Einführung in SIMULINK 3.0

Paul Weber

22.07.99

Was ist Simulink?

- interaktives, grafikorientiertes Programm
  - zur Generierung von dynamischen Modellen
  - zur Simulation von dynamischen Modellen
- in Matlab integriert
  - Simulink-Variable sind in Matlab verfügbar
  - Simulink-Modell ist unter Matlab ausführbar
- Blockbibliothek für viele Funktionen
- Einbinden von C- und Fortran-Programmen
- Definition neuer Blöcke

Welche Systeme können modelliert werden?

- zeitkontinuierliche Systeme
- zeitdiskrete Systeme
- Mischungen
- Systeme mit diskreten Ereignissen

Blockbibliothek

Die Simulink-Blockbibliothek ist in funktionale Gruppen eingeteilt

- Sources — Signalquellen, Eingabe von Dateien und aus dem Workspace, Uhren, konstante Werte
- Sinks — Ausgabe von Variablen in Dateien oder Workspace, Grafikausgabe
- Continuous — Integration und Ableitung, Transferfunktionen, linearer Zustandsraum, Zero-Pole
- Discrete — Blöcke für zeitdiskrete Simulation

Blockbibliothek

• Math	Summe, Produkt, Multiplikation, math. Funktionen log. Operationen, Datenkonvertierung
• Functions & Tables	S-Function, Matlab-Funktion, Definition einer bel. Funktion, Look-Up Tables
• Nonlinear	Nichtlineare Funktionen und Switches
• Signals & Systems	Multiplexfunktionen, Signalein- und ausgänge, Datenein-/ausgabe, Datenübertragung
• Blocksets & Toolboxes	verschiedene spezielle Blöcke, die auch auf Matlab-Toolboxen zugreifen, Demos

Blöcke

- Blockbibliothek wird durch Mausklick geöffnet
- Block wird mittels gedrückter linker Maustaste in das Arbeitsfenster geschoben
- zweifacher Mausklick auf einen Block öffnet das dazugehörige Fenster, in dem Blockparameter gesetzt werden können
- mehrere Blöcke können zu einem neuen Block zusammengefaßt werden, der individuell beschriftet und mit einer Grafik versehen werden kann
- Blocksymbole enthalten Ein- und Ausgänge, die miteinander verbunden werden müssen. Verbindungen werden erzeugt, indem man den Cursor mit gedrückter linker Maustaste vom Ausgang eines Blocks zum Eingang eines anderen Blocks zieht.
- mit Ctrl- und linker Maustaste werden Blöcke kopiert und Verzweigungen von Blockverbindungen erzeugt.

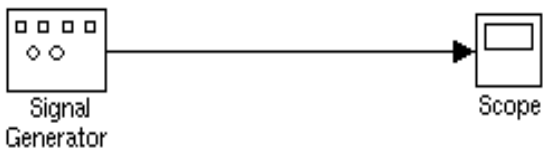
Editieren von Blöcken

Blöcke können zur besseren Dokumentation und zur besseren Übersicht

- maskiert werden; d.h. das Block-Icon kann verändert werden
- gruppiert werden; d.h. mehrere Blöcke können zu einem neuen Block zusammengefaßt werden
- die Blockunterschrift kann geändert werden, in dem man sie anklickt und überschreibt

Maskieren und Gruppieren von Blöcken geschieht im `EDIT`-Menü.

1. Beispiel

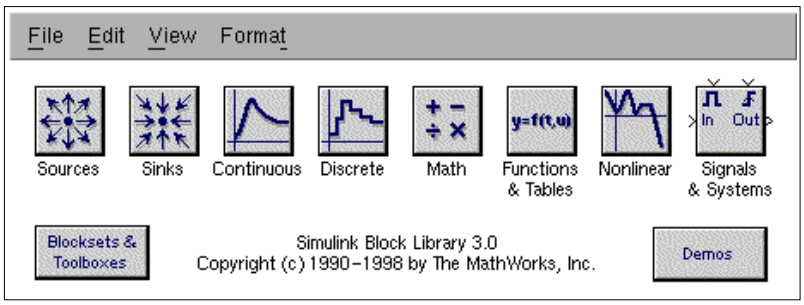


Frequenz:	6 (rad/sec)
Periodendauer:	$T = 2 \pi / \omega = 1.047$
Amplitude:	1

1. Beispiel

Aufruf von Simulink

```
matlab
>>simulink
```

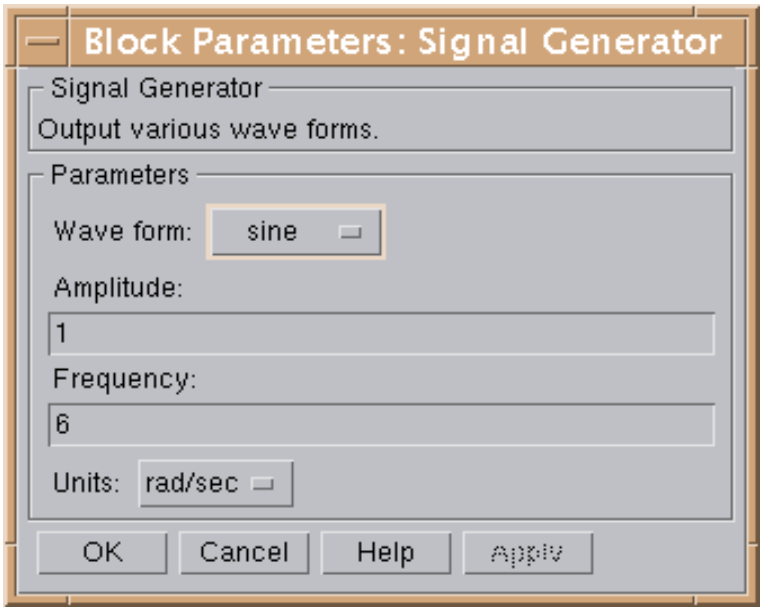


`File`	übliche Dateiverwaltungstools, wie Sichern, Öffnen, usw.
`Edit`	Editierfunktionen für Blöcke, erzeugen von Subsystemen, Maskieren von Subsystemen, usw.
`View`	Zoom-Funktionen
`Format`	Bearbeiten von Blocksymbolen

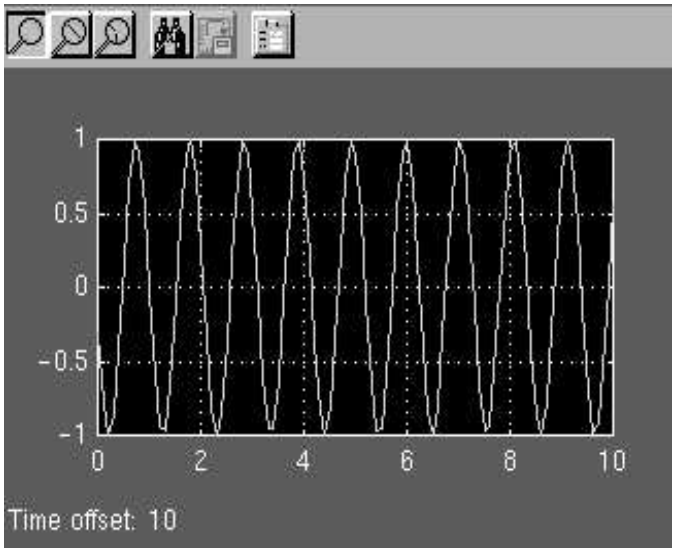
1.Beispiel

- beim SIMULINK-Aufruf wird ein Fenster eröffnet
- nacheinander `Sources` und `Sinks` anklicken und die Blöcke in das Fenster ziehen, Parameter setzen
- Blöcke miteinander verbinden
- im Simulation-Menü `Parameters...` anklicken und Simulationsparameter setzen:
  - Start time: 0.0
  - Stop time: 20.0
  - Solver options: Type: Fixed-step ode1 (Euler)
    Fixed Step size: 0.01
- im Simulations-Menü `Start` anklicken, Simulation läuft
- Anklicken des Scope-Blocks öffnet das Fenster mit dem Signalverlauf

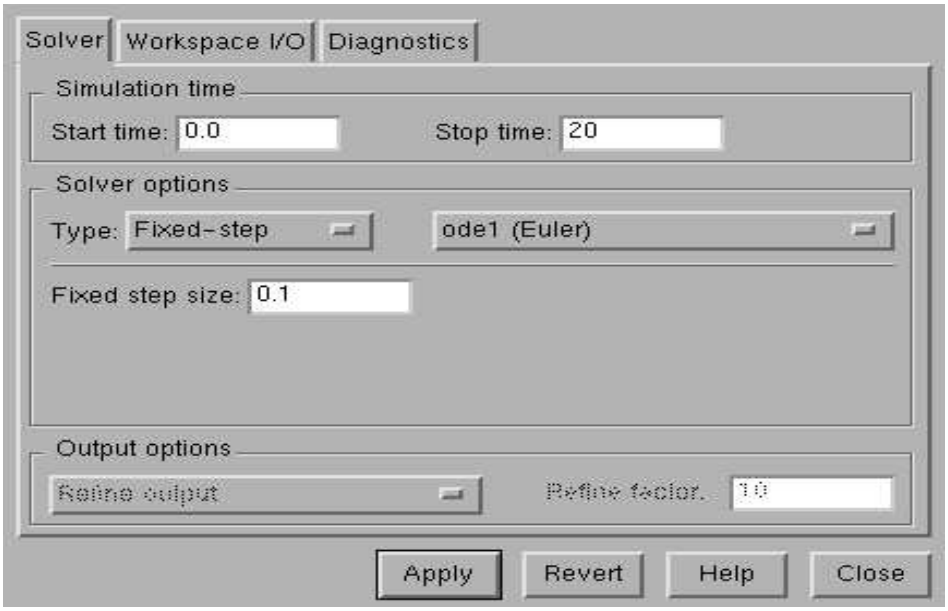
1.Beispiel



1. Beispiel



1. Beispiel



Sources

Signal Generator	Erzeugen von Sinus, Sägezahn, Stufen und Rauschen Eingabe: Frequenz und Amplitude
Sine Wave	Erzeugung von Sinuswellen Eingabe: Frequenz, Amplitude und Phase
Constant	Eingabe einer Konstanten
Pulse Generator	Rechteckpulse Eingabe: Periode, Pulslänge und Pulshöhe
Clock/Digital Clock	Uhrensignale

Sources

Random Number Uniform Random No.	Generator für normal- und gleichverteilte Zufallszahlen
From Workspace	Einlesen von Werten aus dem Arbeitsspeicher Eingabe: Matrix aus Zeit und Variablen
From File	Einlesen von Werten aus einer Datei, es wird reihenweise die Matrix aus Zeit und Variablen-werten eingelesen

Sinks

Scope	zeichnet einlaufende Signale auf (Oszilloskop) Eingabe: Zeitachse und Amplitude
Display	einlaufende Werte werden angezeigt
XY Graph	zeichnet Bahnkurven auf Eingabe: zwei zeitabhängige Variable
To Workspace	Ausgabe in den Arbeitsspeicher
To File	Ausgabe in eine Datei

Continuous

Derivative	bildet die Zeitableitung der Eingangsgröße
Integrator	integriert die Eingangsgröße Eingabe: Anfangswert
Transfer Fcn	Übertragungsfunktion Eingabe: Zähler- und Nennerkoeffizienten
Zero-Pole	faktorisierte Darstellung der Übertragungs-funktion Eingabe: Vektoren mit den Nullstellen und Polen als Komponenten, Koeffizient
State-Space	Simulation eine linearen Systems in der Zustandsraum-Darstellung Eingabe: Modellmatrizen

Discrete

Blöcke für zeitdiskrete Simulation. Hier findet man zu den kontinuierlichen Blöcken die diskreten Entsprechungen, also

- diskreter Integrator
- diskrete Übertragungsfunktion und Zero-Pole-Funktion
- diskreter Zustandsraum-Block
- verschiedene andere

Math

Gain Matrix Gain Slider Gain	multipliziert die Eingangsgröße (bzw. Matrix) mit einem konstanten Wert; bei *Slider Gain* kann der Wert mit einem Schieberegler modifiziert werden
Sum	Summiert die Eingangsgrößen Eingabe: Anzahl der Summanden
Product	Multipliziert zwei Eingangsgrößen
Inner Product	bildet das Skalarprodukt zweier Eingangsvektoren

Dazu kommen verschieden andere

- mathematische Funktionen
- logische Funktionen
- Konvertierungsfunktionen

Nonlinear

Verschiedene nichtlineare Funktionen wie
- Sättigungskurven
- Quantisierer
- Schalter
- Blöcke mit nichtlinearen Kennlinien

Signals & Systems

Inport/ Outport	formale Ein- und Ausgänge des Systems
Mux	Überlagerung von mehreren Signalen zu einem Vektorsignal Eingabe: Anzahl der Signale
Demux	Zerlegung eines Vektorsignals
Data Store ...	Definition, I/O auf einem Memorybereich
Ground, Terminate	Behandlung nicht verbundener Ports
From, Goto	Signalübertragung zwischen Blöcken ohne Verbindung

Functions & Tables

Fcn	hier kann eine beliebige Funktion der Eingangsgröße angegeben werden
S-Function	Aufruf einer S-Funktion
Matlab-Function	Aufruf einer Matlab-Funktion
Look-Up Tables	Interpolieren diskrete Werte

Block Sets & Toolboxes

Hier werden spezielle Blöcke zur Verfügung gestellt, die z.T. von Matlab-Toolboxen Gebrauch machen.

Des weiteren findet man hier

- Demos der einzelnen Blöcke
- Demo-Beispiele

2. Beispiel

Erstellen Sie das Blockdiagramm für folgende Differentialgleichung 1. Ordnung:

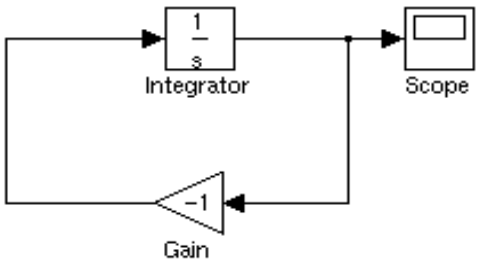
$$\frac{dx}{dt} = \lambda \cdot x$$

mit

$$x_0 = 1$$

$$\lambda = -1$$

2. Beispiel

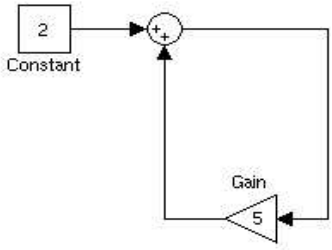


Schleifen kommen praktisch immer vor und stellen kein Problem dar, wenn dadurch verschiedene Variable oder Ableitungen miteinander verknüpft werden.

Algebraische Schleifen

Wenn zwei oder mehrere Blöcke, in denen die Eingangsgröße direkt durchgereicht wird, eine Rückkopplung bilden, muß die Schleife iteriert werden. Solche Schleifen verlangsamen den Lösungsprozeß bzw. haben keine Lösung und sollten daher vermieden werden.

Beispiel:



Dies entspricht der Gleichung

$$x = 5x + 2$$

Algebraische Schleifen

Startet man die Schleife, wird die Meldung

```
Warning:Block diagram ´name´ contains 1 algebraic
loop(s).
Found algebraic loop containing block(s):
          ´name/Gain´
          ´name/Sum´(algebraic variable)
```

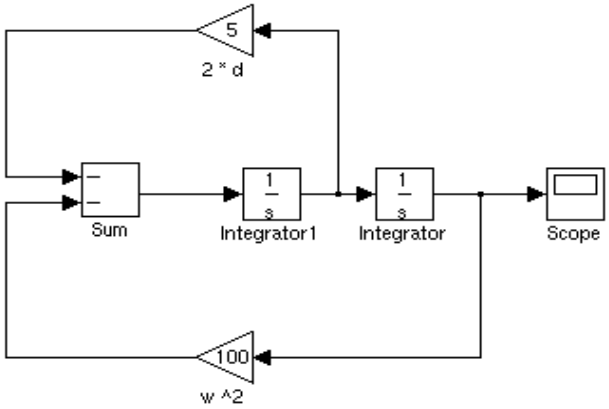
ausgegeben. Falls nach 200 Iterationen keine Konvergenz erreicht wird, bricht Simulink ab.

Für algebraische Schleifen sind folgende Blöcke kritisch:

- Verstärker-Blöcke (Gain) und die meisten nichtlinearen Blöcke
- Transfer-Blöcke, wenn der Zähler von derselben Ordnung wie der Nenner ist
- Zero-Pole-Blöcke, wenn gleichviele Nullstellen wie Pole vorkommen
- State-Space Blöcke, wenn die D-Matrix ungleich Null ist

3. Beispiel

$$\ddot{x} + 2 \cdot d \cdot \dot{x} + \omega^2 \cdot x = 0$$



4. Beispiel

Oeko-System: Zwei Populationen,
Tang (T) und Seeschnecken (S)

$$\dot{T} = k_1 \cdot T - k_2 \cdot T^2 - k_3 \cdot T \cdot S$$

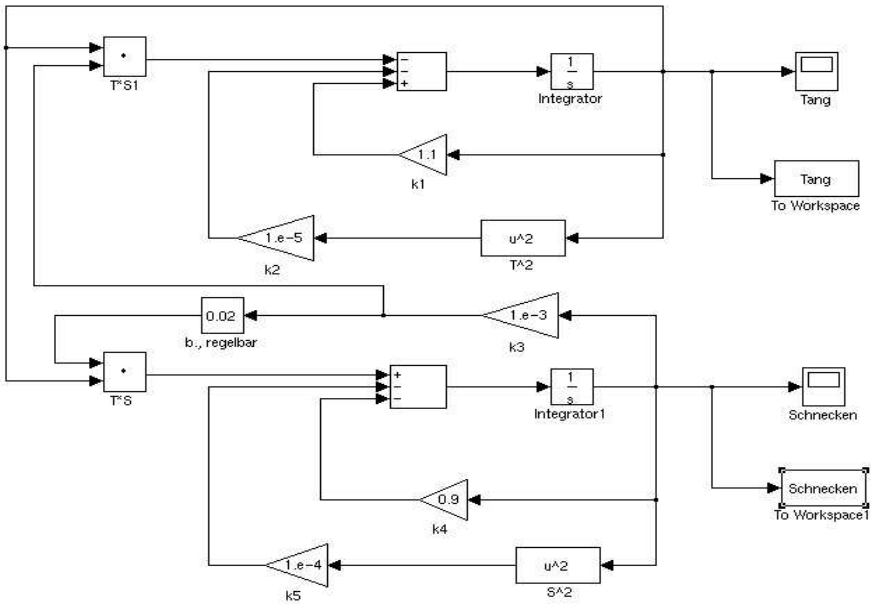
$$\dot{S} = b \cdot k_3 \cdot T \cdot S - k_4 \cdot S - k_5 \cdot S^2$$

$$T_0 = 10, S_0 = 100$$

$$k_1 = 1{,}1, k_2 = 10^{-5}, k_3 = 10^{-3}, k_4 = 0{,}9, k_5 = 10^{-4}$$

$$b = 2 \times 10^{-2}$$

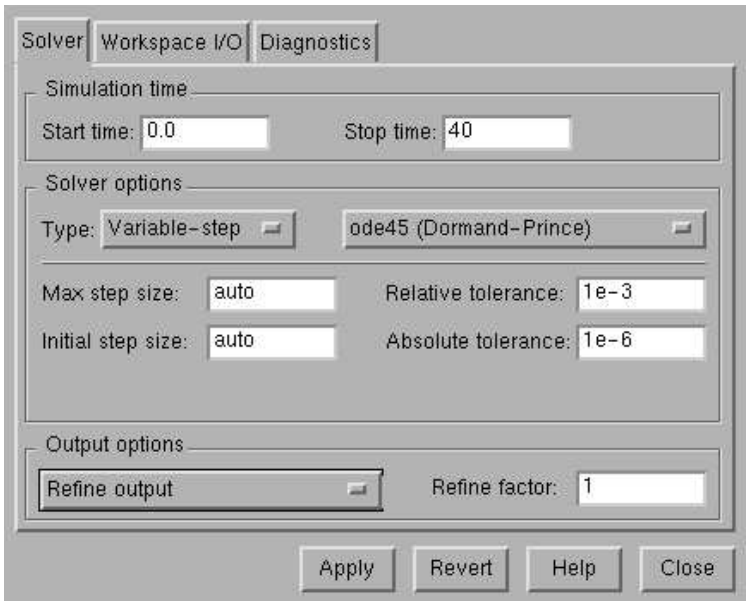
4.Beispiel

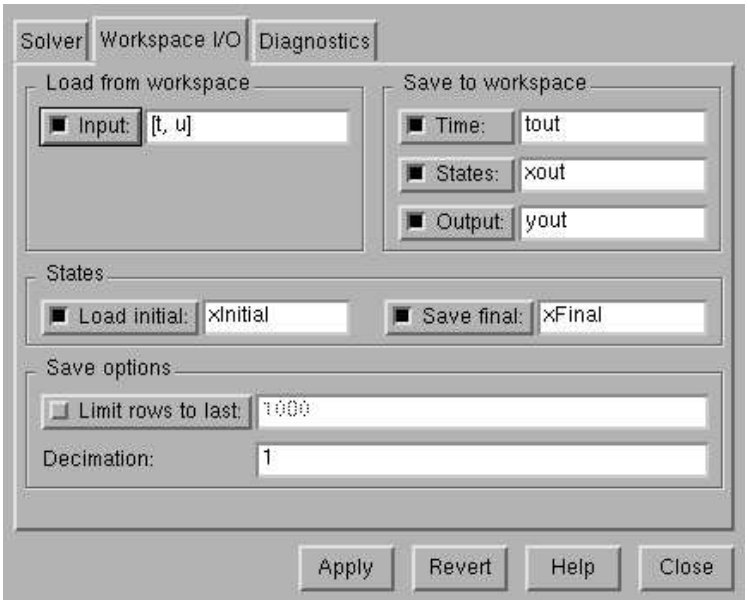


Ausführen von Simulink-Modellen

Ein Simulink-Modell wird gestartet, indem es aus dem `Simulation` Menü heraus gestartet wird.

- Im `Solver` Panel werden Steuerparameter für den Simulationslauf gesetzt
  - Integrationsalgorithmus
  - Integrationsschrittweite
  - Fehlertoleranzen
  - Ausgabefrequenz


Solver
Workspace I/O
Diagnostics
Simulation time
Start time: 0.0
Stop time: 40
Solver options
Type: Variable-step
ode45 (Dormand-Prince)
Max step size: auto
Relative tolerance: 1e-3
Initial step size: auto
Absolute tolerance: 1e-6
Output options
Refine output
Refine factor: 1
Apply
Revert
Help
Close


Solver
Workspace I/O
Diagnostics
Load from workspace
Input: [t, u]
Save to workspace
Time: tout
States: xout
Output: yout
States
Load initial: xInitial
Save final: xFinal
Save options
Limit rows to last: 1000
Decimation: 1
Apply
Revert
Help
Close

Im `Workspace I/O` Panel wird die Ausgabeinformation spezifiziert. Im Bereich `Save to Workspace` je nach aktivierter Checkbox

- t und x werden in den Workspace ausgegeben
- y wird nur dann in den Workspace ausgegeben, wenn es im Blockmodell in einen *Outport*-Block geleitet wird
- im Bereich `Loading from Input` können Variable und Matlab-Funktionen über *Inport*-Blöcke eingelesen werden.
- es können Anfangsbedingungen eingegeben werden, die die entspechenden Werte in den Blöcken überschreiben
- Sicherung des Endzustands

Funktionale Beschreibung

Simulink-Modelle können auf verschiedene Arten beschrieben werden:

Model-File	Textuelle Beschreibung des Blockmodells: *model*.mdl
S-Function	Matlab-Beschreibung des Modells: *model*.m
C-Code	Modellbeschreibung als C-Programm, Übersetzung in ein Matlab CMEX-File: *model*.mex

S-Functions können als Blocksymbol in Simulink integriert werden.

Alle 3 Versionen werden aus Matlab heraus gestartet.

Aufruf:

`[t,x,y,]=sim('model',timespan,options,ut,`*`parameter`*`)`

`model`	erster Namensteil des Modell-Filenamens (in Hochkomma)
`timespan`	Simulationsendzeitbereich; hier muß hier ein Vektor aus Start- und Endzeitpunkt eingegeben werden: `[tstart,tend]`
`options`	wird durch das `simset` Kommando erzeugt.
`ut`	Eingänge, die sich auf *Inport*-Blöcke im Modell beziehen; hier steht der Name einer Matrix mit zwei Spalten, bei der die erste Spalte die Zeitpunkte und die zweite Spalte die zugehörigen Eingangswerte enthält. Hier kann auch (in Hochkomma) ein mathematischer Ausdruck in Matlab-Syntax stehen, z.B. `'sin(3*t)'`
`parameter`	steht für eine Liste von Parametern, die an die S-Funktion durchgereicht werden.

Wenn ein Blockmodell gestartet wird, reicht die Angabe des Dateinamens völlig aus, da alle anderen Parameter im Blocksystem spezifiziert sind.

Die Ergebnisse werden nicht automatisch geplottet, selbst wenn *Scope*-Blöcke im Modell vorhanden sind.

Plots müssen über die bekannten Matlab-Funktionen erzeugt werden. Dazu werden die Variablen im Workspace benutzt.

Laplace-Transformation

f(t) ist eine reelle Funtion einer reellen Variablen *t*.

$$F(s) = \mathrm{L}\,[f(t)] = \int_0^\infty f(t)e^{-st}dt$$

heißt die Laplace-Transformierte von *f(t)*.

Eigenschaften:

- Skalierung: $\mathrm{L}[f(t/c)] = c\,F(c*s)$
- Linearität: $\mathrm{L}[c_1 f_1(t) + c_2 f_2(t)] = c_1 F_1(s) + c_2 F_2(s)$
- Ableitung: $\mathrm{L}[f^{(n)}(t)] = s^n F(s)$
- Integrale: $\mathrm{L}[\int_0^t \dots \int_0^t \dots \int_0^t f(t)dt^n] = s^{-n} F(s)$

Transfer-Funktion

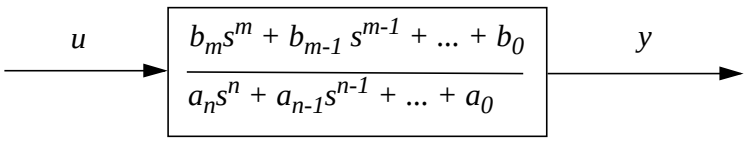
Allg. Differentialgleichung n-ter Ordnung:

$$a_n y^{(n)}(t) + a_{n-1}y^{(n-1)}(t) + ...+ a_0 y(t) = b_m u^{(m)}(t) + b_{m-1}u^{(m-1)}(t) + ... + b_0 u(t)$$

mit m $\leq$ n.

Laplace-Transformation:

$$Y(s) = \frac{b_m s^m + b_{m-1}\,s^{m-1} + ... + b_0}{a_n s^n + a_{n-1}s^{n-1} + ... + a_0}\;U(s)$$



Linearer Zustandsraum

Spezialfall: Lineare DGl n-ter Ordnung,

Normierung, so daß $a_n = 1$ ist

$$y^{(n)}(t) + a_{n-1}\, y^{(n-1)}(t) + ... + a_0\, y(t) = b_0\, u(t)$$

Definiere $x_1 = y$

$$x_2 = \dot{x}_1 = \dot{y}$$

$$\vdots$$

$$x_n = \dot{x}_{n-1} = y^{(n-1)}$$

Daraus folgt kompakt:

$$\dot{x} = Ax + bu$$

$$y = Cx$$

- die x_i heißen **Zustandsvariable** bzw. x ist der **Zustandsvektor**
- u und y sind **Eingangs-** bzw. **Ausgangsvektoren**

$$A = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ -a_0 & -a_1 & -a_2 & \dots & -a_{n-1} \end{bmatrix} \qquad b = \begin{bmatrix} 0 \\ 0 \\ \dots \\ b_0 \end{bmatrix} \qquad C = \begin{bmatrix} 1 & \dots & 0 & 0 \end{bmatrix}$$

Konvention in *MATLAB/SIMULINK*:

$$y = b_0 x \quad => \quad b = \begin{bmatrix} 0 \\ 0 \\ \dots \\ 1 \end{bmatrix} \qquad C = \begin{bmatrix} b_0 & \dots & 0 & 0 \end{bmatrix}$$

Reihenfolge der x_i ist manchmal umgekehrt.

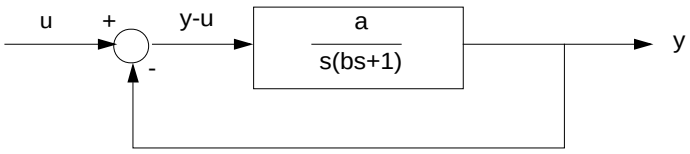
MATLAB-Kommandos

`[A,B,C,D]=linmod('modell')`
erzeugt die Matrizen für die lineare Zustandsraum-Darstellung des kompletten Modells

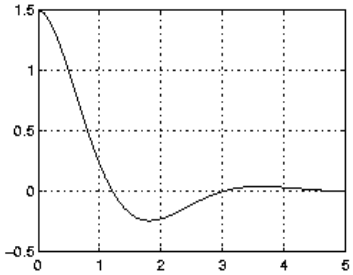
`[zähler,nenner]=ss2tf(A,B,C,D)`
berechnet aus den Matrizen A,..D den Zähler und Nenner für die entsprechende Transfer-Funktion

`[A,B,C,D] = tf2ss(zähler,nenner)`
berechnet die Matrizen des linearen Zustandsraums aus den Zähler- und Nenner-Polynomen

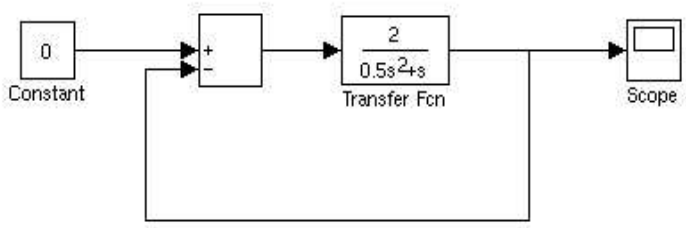
Beispiel: Linearer Servo



a=2; b=0.5; u(t)=0; y(0)=1.5



1. Lösung der Transfer-Funktion



Anfangswert für y muß über die `xInitial` Variable im Workspace I/O Panel eingegeben werden. Diese ist aber der Anfangswert der Zustandsvariable des Systems. Mittels

```
>> tf2ss([2],[0.5 1 0])
```

findet man `C=[0 4]` und somit $y = 4x$ ==> `xInitial = 1.5/4`

2. Lösung der Differentialgleichung

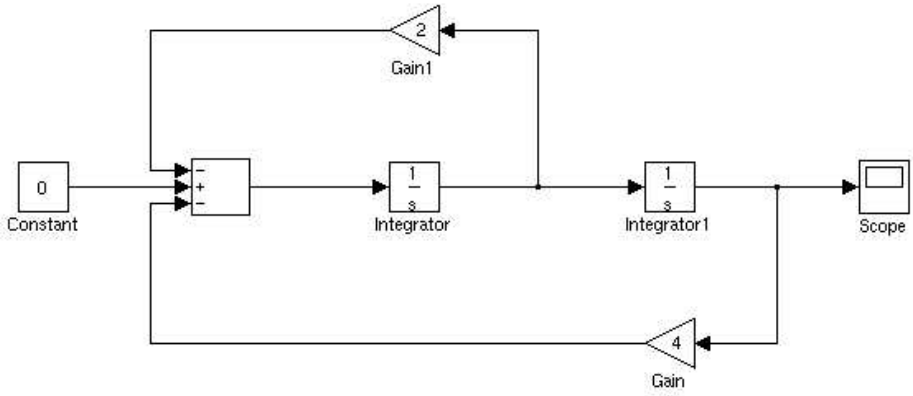
Aus dem Blockschaltbild findet man für die Laplace-Transformierten:

$$\frac{Y(s)}{U(s)-Y(s)} = \frac{a}{bs^2+s} = \frac{4}{s^2+2s} \qquad \text{oder} \qquad s^2Y + 2sY + 4Y = 4U$$

Daraus folgt: $\ddot{y} + 2\dot{y} + 4y = 4u$

Die Anfangsbedingung $y(0) = 1.5$ wird entweder über die `xInitial` Variable gesetzt oder im `Integrator1`. Dazu doppelklickt man auf das Symbol und trägt im folgenden Menü den Anfangswert ein.

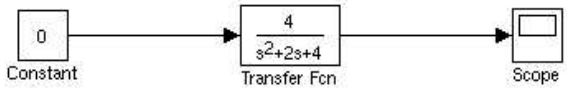
$u = 0$ wird explizit über eine konstante Quelle verifiziert, kann aber auch wegfallen.



3. Transferfunktion des Systems

Laplace-Transformation der o.g. Differentialgleichung ergibt

$$\frac{Y}{U} = \frac{4}{s^2 + 2s + 4}$$



Anfangsbedingungen: über `xInitial`

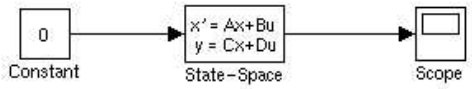
wegen $y = 4x$ ==> `xInitial=1.5/4`

4. State-Space Block in Simulink

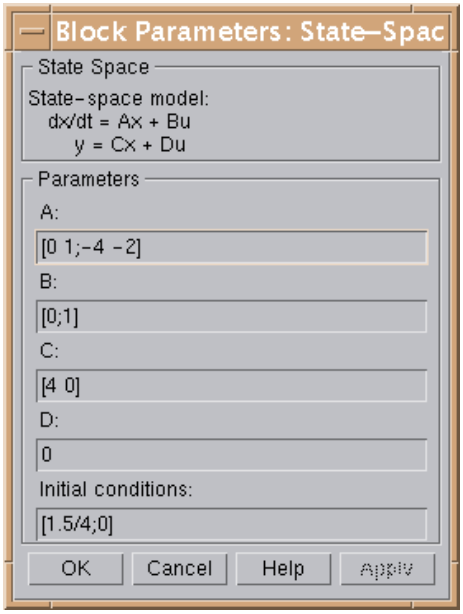
Aus der o.g. Transfer-Funktion findet man das lineare System

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -4 & -2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u$$

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 4 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$



Eingabe der Zustandmatrizen
und Anfangszustand
(oder über `xInitial`)



Weitere nützliche Kommandos

im MATLAB Fenster:

`>>who` listet die Namen aller Variablen im Workspace

`>>`*`var`* Eingabe eines Variablennamens listet den oder die Werte

`>>`*`var`*`[ ]` löscht die Werte der Variablen *`var`* (Initialisierung)

`>>clear` löscht alle Variablen aus dem Workspace

`>>plot(t,`*`var1`*`,t,`*`var2`*`,....)`
plottet die Variablen *`var1, var2, ....`* alle in einem Diagramm

Weitere nützliche Kommandos

`>>[size,x0,xstr]=modell`
listet die Modellcharakteristika. Die Ausgabevariable haben folgende Bedeutung:

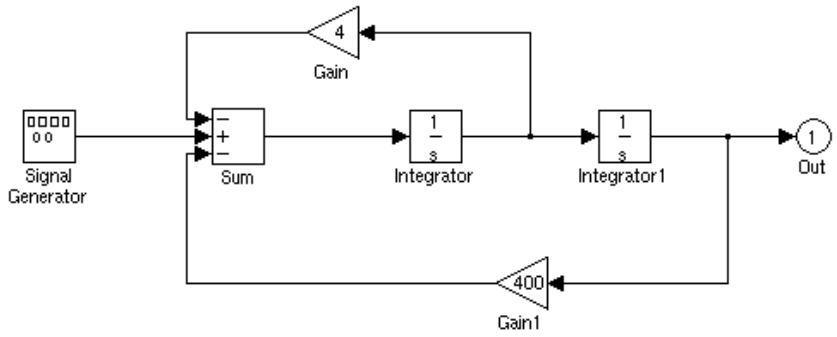
`size` ist ein 7-komponentiger Vektor, wovon hier nur die ersten Komponenten von Belang sind:

- Anzahl kontinuierlicher Zustände
- Anzahl diskreter Zustände
- Anzahl der Outputs
- Anzahl der Inputs

`x0` enthält die Anfangswerte des Zustandsvektors

`xstr` ist ein Textstring, der die Reihenfolge der Komponenten des Zustandsvektors den Blöcken zuordnet.

Beispiel: `osz.m`




```
>> [size,x0,xstr]=osz
size =            xstr     =
     2            /osz/integrator1            x1
     0            /osz/integrator             x2
     1
     0
     0
     0
     1
x0 =
     1
     0
```