Toleranz **`relTol`** (Menü ***Simulation/Einstellungen Transient…/Allgemein***) darf nicht negativ sein.<br>➢ Setzen Sie **`relTol`** auf einen nichtnegativen, möglichst auf einen positiven Wert.<br>➢ Sollte **`relTol`** bereits nichtnegativ sein, wurde **`relTol`** für die angezeigte Zustandsgröße oder ein Synonym (***Gleichungen/Variablen***) geändert. Lassen Sie sich dazu die "Gleichungen nach der symbolischen Analyse" (***Gleichungen/Gleichungen nach der symbolischen Analyse***) anzeigen. Expandieren Sie die Ausgabe, suchen Sie dort nach **`relTol`** mit negativem Wert, suchen Sie die zugehörige Variable im Modell und ändern Sie **`relTol`** im Attributdialog (siehe **Kapitel 5.3.1**) dieser Veränderlichen. |
| ***Fehler: Die Anzahl der Zustandsgrößen und die Anzahl der Gleichungen sind verschieden voneinander!*** | Dieser Fehler tritt auf, wenn eines der beteiligten Elemente falsch programmiert wurde, d.h. wenn die Anzahl der Gleichungen in diesem Element nicht mit der Anzahl der Zustandsgrößen übereinstimmt. |
| ***Die Berechnung über der Zeit dauert sehr lange!*** | Lange Rechenzeiten können u. a. folgende Ursachen haben:<br>- (a) Umfangreiche Modelle,<br>- (b) Stark nichtlineare Modelle,<br>- (c) Hohe Eigenfrequenzen im Modell,<br>- (d) Häufig auftretende Ereignisse (Unstetigkeiten),<br>- (e) Rechner mit zu geringer Hardwareausstattung.<br><br>Zu a) Große Modelle sind oft mit höheren Rechenzeiten verbunden.<br>➢ Halten Sie bei der Modellierung folgende Grundsätze ein:<br>Erstellen Sie nicht zu detaillierte Modelle. Passen Sie die Modelle an die Berechnungsaufgabe an!<br>Nutzen Sie die mitgelieferten Bibliotheken für die Modellierung. Die Nachbildung physikalischer Systeme mittels Signalblöcken ist oftmals ineffizient und fehleranfällig.<br>Komplexere Zusammenhänge (Steueralgorithmen, Verhaltensmodelle...) werden häufig mit komplizierten |

Signalstrukturen abgebildet. Oftmals geht das einfacher und effektiver mit Modelica zu programmieren.

Die ITI GmbH hat vielfältige Erfahrungen mit der effizienten Modellierung komplexer Systeme. Greifen Sie auf diese Erfahrungen zurück!

Zu b) Bei hochgradig nichtlinearen Modellen müssen Stellen, an denen sich Anstiege stark ändern, mit geringer Schrittweite abgetastet werden.

- Überprüfen Sie, ob das Modell linearisiert werden kann. Eine andere Möglichkeit besteht in der Modellierung der starken Änderung mit einer Kennlinie, die einen Knick aufweist. Dieses Verfahren wird beispielsweise bei der elektrischen Diode angewendet.

Zu c) Um herauszufinden, ob das Modell hohe Eigenfrequenzen hat, starten Sie die Eigenfrequenzanalyse (siehe **6.5.1**). Solange ein Modellteil mit einer Frequenz in diesem Bereich schwingt, muss mit entsprechend kleinen Schrittweiten gerechnet werden. Erst wenn sie genügend abgeklungen sind, kommen die Vorteile des „steifen“ BDF-Verfahrens zur Wirkung.

- Oftmals können steife Modellteile als ideal starr modelliert oder eine Modellreduktion durchgeführt werden.

Zu d) Ereigniszeitpunkte werden von der Schrittweitensteuerung genau angefahren. Die kontinuierliche Integration wird gestoppt, und das Modell wird reinitialisiert. Alle diese Schritte benötigen Rechenzeit.

- Versuchen Sie deshalb, so wenige Ereignisse wie möglich zu erzeugen. Die auftretenden Ereignisse können mit ***Unstetigkeiten/Vorzeichenänderung von Zerofunctions*** angezeigt werden.

Wenn Sie mit Kennlinien arbeiten, wird bei linearer und Treppeninterpolation bei jeder Stützstelle ein Ereignis erzeugt. Die Anzahl der Ereignisse kann dort reduziert werden, indem:

- die Anzahl der Stützstellen reduziert wird,
- Spline-Interpolation, quadratische, Kreisbogen- oder hyperbolische Approximation benutzt oder
- Keine Behandlung von Unstetigkeiten aktiviert wird (falls möglich)

| | |
|---|---|
| | Zu e) Beachten Sie die Hardwareanforderungen von SimulationX. Falls Sie sehr viele Ergebnisgrößen protokollieren und nur wenig Hauptspeicher verfügbar ist, werden die Ergebnisse während der Rechnung in die Auslagerungsdatei auf Ihre Festplatte (siehe auch **Kapitel 7.6.4**) geschrieben. Die häufigen Zugriffe verlangsamen die Rechnung stark.<br>Sie sollten dann:<br>➢ freien Hauptspeicher schaffen (Menü ***Extras/ Optionen/Speichern*** ),<br>➢ die Anzahl der zu protokollierenden Ergebnisse reduzieren,<br>➢ den Wert für die minimale Ausgabeschrittweite **`dtProtMin`** vergrößern.<br>Weitere Gründe für geringe Rechengeschwindigkeiten sind:<br>- viele Ergebnisfenster, zwischen denen bei der Rechnung gewechselt wird (Aktualisierung bei vielen Protokollierungszeitpunkten erfordert viel Zeit)<br>- viele Momentananzeigen,<br>- viele Tracing-Ausgaben (Ausgabefenster),<br>- Animation komplexer 3D-Modelle auf einer Grafikkarte ohne 3D-Hardwarebeschleunigung.<br>Untersuchung der Ursache rechnungsbedingter langsamer Rechnung:<br>➢ Am Ende der Rechnung den Aufwand ausgeben lassen (***Solver/Auswertung***). Dort erkennt man, ob es viele Unstetigkeiten oder viele ungültige Schritte gibt. Neben der Begrenzung und der Rücksetzung der Zeit durch Zerofunctions und der Begrenzung der Zeitschrittweite durch den Fehlerschätzer sind dies die Ursachen, die die Zeitschrittweite unterhalb der maximalen Rechenschrittweite begrenzen. Durch Anzeige der Ergebnisgrößen **`dt`** (Zeitschrittweite) (Modellexplorer oder Eigenschaftsfenster des Modells) können die kritischen Zeitbereiche ermittelt werden. Dabei ist **`dtProtMin`** genügend klein zu wählen, um alle interessanten Zeitschrittweiten anzeigen zu lassen.<br>➢ Beim (ME)BDF-Verfahren wird bei den meisten Modellen die meiste Zeit für die Residuen-berechnungen benötigt. Deshalb ist die Anzahl der Residuenberechnungen ein gutes Maß, an welchen |

| | |
|---|---|
| | Stellen langsam gerechnet wird. Die Anzahl der Residuenberechnungen ist die 7. Komponente des **`solverInfo`**-Vektors (Modellexplorer oder Eigenschaftsfenster des Modells). Interessant sind die Zeitbereiche mit großem Anstieg.<br>➢ Die mit diesen beiden Punkten ermittelten Zeitbereiche können in **`traceOn`** eingetragen werden.<br>➢ Bei vielen Unstetigkeiten können diese mit ***Unstetigkeiten/Unstetigkeitsstellen*** angezeigt werden. Die Ursachen für die Unstetigkeiten können durch zusätzliche Anzeige von ***Variable/Änderung von Werten diskreter Variablen*** und ***Unstetigkeiten/ Vorzeichenänderung von Zerofunctions*** ermittelt werden. Wenn lediglich die Zeitpunkte der Unstetigkeitsstellen interessant sind, können diese aus der 8. Komponente des **`solverInfo`**-Vektors abgelesen werden. Dabei ist **`dtProtMin`** höchstens auf **`dtMin`**, bei sehr kleinen Unstetigkeitsabständen auf **`dtDetect`** zu setzen.<br>➢ Bei vielen ungültigen Schritten können diese ebenfalls in einem Ergebnisfenster oder beim Wunsch nach näheren Informationen im Ausgabebereich angezeigt werden. Im Ergebnisfenster sind die 3. (Anzahl der verworfenen Schritte (ungültige Schritte oder im Zusammenhang mit einer Unstetigkeit verworfene)), 4. (Anzahl der ungültigen Schritte mit zu großem Fehlerschätzer) und 5. (Anzahl der (sonstigen) ungültigen Schritte wegen mangelnder Konvergenz) Komponente des **solverInfo**-Vektors zu betrachten. Da jeweils die Anzahl zwischen vorhergehendem und aktuellem gültigen Schritt ausgegeben wird, ist wieder **`dtProtMin`** genügend klein zu wählen. Auch kann die Ausgabe ungültiger Schritte (***Solver/ Ungültige Schritte/Details***) im Ausgabebereich angeschaltet werden. Dort ist auch jeweils die Ursache erkennbar. Das weitere Vorgehen entspricht dem bei der jeweiligen Fehlermeldung beschriebenen für den entsprechenden Zeitbereich.<br>➢ Die Beschränkung der Zeitschrittweite durch die Unstetigkeitssuche kann durch Aktivieren von ***Unstetigkeiten/Begrenzung der Zeit durch Zero-Function-Schätzung vor einem Schritt*** und ***Unstetigkeiten/ Rücksetzung der Zeit durch Zero-*** |

| | |
|---|---|
| | ***Function-Nulldurchgang nach gültigem Schritt*** angezeigt werden.<br>➢ Wenn nichts davon zutrifft und es trotzdem viele gültige Schritte mit kleiner Zeitschrittweite gibt, so liegt dies am Fehlerschätzer. Dann darf die 6. Komponente des **solverInfo**-Vektors nicht wesentlich größer als 1 werden (wieder **`dtProtMin`** genügend klein wählen). Welche Zustandsgrößen die kleinen Zeitschrittweiten erzwingen, kann mit Hilfe der Performanceanalyse (Menü ***Simulation/Performance-analyse…***) festgestellt werden. Dazu ist sie zu aktivieren und die Werte müssen über der Simulationszeit aufgezeichnet werden. Dann kann das entsprechende Zeitintervall ausgewählt und die Zustandsgrößen nach ihrem Einfluss sortiert werden. Interessant sind die Größen, deren Einfluss vom Anfang zum Ende des Zeitintervalls stark wächst. Diese Größen können dann in einem Ergebnisfenster angezeigt werden, falls sie Ergebnisgrößen sind (wieder **`dtProtMin`** genügend klein wählen). Wenn das nicht der Fall ist, können Funktionsglieder genutzt werden. Als Veränderliche wird die anzuzeigende Größe eingetragen. Wenn die starke Änderung der entsprechenden Zustandsgrößen korrekt ist, müssen normalerweise entsprechend kleine Zeitschritte gewählt werden, um diese Größe mit hinreichender Genauigkeit darzustellen. Sind die entsprechenden Zustandsgrößen für die Berechnung uninteressant, so können die größeren Änderungen übersprungen werden, ohne die Zeitschrittweite zu verkleinern. In diesem Fall ist im dazugehörigen Attributdialog (siehe **Kapitel 5.3.1**) **`discontChange`** auf true zu setzen. |
| ***Wie kann ich die Richtigkeit / Genauigkeit der Ergebnisse prüfen?*** | In die Berechnungsverfahren sind Tests eingebaut, die prüfen, ob die Ergebnisse korrekt sind. Diese beziehen sich auf die Zustandsgrößen. Welche das sind, wird im Ausgabebereich nach Aktivieren von ***Simulation/ Einstellungen Transient…/Tracing/Solver/Zustandgrößen*** angezeigt. Dazu ist nur die Ausführung eines Zeitschritts erforderlich. Oftmals sind unerklärliche Ergebnisse<br>– in Modellierungsfehlern,<br>– in der Eingabe falscher Parameter (z. B. auch falsche Einheiten), |

| | |
|---|---|
| | – im Einsatz von Elementen außerhalb ihres Gültigkeitsbereiches,<br>– in ungenügenden Systemkenntnissen,<br>– in falschen Erwartungen,<br>– in unzureichenden Kenntnissen über die Software<br>begründet. Wenn der Löser ein Modell ohne Ausgabe von Fehlermeldungen berechnet, entsprechen die Ergebnisse in der Regel auch dem Modell. Ausnahmen bestehen, falls die Fehlerkriterien (relativer und absoluter Fehler) nicht passend zur Modellqualität eingestellt sind oder die Zeitschrittweiten zu groß sind.<br>Frieren Sie dann die Ergebnisse ein und wiederholen Sie die Berechnung mit kleineren Fehlerschranken und bei Bedarf mit kleineren Schrittweiten. Wenn sich bei weiterer Verkleinerung die Ergebnisse nicht mehr wesentlich ändern, haben Sie die optimale Einstellung erreicht.<br>Diesen Vergleich können Sie auch mit einem anderen Solver (Umschalten zwischen BDF- und MEBDF-Verfahren) durchführen. |

## 11.3.2 Externe Löser (Löser mit konstanter Schrittweite und CVODE)

Das folgende Kapitel beschreibt den Inhalt von Warnungen und Fehlermeldungen, die bei Verwendung der externen Löser (Löser mit konstanter Schrittweite oder CVODE) auftreten und gibt Hinweise zu deren Ursachen. Dabei ist es für die Fehlersuche und Problembehebung hilfreich, die entsprechenden Abschnitte

- **6.1.3.2 Löser mit konstanter Schrittweite (externer Solver)** bzw.
- **6.1.3.3 CVODE (externer Solver)**

nachzulesen sowie die verwendeten Begriffe gemäß **Kapitel 12** zu kennen.
Die Aktivierung von Tracingausgaben wurde bereits in **Kapitel 6.1.1.4**. beschrieben. Im Folgenden ist deshalb nur noch die jeweilige Rubrik angegeben, z.B. ***Gleichungen/Symbolische Jacobimatrix***.

### 11.3.2.1 Warnungen, die im Zusammenhang mit nichtlösbaren Blöcken auftreten

Eine Besonderheit bei den externen Lösern ist, dass es Größen im Modell gibt, die gleichzeitig innerhalb eines so genannten Blocks berechnet werden. Dabei handelt es sich um ein System linearer oder nichtlinearer Gleichungen, welches mit einer Variante des (iterativen) Newtonverfahrens gelöst wird. In manchen Fällen schlägt die Berechnung dieses Systems fehl. Dann treten die folgenden Fehlermeldungen auf:

| Fehlermeldung / Verhalten | Ursache/Hilfe |
|---|---|
| ***"Error: System of equations with the DAE state(s) … cannot be solved! …"*** ***bzw.*** ***"Error: Equation with the DAE state(s) … cannot be solved! …")*** | Hierin finden Sie alle Größen, die in den nichtlösbaren Block eingehen, und den Iterationsschritt, in dem der Blockgleichungslöser die Berechnung abbricht. Wird in dem Block nur eine Größe berechnet, dann wird die untere Meldung ausgegeben. |

Nach dieser Fehlermeldung folgt eine weitere, die für die Suche nach den Blockgleichungen innerhalb des Tracings ***Gleichungen/Symbolische Jacobimatrix*** wichtig ist. Sie sagt aus, wann der Fehler auftritt (während der Initialisierung, während eines Zeitschritts, nach einer Unstetigkeit, etc.).
In den meisten Fällen wird eine der vier folgenden Fehlermeldungen vor der obigen Meldung ausgegeben. Sie gibt Ursachen an, warum der Gleichungsblock nicht erfolgreich gelöst werden konnte.

| Fehlermeldung / Verhalten | Ursache/Hilfe |
|---|---|
| ***"Residual is not in the image of … Jacobian ... with the DAE state(s) ..."*** | Aus dieser Fehlermeldung gehen die Größe des nichtlösbaren Blocks, die eingehenden Blockvariablen, die Nummer der nichtlösbaren linearisierten Gleichung (falls die vorhergehenden Gleichungen gelöst werden konnten) und bei nur einer Spalte den entsprechenden Eintrag in der Jacobimatrix und den Wert des Residuums hervor.<br>➢ Lassen Sie sich mittels ***Gleichungen/Symbolische Jacobimatrix*** die "Symbolische Jacobimatrix (initial equations)" bzw. die "Symbolische Jacobimatrix" anzeigen. Suchen Sie dort nach einer der beteiligten Variablen. Sie finden dann den entsprechenden Block mit allen beteiligten Veränderlichen. Versuchen Sie herauszufinden, warum er nicht lösbar ist. |

Ein Fehler, der zur Ausgabe dieser Warnung führt, kann zum Beispiel während der Initialisierung wegen inkonsistenter Anfangswerte auftreten:

*21. Step: Residual is not in the image of 2*1-Jacobian with the DAE state _der_connection3.phi: res[1] = 6.5741692360397401 (>4.4408920985006302e-013) != jac[1][0] * s = 1 * 0*

Bevor die Simulation abbricht, folgen die beiden Fehlermeldungen

*Error: System of equations with the DAE state(s) _der_connection3.phi cannot be solved! After 21 steps residual is not in the image of the Jacobian.*
*Calculation aborted because of a problem during the calculation of initial conditions.*

Betrachten wir in ***Gleichungen/Symbolische Jacobimatrix*** die Blockgleichungen und die Anfangswerte der eingehenden Variablen (mehrere Gleichungen wurden hier zusammengefasst),

*initial: connT.om0.fixed:=true;;*
*initial: connection3.om0.fixed:=true;*
*initial: der_connection3.phi=connection3.om.start=connection3.om0=0;*
*initial: der_connT.phi=connT.om.start=connT.om0=20;*

*initial:{{*
*EQUATION<>{_der_connection3.phi=0.65741692360397397*_der_connT.phi}*
*EQUATION<>{_der_connection3.phi=connection3.om.start}*
*}};*

dann erkennen wir schnell die Ursache des Problems: Die vorgegebenen Anfangswerte wurden fixiert und sind inkonsistent.

| **Fehlermeldung / Verhalten** | **Ursache/Hilfe** |
|---|---|
| ***"the value … is below the minimum (…) of the DAE state …",***<br>***"the value … exceeds the maximum (…) of the DAE state …",*** | Tritt eine dieser beiden Fehlermeldungen auf, dann hat eine Größe versucht, ihren zulässigen Bereich zu verlassen, d.h. sie hat ihr Minimum unter- oder ihr Maximum überschritten. Als weitere Informationen finden Sie zudem den Überschreitungswert, die im Modell gesetzten Grenzwerte und den Dämpfungsfaktor, der die Überschreitung verhindert.<br>➢ Lassen Sie sich mittels ***Gleichungen/Symbolische Jacobimatrix*** die "Symbolische Jacobimatrix (initial equations)" bzw. die "Symbolische Jacobimatrix" anzeigen. Suchen Sie dort die entsprechende Variable und die dazugehörigen Blockgleichungen. Schauen Sie sich diese und die aktuellen Werte der Blockvariablen in Ergebnisfenstern an und versuchen Sie herauszufinden, warum die Veränderliche versucht, ihren zulässigen Bereich zu verlassen. |

| | |
|---|---|
| ***"Reducing the damping factor to … because of the DAE state …", "DAE state … exceeds a border"*** | Treten obige Grenzwertüberschreitungen mehrfach hintereinander auf, dann verkleinert der Blockgleichungs-löser sukzessive den Dämpfungsfaktor. Dies ist solange möglich, bis dieser gleich Null ist. Dann wird die untere Fehlermeldung ausgegeben. |

Die Werte der Blockvariablen können aktuell nur in Ergebnisfenstern angezeigt werden - falls sie Ergebnisgrößen sind (wieder **`dtProtMin`** genügend klein wählen). Wenn letzteres nicht der Fall ist, können Funktionsglieder genutzt werden, in denen die anzuzeigende Größe als Veränderliche eingetragen wird.

SIMULATION X®

Powered by ITI

Definitions

# Definitions

# 12 Begriffsdefinitionen

| | |
|---|---|
| 3D-Ansicht | Für Modelle mit Elementen aus den Bibliotheken MKS-Mechanik, Ebene Mechanik, Antriebstechnik und Animationskörper ist eine 3D-Ansicht verfügbar, in der sich die räumliche Anordnung und die Geometrie darstellen und bearbeiten lassen sowie der berechnete Bewegungsverlauf visualisiert wird. |
| Algorithmus | Teil der Verhaltensbeschreibung eines Typs (Modelica-Klasse). Dient zur Berechnung von Variablen und Ergebnisgrößen während der Simulation. |
| Analyse | Werkzeuge innerhalb von SimulationX zur schnellen Bewertung eines Modells. Es bestehen folgende Analysemöglichkeiten:<br>*Modellanalyse*<br>– Transiente Analyse im Zeitbereich<br>– Stationäre Analyse im Frequenzbereich<br>– Lineare Systemanalyse<br>• Eigenfrequenzen und Schwingformen für lineare und nichtlineare Systeme<br>• Übertragungsverhalten<br>*Ergebnisanalyse*<br>– FFT – Frequenzanalyse<br>– FRF – komplexer Übertragungsfrequenzgang<br>– Ordnungsanalyse<br>– Transformationen von Verläufen (Addition, Mittelwert, usw.) |
| Anfangswert | Variablen benötigen Anfangswerte. Sie werden für den Anfang des Lösungsverlaufs von Differentialgleichungssystemen oder Integratoren (Integrationskonstanten) benötigt und festgelegt.<br>Die Anfangswerte ergeben sich aus den Startbedingungen (Startwerte) und den expliziten Zuweisungen (Gleichungen, Algorithmen) während der Anfangswertberechnung. Eingegebene Startwerte müssen in SimulationX fixiert (fest) sein um auch als Anfangswert zu gelten. Andernfalls werden sie als Startwert für die Anfangswertiteration interpretiert. Der Anfangswert weicht dann vom Startwert ab. |

| | |
|---|---|
| Anschluss | auch mit Pin, Port oder Connector bezeichneter Punkt an einem Element (Modell-Komponente), an dem eine Verbindung (Connection) zu anderen Elementen erfolgen kann. Es gibt verschiedene Typen von Anschlüssen (z. B. für translatorische und rotatorische Mechanik, Hydraulik, Elektrotechnik, Signalein- und -ausgang). Nur Anschlüsse gleichen Typs lassen sich miteinander verbinden. Jeder Anschluss besitzt – bezogen auf das jeweilige Element – einen eindeutigen Namen. |
| Assistent | Programm-Modul, das den Benutzer schrittweise durch eine Anwendung führt (z. B. Optimierungs-Assistent oder Varianten-Assistent ). |
| Ausgabebereich | Im Ausgabebereich werden Meldungen, Warnungen und Fehlermitteilungen protokolliert. Diese Nachrichten sind verschiedenen Kategorien zugeordnet (z. B. Simulation, Datei). Der Inhalt des Ausgabebereichs kann gespeichert, im Textformat exportiert und ausgedruckt werden. |
| Auswahlbox | Fenster zur Auswahl vorgegebener Varianten (z. B. Maßeinheiten im Eigenschaftsfenster oder Ergebnis-größen im Ergebnisfenster ) |
| Bibliothek | Gruppe von zusammengehörigen Element-Typen (z.B. sind die translatorischen Elemente „Feder“, „Dämpfer“, „Reibstelle“ in der Bibliothek *Translatorische Mechanik* enthalten). |
| Bibliotheksleiste | Über die Baumansicht werden die Bibliotheken (z. B. *Mechanik*, *Signalglieder* etc.) mit jeweils logisch zusammengehörigen (Modell-)Typen (z. B. Feder, Dämpfer, Masse) entsprechend ihrer Hierarchie angezeigt.<br>Die Bibliotheksleiste bietet Zugang zu den installierten Typen über Symbole. |
| Code-Export | In SimulationX integriertes Werkzeug zur Überführung der Funktionalität eines kompletten SimulationX-Modells in C-Code. Vor dem Export wird festgelegt, welche Eingänge, Ausgänge, Parameter sowie Kennlinien im exportierten Modell als Interfaces bereitstehen sollen. |

| | |
|---|---|
| Compound | Als Compound wird ein Modellelement bezeichnet, das durch Instanziierung eines Compound-Typs erzeugt wird. |
| Compound-Typ | Als Compound-Typ wird die Zusammenfassung einer (Teil-) Modellstruktur (einschließlich enthaltener Parameter und Verhaltensbeschreibung) bezeichnet. Compound-Typen können mit dem TypeDesigner aus einer bereits existierenden Modellstruktur erstellt werden. |
| Diskrete Variable | Eine diskrete Variable ist eine Ergebnisgröße, deren Wert sich nur zu diskreten Zeitpunkten (d. h. an so genannten Ereignissen) ändert. |
| Domäne | Unter einer (physikalischen) Domäne versteht man in SimulationX die Gesamtheit von Verbindungs-Typ, Anschluss-Typ, eventuell Fluid-Typen sowie den zugehörigen physikalischen Größen. Domänen orientieren sich vorwiegend an physikalischen Fachgebieten (z.B. translatorische Mechanik, rotatorische Mechanik, Hydraulik, Elektronik, usw.). Domänen sind in SimulationX intern vordefiniert. Die Verfügbarkeit der einzelnen Domänen (z.B. der Anschluss-Typen) kann lizenzabhängig sein. Der Anwender kann selbst keine neuen Domänen erstellen. |
| Eigenschaftsfenster | Eigenschaftsfenster dienen zur Anzeige und Bearbeitung von Eigenschaften der Komponenten (Elemente, Verbindungen) eines Simulationsmodells. Über die einzelnen Registerkarten können Werte, Maßeinheiten und Protokollattribute eingegeben bzw. gesetzt werden. |
| Element | Das Element ist Bestandteil eines Modells. Es entsteht durch Erzeugung aus einem Element-Typ. Dieser legt die Auswahl und Art von Parametern, Anschlüssen, Ergebnisgrößen und der Algorithmen bzw. Gleichungen fest. Das Element selbst trägt die Werte der Parameter und Variablen (z. B. wird das Element „mass1“ in der Strukturansicht vom Element-Typ „Mechanics.Translation.Mass“ erzeugt und hat dessen Eigenschaften). |
| Element-Typ | Beschreibung einer Klasse von Elementen, die alle in Anschlüssen, Parametern, Algorithmus, Gleichungen, inneren Komponenten usw. übereinstimmen (z. B. repräsentiert die „Masse“ in der Bibliothek |

| | |
|---|---|
| | „Translatorische Mechanik“ einen Element-Typ mit dem Identifier „Mechanics.Translation.Mass“) |
| Externe Typen | Vom Anwender, z.B. mittels *TypeDesigner*, erstellte Typen. |
| Ereignis | Zeitpunkt während des Simulationslaufes, an dem sich<br>- Zustandsgrößen des Modells (oder deren Ableitungen) sprungartig ändern (Unstetigkeitsstelle) , oder<br>- Änderungen in der Struktur der Gleichungen des Modells ergeben.<br>Beispiele für Unstetigkeiten sind Knicke und Sprünge in Kennlinien. Strukturumschaltungen treten bei Anschlägen, beim Übergang zwischen Haft- und Gleitreibung, Öffnen/Schließen von Ventilen, Spiel in mechanischen Elementen auf. Ereignisse werden vom Löser im Vorfeld erkannt und gesondert behandelt. |
| Ergebnisfenster | Fenster zur Darstellung der Simulationsergebnisse in grafischer Form, als x(t)- oder x(y)-Diagramme. Es gibt die Möglichkeit, mehrere Ergebnisverläufe in einem Fenster darzustellen sowie mehrere Ergebnisverläufe miteinander zu verrechnen oder einzelne Verläufe umzurechnen. |
| Ergebnisgröße | Variable eines Modells, eines Elements oder einer Verbindung, für die während der Simulation ein Zeitverlauf berechnet wird. |
| Fluid | Das Fluid ist ein Bestandteil einer hydraulischen, pneumatischen oder Thermofluid-Verbindung. Jedes Fluid hat einen eindeutigen Bezeichner (Namen), der dem Fluid einen Datensatz an Fluideigenschaften (bzw. die benötigten Parameter zu deren Berechnung) aus einer Datenbank zuordnet. Die Fluid-Datenbanken werden beim Start von SimulationX eingelesen. Fluide werden nach ihrer Auswahl in der Verbindung von den Elementen propagiert, d.h. alle Verbindungen in demselben Kreislauf erhalten dasselbe Fluid. |
| FluidDesigner | In SimulationX integriertes Werkzeug zur Entwicklung nutzerdefinierter Fluidtypen für die Bibliotheken Hydraulik und Pneumatik (Fluide, Gase und Mischungen). |

| | |
|---|---|
| Flussgröße | Zur physikalischen Modellierung verwenden die SimulationX-Bibliotheken einen Netzwerkansatz, d. h. es wird von Knoten und Zweigen ausgegangen. Flussgrößen fließen durch die Zweige (z. B. Strom in elektrischen Netzwerken). |
| Frequenzanalyse | • Im Ergebnisfenster (FFT): Die Frequenzanalyse gestattet die Ermittlung eines Amplitudengangs über der Frequenz für ein vorliegendes Zeitsignal.<br>• In der linearen Systemanalyse: Analytisch bestimmte Antwort eines Modells an einem bestimmten Ausgang bei Erregungen an einem bestimmten Eingang. Dargestellt als Amplitude/Phase oder Real- und Imaginärteil der komplexen Systemantwort über der Frequenz. |
| Gleichgewicht | Das Gleichgewicht bezeichnet den Zustand eines Modells, bei dem<br>- die zeitliche Änderung (d. h. die Ableitungen) aller Zustandsgrößen Null ist, oder<br>- die höchsten Ableitungen (z. B. die Beschleunigungen) Null sind.<br>In der Mechanik wird dies häufig als „statisches Gleichgewicht“ bezeichnet, in der Fluidtechnik hingegen als „stationäres Gleichgewicht“. Ein System (d. h. Modell) kann genau einen, mehrere oder auch keinen Gleichgewichtszustand haben. |
| Gleichung | Mit Gleichungen werden Eigenschaften, Abhängigkeiten und physikalische Gesetze formuliert, die in einem Modell oder Typ gelten. In den Gleichungen kommen neben Parametern im Allgemeinen mehrere Zustands- und Ergebnisgrößen vor, dadurch sind mehrere Gleichungen miteinander gekoppelt. Um die unbekannten Variablen (Zustandsgrößen) zu berechnen, muss das Gleichungssystem mit Hilfe eines Solvers gelöst werden. |
| Initialer Algorithmus | Wie Algorithmen, jedoch gelten diese nur während der Anfangswertberechnung. Sie gelten dann zusätzlich zu den Gleichungen und Algorithmen. Man kann mit ihnen Variablen oder konstante Parameter berechnen, die z.B. iterativ in einer Schleife zu bestimmen sind. |
| Initiale Gleichungen | Wie Gleichungen, jedoch gelten diese nur während der Anfangswertberechnung. Sie gelten dann zusätzlich zu den Gleichungen und Algorithmen. Damit lassen sich |

| | |
|---|---|
| | zum Beispiel zusätzliche Zwangsbedingungen beschreiben um den Anfangszustand eines Modells (oder Elementes) zu definieren. Man kann mit ihnen Variablen oder konstante Parameter berechnen. |
| Interne Typen | Alle mit der Software gelieferten Typen (Inhalt der Bibliotheken) werden als interne Typen bezeichnet. |
| ITI-Script | In SimulationX integrierte Scriptsprache zur Steuerung der Simulation über das vorhandene COM-Interface. |
| Komponente | Bestandteil eines Modells oder eines Element-Typs, Parameter, Variablen, Elemente und Verbindungen. |
| Komplexe Übertragungsfunktion (FRF) | Numerisch ermittelte Übertragungsfunktion zwischen einem Eingangs- und Ausgangssignal. Besteht aus Amplitudengang und Phasengang (jeweils über der Frequenz). |
| Konvergenzbereich | a) Bei der Anfangswert- oder Gleichgewichtsberechnung: Die Menge derjenigen Startwerte, für die die Iteration zu einer Lösung des Gleichungssystems konvergiert. bzw.<br>b) in einem Zeitschritt bei impliziten Verfahren: Die Menge derjenigen Prädiktorwerte, für die die Korrektoriteration zu einer Lösung des Gleichungssystems konvergiert. |
| Kreislauf | Teil einer Modellstruktur aus Elementen einer der *Bibliotheken* Hydraulik, Pneumatik oder Thermo-Fluidtechnik und der zugehörigen Verbindungen. Aufgrund der Struktur müssen alle Verbindungen des Kreislaufes dasselbe FluidFluid besitzen. Dies wird durch die Propagierung des Fluids im Kreislauf gesichert (siehe auch Propagierung). |
| Lineare Systemanalyse | Bei der linearen Systemanalyse findet eine Linearisierung des Modells um den aktuellen Arbeitspunkt statt. Die Ergebnisse gelten also bei nichtlinearen Modellen nur innerhalb einer (kleinen) Umgebung um diesen Arbeitspunkt.<br>Dazu zählen:<br>- Eigenfrequenzen und Schwingformen<br>- Übertragungsverhalten. |
| Lokale Typen | Diese Typen werden innerhalb eines anderen (übergeordneten) Typs (z.B. Modell oder Element-Typ) definiert. Sie lassen sich komfortabel mit einem Simulationsmodell weitergegeben. |

| | |
|---|---|
| Modelica® | Quasi-Standard einer Modellbeschreibungssprache, die von der Modelica Association entwickelt und gepflegt wird. Modelica ® ist ein eingetragenes Warenzeichen der Modelica Association. |
| Modell | Ein Modell ist das Abbild eines Systems oder Prozesses unter Verwendung von Elementen, Verbindungen, Gleichungen und Algorithmen.<br>Ein Modell ist immer ein Typ aus der Klasse *model* (Element-Typ) und kann instanziiert werden. |
| Modellexplorer | Strukturiertes Anzeige- und Eingabewerkzeug für Eigenschaften von Elementen und Verbindungen eines Simulationsmodells. Die Baumansicht stellt die Element- und Typenhierarchie des Modells dar. Die Parameter und Ergebnisgrößen der jeweils ausgewählten Komponente werden in Tabellenform angezeigt und lassen sich dort bearbeiten. |
| Modul | Ladbare Software-Komponente (exe, dll, cat) in der auch Element-Typen implementiert sein können. |
| Ordnungsanalyse | Methode zur Frequenzanalyse, bei der die Amplituden nicht nur über der Frequenz sondern auch über der Dreh- oder Taktzahl bzw. Ordnung dargestellt werden. Sie eignet sich besonders für die Analyse von rotierenden oder periodisch arbeitenden Systemen, wie z.B. Antriebssystemen.<br>Mit ihr können Eigenfrequenzen, Erregerfrequenzen sowie Resonanzen übersichtlich dargestellt und leicht erkannt werden. |
| Optimierung | Prozess, bei dem für ausgewählte Parameter des Modells Werte derart bestimmt werden, dass das Verhalten des Modells (Ergebnisverlauf oder Wert) einem vorgegebenen Soll-Verlauf oder –Wert best-möglich angepasst wird. Für die Optimierung steht eine separate Software zur Verfügung, die die Angabe des Optimierungsproblems unterstützt und den Optimierungsprozess automatisiert. |
| Parameter | Ein Parameter ist Bestandteil eines Modells, Elementes oder einer Verbindung und legt quantitative Eigen-schaften fest. Die Art der Parameter ist mit dem Element-Typ festgelegt. Der Wert (eine Konstante oder ein Ausdruck) der Parameter wird bei der Modellierung durch den Benutzer an der Oberfläche eingegeben oder über das COM-Interface festgelegt. |

| | |
|---|---|
| Potenzialgröße | Zur physikalischen Modellierung verwenden die SimulationX-Bibliotheken einen Netzwerkansatz, d. h. es wird von Knoten und Zweigen ausgegangen. Potenzialgrößen sind Variablen physikalischer Größen (z. B. Temperatur oder Druck), die den Systemzustand charakterisieren. Sie liegen an den Knoten an und können dort gemessen werden (z. B. Spannung in elektrischen Netzwerken). Die Differenz von Potenzialgrößen wird auch als Across Quantities oder Potenzialdifferenz bezeichnet. |
| Propagierung | Mechanismus, der sicherstellt, dass alle Verbindungen eines Kreislauf dasselbe Fluid haben. Zusätzlich zum Fluid werden noch weitere Einstellungen der Verbindung propagiert, d. h. an alle anderen Verbindungen des Kreislaufes weitergeleitet. Dazu gehören z. B. in der Hydraulik die Optionen zur Temperaturberechnung und zur Berechnung der Luftanteile. |
| Protokollattribut | Das Protokollattribut einer Ergebnisgröße wird gesetzt, wenn der zeitliche Verlauf der Ergebnisgrößen während der Simulation gespeichert werden soll und ist die Voraussetzung für eine Auswertung nach Ende der Simulation. |
| Residuum | Ergebnis der rechten Seite einer Gleichung, die so umgestellt wurde, dass auf der linken Seite eine Null steht. Für jeden Vektor von Zustandsgrößen ergibt sich ein Vektor von Residuen. Der Solver versucht, die Residuen im Rahmen eines Iterationsverfahrens zu minimieren. |
| Simulationslauf (kurz: Simulation) | In der Simulation wird das Verhalten des Modells über der Zeit ermittelt. Dieses hängt von der Modellstruktur sowie den verwendeten Parametern und Anfangswerten ab. Als Resultat erhält der Anwender den zeitlichen Verlauf aller Ergebnisgrößen und kann diesen auswerten. |
| Solver | Paket numerischer Lösungsverfahren, mit dem während des Simulationslaufes für jeden Zeitpunkt die Zustandsgrößen eines Modells berechnet werden. |
| Startwert | In der Regel werden nicht alle Anfangswerte vom Anwender eingegeben. Der Löser versucht anhand von Startwerten und expliziten Zuweisungen vor der Lösung des Differenzialgleichungssystems iterativ konsistente Anfangswerte zu ermitteln. |

| | |
|---|---|
| | Startwerte von Variablen lassen sich über den Attributedialog dieser Variablen eingeben und fixieren oder indirekt über verknüpfte Startwert-Parameter, die i.d.R. mit „Anfangs…“ kommentiert sind, festlegen (Anfangsweg, Anfangsgeschwindigkeit usw.). Dort kann auch die Fixierung gesetzt werden.<br>Startwerte sind nicht gleich Anfangswerte. Im Unterschied zu den Startwerten können die Anfangswerte während der Anfangswertberechnung verändert werden. Nur wenn der Startwert fixiert ist, wird er als Anfangswert übernommen. |
| Statechart Designer | Tool zur komfortablen Modellierung und Simulation von zeit- und zustandsdiskreten Kontrollalgorithmen in SimulationX. Basierend auf einer großen Teilmenge des UML Statechart Standards unterstützt er die zuverlässige Modellierung von physikalischen Effekten oder technischen Subsystemen mit diskreten Zuständen (z.B. Reibstellen, Hysterese, Ventile und Schalter). |
| Stationäre Analyse | Berechnung periodisch eingeschwungener Zustände nichtlinearer und linearer Systeme in Abhängigkeit einer wachsenden oder fallenden Bezugsgröße (im Gegensatz zur transienten Simulation ohne Berücksichtigung deren Änderungsgeschwindigkeit bzw. für unendlich langsame Änderung). So lassen sich beispielsweise die eingeschwungenen Zustände für verschiedene Drehzahlen (d.h. einen Drehzahlbereich) berechnen und darstellen. |
| Strukturansicht | In der Strukturansicht wird die Struktur eines Modells unter Verwendung von Element-Symbolen und Verbindungslinien erstellt, bearbeitet und dargestellt.<br>Die Strukturansicht lässt sich animieren, d.h. Symbole, Positionen und Texte verändern sich dann automatisch anhand der Werte von Variablen. |
| Tracing | Informationen, die während der Simulation angezeigt werden können, um eine Fehlersuche zu erleichtern. Mit Hilfe der Baumansicht der verschiedenen Trace-Optionen kann die Protokollierung einzelner Ereignis-Arten im Ausgabebereich ein- bzw. ausgeschaltet werden |
| Transiente Analyse | Lösung des Differenzialgleichungssystems per Integration im Zeitbereich. Für lineare und nichtlineare Modelle. Es werden die instationären Lösungen unter Berück- |

| | |
|---|---|
| | sichtigung des Zeitverhaltens (z.B. Geschwindigkeit einer Resonanzdurchfahrt oder der Änderung einer Größe) berechnet (im Gegensatz zur stationären Analyse nicht eingeschwungen). |
| Typ | Beschreibung einer Klasse von Elementen oder Verbindungen, die alle in Anschlüssen, Parametern, Algorithmus, Gleichungen, inneren Komponenten usw. übereinstimmen. |
| TypeDesigner | In SimulationX integriertes Werkzeug zur Entwicklung neuer Typen (Modelltypen, Packages usw.) Der Anwender (Typ-Entwickler) spezifiziert Symbol, Anschlüsse, Komponenten, Activity Groups, Verhalten (3D-Ansicht und Gleichungen) und Dokumentation des neuen Element-Typs. |
| Variantenassistent | In SimulationX integriertes Werkzeug zur effektiven Durchführung von Parameterstudien. |
| Verbindung | Bindeglied zwischen Elementen, charakterisiert durch Typ (z. B. translatorisch-mechanisch) und Daten (Parameter, Ergebnisgröße). Eine Verbindung kann auch mehr als 2 Elemente verbinden. Dazu können Verbindungen verzweigt werden, sie besitzt dann einen oder mehrere Knoten. |
| Zerofunctions | Zerofunctions werden zur Detektierung zustandsabhängiger Ereignisse verwendet. Sie haben zum Zeitpunkt der Unstetigkeit einen Nulldurchgang. Zerofunctions werden von der symbolischen Analyse automatisch erzeugt. Beispielweise wird für den Ausdruck "if x>10 then 0 else 5" die Zerofunction "x-10" generiert. |
| Zustandsgröße | Variable, die eine wesentliche Eigenschaft des Modells beschreibt. Die Berechnungsvorschrift für Zustandsgrößen kann in der Regel nicht explizit vom Modellierer angegeben werden.<br>Die Werte der Zustandsgrößen werden durch einen Gleichungslöser (Solver) (im Allgemeinen iterativ) ermittelt. |

# 13 Literaturverzeichnis


**[1]** **K.E. Brenan, S.L. Campbell, L.R. Petzold:**
*Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations.*
Society for Industrial and Applied Mathematics
Philadelphia 1996

**[2]** **E. Hairer, S.P. Norsett, G. Wanner:**
*Solving Ordinary Differential Equations II - 2nd rev. Edition.*
Springer Verlag
Berlin Heidelberg 1993

**[3]** **C. Geiger, C. Kanzow:**
*Numerische Verfahren zur Lösung unrestringierter Optimierungsaufgaben*
Springer Verlag
Berlin Heidelberg 1999

**[4]** **Modelica Association**
*Modelica® – A Unified Object-Oriented Language for Physical System Modeling*, Language Specification, Version 3.2
Modelica Association 2010, Link, http://www.Modelica.org

**[5]** **A. C. Hindmarsh, P. N. Brown, K. E. Grant, S. L. Lee, R. Serban, D. E. Shumaker, and C. S. Woodward,** "SUNDIALS: Suite of Nonlinear and Differential/Algebraic Equation Solvers," ACM Transactions on Mathematical Software, 31(3), pp. 363-396, 2005.
(www.llnl.gov/casc/sundials/

**[6]** **Object Management Group**
OMG Unified Modeling Language (UML) Specification, version 2.1.2
http://www.omg.org/docs/formal/07-11-02.pdf
November 2007

**[7]** **Mechanical Simulation Corporation**
The VehicleSim API, Running and Extending VSSolver Programs.
August 2010

# 14 Index


### 3

3D-Ansicht ..... 5-53
- Auswahl von Elementen ..... 5-60
- Drahtgitter ..... 5-56
- Elemente erstellen ..... 5-61
- Elemente trennen ..... 5-63
- Elemente verbinden ..... 5-62
- globale Einstellungen ..... 5-55
- Kamera ..... 5-54
- Koordinatensystem ..... 5-53
- lokalen Einstellungen ..... 5-58
- Objektlage ändern ..... 5-61
- Realistisch ..... 5-56
- Standardansicht ..... 5-56
- Technische Ansicht ..... 5-56

### A

Activity Groups ..... 8-19, 8-21
Algebro-Differenzialgleichung ..... 6-13
Algorithmus ..... 3-6, 8-23, 12-1
allgemeine Kennlinie ..... 9-4
Analyse ..... 12-1
- Schwingungsanalyse ..... 6-43

Anfangswert ..... 5-23, 5-29, 6-12
Animation ..... 6-75
- aufzeichnen ..... 6-41
- Berechnungsergebnisse ..... 6-38
- externe Wiedergabe ..... 6-42
- wiedergeben ..... 6-41

Anschluss ..... 5-9, 5-11, 8-10, 12-2
- Innere Anschlüsse ..... 8-29

Array ..... 10-16

### B

Benutzeroberfläche
- 3D Ansicht ..... 4-10
- Ausgabebereich ..... 4-15
- Bibliotheksleiste ..... 4-3
- Dokumentationsansicht ..... 4-10
- Ergebnisfenstermanager ..... 4-15
- Modellansicht ..... 4-10
- Symbolleiste ..... 3-2
- Taskleiste ..... 4-15
- Textansicht ..... 4-10

Bibliothek ..... 7-56, 12-2
- Bibliotheksleiste ..... 4-3, 12-2

### C

Code-Export ..... 9-49, 12-2
- Ausführbares Modell ..... 9-58
- dSpace DS1006 ..... 9-58, 9-75
- Externes Modell ..... 9-58, 9-74
- MATLAB/Simulink S-Function .. 9-58
- Modelisar FMI for Co-Simulation 9-84
- Modelisar FMI for Model Exchange 9-81
- NI LabView ..... 9-78
- Post-Propcessing ..... 9-58
- Projekt ..... 9-52
- ProSys-RT ..... 9-58
- Quellcode ..... 9-49
- Scale-RT ..... 9-73
- Simpack ..... 9-70
- VehicleSim ..... 9-87

Code-Import
- FMU Code Import ..... 9-90
- Simulink/RTW ..... 9-92

COM-Schnittstelle ..... 7-55, 9-8
Co-Simulation ..... 9-23
- Steuerpakete ..... 9-24

### D

Daten
- externe - ..... 9-2, 9-5

Datenbank ..... 9-102
- Binden von Parametern ..... 9-106
- Datenbankanbindung ..... 9-97
- Datenbankkomponente ..... 9-98
- Datensatz ..... 9-108
- Filter ..... 9-108
- Tabelle ..... 9-104
- Verbindung ..... 9-103

Datenformat ..... 9-1, 9-7
- DIAdem-Header ..... 9-7
- IEEE-binär ..... 9-4
- ITI-binär ..... 9-7
- Text ..... 9-1

Demultiplexer ..... 9-26
Dokumantation ..... 8-25

Dokumentenfenster ..... 4-2
Drag&Drop 3-9, 4-9, 5-10, 5-15, 6-84, 7-14, 8-11, 8-31

## E

Edition
ANALYZER ..... 2-5, 2-17
PROFESSIONAL ..... 2-5, 2-17
Programm-Funktionen ..... 2-5
STUDENT ..... 2-4, 2-17
Test Version ..... 2-4
VIEWER ..... 2-5, 2-17
Eigenfrequenz..... 1-3, 3-16, 6-67
Eigenschaftsfenster...3-2, 3-5, 5-17, 12-3
Eigenvektor ..... 6-70
Einbettung ..... 6-16
automatisch ..... 6-20
natürlich ..... 6-17
Einstellung.....9-4, 9-7
Element 4-12, 5-9, 5-10, 5-15, 7-60, 12-3
Ableiten..... 3-16
Auswahl ..... 5-13
Kopieren..... 5-15
Mehrere Elemente auswählen ..... 5-14
mehrfach Platzieren..... 5-11
Platzieren..... 5-10
Verschieben..... 5-14
Element-Typ..... 8-1
global..... 8-1
lokal..... 8-1
Ereignis.....6-25, 12-4
Ereignisse ..... 6-26
Ergbnisfenster..... 7-4
Ergebnisfenster 3-4, 3-9, 4-13, 7-30, 9-7, 12-4
Balkendiagramm ..... 7-10
Darstellung ..... 7-9
Eigenschaften ..... 7-32
Eigenschaftsdialog ..... 7-6
Ergebnisleiste ..... 7-7
Fensterlayout..... 7-9
Hilfslinien..... 7-19
Kurvendarstellung..... 7-39
Öffnen ..... 7-5
Panels .....7-8, 7-35
Polarkoordinaten ..... 7-10
Sankey-Diagramm ..... 7-11
Seite.....7-8
Tortendiagramm .....7-11
Transformation .....7-22
Verknüpfungen.....7-22
y(t)-Darstellung .....7-12
y(x)-Darstellung .....3-10, 7-12
Zoom .....7-17
Ergebnisfenstermanager .....7-8, 7-45
Ergebnisgröße 3-3, 3-4, 3-6, 3-9, 4-12, 5-64, 7-1, 7-14, 12-4
Attribute .....5-19
Hinzuzufügen .....4-14
Kopieren .....7-14
Löschen .....4-14
Protokollattribut .....4-13
Steuerelement .....5-24
Verschieben .....7-14
Ergebnisverlauf.....9-2, 9-4
Ausmessen.....7-27
Einfrieren.....7-30

## F

FAQ
Externe Löser .....11-27
Interne Löser .....11-4
Lizenzierung.....11-2
FFT .....7-47
Fluid.....12-4
FluidDesigner .....8-47, 12-4
Hydraulik.....8-47
Pneumatik.....8-53
Flussgrößen.....10-4
Formen.....5-64
Frequenz
-bereich.....7-48
Frequenzanalyse ..... 6-77, 7-47, 12-5
Frequenzgang.....7-48
FRF .....7-48

## G

Genauigkeit.....9-5
Gleichgewicht.....12-5
Gleichgewichtszustand ..... 1-3, 6-1, 6-65
Gleichungen.....8-23
Globale Parameter .....5-24
Globale Symbolische Analyse.....6-11
Globale Typen .....8-27

## H

HelpDesk....11-1

## I

Innere Komponenten....8-31
Installation
  Ändern der Lizenz....2-25
  Einzelplatzlizenz....2-21
  Grundlagen....2-1
  Lizensierung....2-12, 2-20
  Lizenzierung....2-25
  Mehrere Lizenzen....2-22
  Netzwerklizenz....2-21
  Stapelbetrieb....2-16
ITI
  Script....9-17
  Scripteditor....9-17

## K

Kennfelder....5-34
  2D-Kennfeld....5-46
  3D-Kennfeld....5-51
  Datenimport....5-47
  Eigenschaften....5-50
  Kennfelddialog....5-46, 5-51
  Toolbar....5-47, 5-52
Kennlinie....5-26
Kennlinien....5-34
  Approximation durch Kreisbögen 5-38
  Datenimport....5-34
  Eigenschaften....5-44
  Funktionsübersicht....5-43
  Hyberbolische Approximation....5-37
  Kennlinie....5-34
  Kennliniensatz....5-45
  Kurvenfortsetzung....5-40
  Kurvenschar....5-45
  Lineare Interpolation....5-36
  Spline Interpolation....5-36
  Treppenfunktion....5-35
Klasse....9-9
  Application....9-9
  Connections....9-11
  Curves....9-12
  Documents....9-9
  Parameters....9-11
  Results....9-12
  ResultWindows....7-8, 9-14, 9-15
  Selection....9-14
  SimObjects....9-10
Kommunikations-Protokoll....9-23
Kommunikationsschrittweite....9-28
Komponenten....3-6, 4-12, 5-9, 8-12, 8-31, 12-6
  Innere Komponenten....8-29
  Mehrfachauswahl....5-18
Kontextmenü....7-5
Koppelelement....9-26

## L

Laden....9-2, 9-5
Leistungsbilanz....6-104
lexikalische Einheiten....10-5
Lineare Systemanalyse....6-1, 6-67, 12-6

## M

Markieren....9-2
Menüleisten....4-2
Methode....9-9
  Add....9-10
  Open....9-9
Modelica....1-3, 8-26, 10-1, 12-7
  Bibliothek....4-5
  Klasse....4-5
  Packages....4-7
  Quellcode....8-24
  Standard Bibliothek....4-5, 5-7, 10-1
  Suchpfad....10-2
  Typen....4-5
Modell....3-6, 12-7
  Element....5-9
  Element Auswählen....3-4
  Elemente verbinden....3-6
  Komponente....5-9, 5-18
  Modellansicht....4-10, 4-12, 12-9
  Modellexplorer....3-6, 4-12, 12-7
  Modellstruktur....5-10, 8-30
  Parametereingabe....3-6
  Protokollattribut....3-3
  Speichern....7-1
  Verbindung....5-9
Modellierung....3-1
  Gleichungs basierend....5-3
  physikalisch-objektorientiert....5-1
  signalorientiert....5-2

Modellstruktur ..... 5-26
Momentananzeige ..... 5-64, 7-1
Öffnen ..... 7-2
Verknüpfung mit Ergebnisgröße.... 7-2
Momentenverlauf ..... 3-11

**N**

Name ..... 9-7

**O**

Optimierung..... 12-7
Isight..... 9-95
modeFRONTIER ..... 9-97
OptiY..... 9-94
Ordnungsanalyse ..... 6-89

**P**

Parameter..... 3-2, 3-6, 4-12, 12-7
Attribute ..... 5-19
Hinzuzufügen ..... 4-14
logische Ausdrücke ..... 5-21
Löschen ..... 4-14
mathematische Ausdrücke ..... 5-21
nichtlinear..... 5-25
physikalische Einheiten ..... 5-23
Steuerelement..... 5-20
variable..... 5-22
veränderlicher - .....5-28, 5-29
Performanceanalyse..... 6-37
Potentialgrößen..... 10-4
Programmschutz.....2-17
Dongle..... 2-17
Lizenzdatei ..... 2-19, 2-26, 2-30
USB Key ..... 2-19
Protokollattribut..... 3-7, 3-9, 4-13, 12-8

**R**

RealTime-Workshop ..... 9-70
Rechenschrittweite ..... 6-4
Rücksetzpunkte ..... 6-5
Intialisierungszustand ..... 6-6

**S**

Schwingform .....3-16, 6-67
Signalanalyse..... 7-48
Simulation ..... 5-29, 5-31, 12-8
Intialisierungszustand ..... 6-6
Rücksetzen ..... 3-2
Rücksetzpunkte ..... 6-5
Simulationseinstellungen ..... 6-4
Simulationsrechnung... 3-8, 5-64, 9-59
Simulationssteuerung .... 6-1, 6-3, 6-43
starten ..... 3-3
stationär ..... 1-3, 3-11, 6-42
Tracing ..... 6-7
transient .....1-3, 6-1
Zeitschritte.....6-25, 6-28
Simulationszeit .....9-28
SimulationX
Ändern von Parametern.....3-2
Arbeiten mit Beispielmodellen ..... 3-1
Bibliothek ..... 4-4
Editionen ..... 2-3
Einstellungen..... 7-55
Modell erstellen..... 3-4
Öffnen von Modellen ..... 3-2
Online-Hilfe ..... 3-3
PrintEngine..... 7-49
Programmoberfläche ..... 4-1
Release .....2-11
Simulatorkopplung
ADAMS .....9-28
VehicleSim .....9-38
Solver.....6-7, 6-28, 6-30, 12-8
BDF .....6-29
CVODE .....6-34
Einstellungen.....6-9
MBDF.....6-29
Speichern .....9-2, 9-4, 9-6, 9-7
Statechart .....8-55
Composite States .....8-63
Diagramm.....8-67
Pseudostates .....8-58
Self-transition .....8-62
States .....8-56
Transistions .....8-57
Trigger.....8-60
Statechart Designer.....8-67
Modelica-Code .....8-73
Modellierung .....8-68
Steuerelemente.....5-20, 5-64
für Ergebnisgrößen.....5-24
für Parameter .....5-21
für Startwerte.....5-23
Summationspunkt .....5-27

Symbolleisten....4-2

**T**

Textblöcke....5-63
Toleranz....6-4
Toolbar....7-32, 8-33
Toolfenster....4-1
Tools
  FEM-Import....9-97
  ModeFRONTIER....9-96
  OptiY....9-94
Tracing....6-3, 6-7, 12-9
Typ....12-10
  Compound....3-16, 8-29, 8-33, 12-3
  Element-Typ....8-21, 12-3
  external....5-8
  Vererbung....8-28
  Verhalten....8-22
TypeDesigner4-14, 8-7, 8-27, 8-29, 9-99, 10-1, 12-10
  Activity Grpoups....8-19
  Anschlüsse....8-10
  Basistyp....8-29
  Dokumantation....8-25
  Enumeration....8-17
  Funktionen....8-17
  Komponenten....8-12
  Lokale Typen....8-16
  Verhalten....8-22

**U**

Übertragungsfrequenzgang....7-48
Übertragungsverhalten....6-67, 6-76
Unstetigkeiten....6-25

**V**

Variable....5-25
Variablen
  Attribute....5-19
Variantenassistent....3-16, 6-81, 12-10
Variantenrechnung....6-88
Verbindung....3-6, 4-12, 5-9, 12-10
  Auswahl....5-14
  Erstellen....5-11

**Z**

Zeitbereich....7-48
Zerofunctions....12-10
Zugriffsrechte....2-1, 2-17
Zwangsbedingung....6-13