3. Einführung in Simulink

Wesentlich erweitert wurde MATLAB durch Simulink, einem Software-Paket für die Modellbildung, Simulation und Analyse linearer und nichtlinearer sowie zeitkontinuierlicher und zeitdiskreter Systeme.
Die Elemente des nachzubildenden Systems werden als Blöcke, die mittels Linien als Elemente des Informationsflusses untereinander verkoppelt werden können, dargestellt. Damit ist es möglich, wie mit einem Bleistift auf Papier – hier mit der Maus auf dem Bildschirm, die gewünschten mathematischen Modelle graphisch darzustellen. Aus einer sehr umfangreichen Bibliothek können für die Modellbildung die erforderlichen Funktionsblöcke ausgewählt und über Linien, entsprechend dem Informationsfluss, miteinander verbunden werden.
Die bei einer Simulation zwischen den einzelnen Blöcken ausgetauschten Daten können, für MATLAB typisch, vom Typ Skalar, Vektor oder Matrix sein, d.h. es wird zwischen eindimensionalen – 1-D – und zweidimensionalen – 2-D – Arrays bzw. Signalen unterschieden.
Simulink ist ein interaktives Programm, das es neben der Vielzahl der gelieferten Funktionen erlaubt, eigene vom Anwender mit Hilfe des M-file-Editors geschriebene Funktionen zu nutzen. Die eigenen Funktionen laufen unter der Bezeichnung "User-Defined Functions" wie z.B. die *S-Functions*.
Die Mehrzahl der Simulinkoperationen ist selbsterklärend und besonders für Anwender, die mit der Konstruktion von Signalflussplänen vertraut sind, sehr schnell zu erlernen. Nachfolgend werden einige grundsätzliche Eigenschaften von Simulink beschrieben.
Für eine Vielzahl von Anwendungsgebieten existieren Toolboxen mit aussagefähigen Modellblöcken.
Die Modelle können beschriftet und farbig gestaltet werden.

3.1 Der Funktionsblock

Die Funktionsblöcke sind in den meisten Unterbibliotheken von Simulink, bis auf die Bibliotheken *Sinks*, *Sources*, *Model Verification* und *Model-Wide Utilities*, enthalten.
Der Funktionsblock ist dadurch gekennzeichnet, dass er mit einem Signal oder mehreren Signalen – Eingangsvektor – belegt werden kann und dass ein Signal oder mehrere Signale – Ausgangsvektor – ihn verlassen:

Bild 3.1 Funktionsblock

Für das Ausgangssignal gilt:

$$y(t) = f\left[u(t)\right] \tag{3.1}$$

bzw. für lineare Systeme im Frequenzbereich:

$$Y(s) = F\left[U(s)\right] \tag{3.2}$$

3.2 Eingabe- und Ausgabeblöcke

3.2.1 Übergabe von Daten der Eingangssignale an das Modell

Die Blöcke mit den Eingangssignalen für die Modellierung sind in der Unterbibliothek *Sources* zu finden. Die *Sources*-Blöcke sind dadurch gekennzeichnet, dass sie nur über einen Ausgangsvektor verfügen. Hier können die Signale aus dafür vorgesehenen Blöcken:

Bild 3.2 allgemeiner Eingangsblock

im Modellfenster erzeugt oder die notwendigen Daten aus einem *file* *.*mat* bzw. von der *Workspace* bezogen werden:

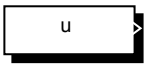

Bild 3.3 Eingabeblock für Daten aus einem file *.*mat* oder von der Workspace

Zu beachten sind die Anordnung der zu übergebenden Werte als Zeitvektor und Datenvektor bzw. Datenvektoren. Die Anordnung ist den Beschreibungen in den Fenstern der *Source Block Parameters* für die Blöcke *From File* bzw. *From Workspace* zu entnehmen.
Eine weitere Möglichkeit das Modell im Simulations-Fenster mit den notwendigen Eingangswerten von der *Workspace* aus zu versorgen, ist der Block *In*:

Bild 3.4 Eingabeblock für Daten von der Workspace

Voraussetzung dafür ist, dass neben dem Vorhandensein der Daten auf der Workspace, im Modellfenster unter *Simulation → Configuration Parameters → Data Import/Export* im Fenster *Load from Workspace* das Feld *Input* aktiviert ist und dahinter die entsprechenden Bezeichnungen der Datenvektoren – Spaltenvektoren – eingetragen sind. Auch können Vektoren mit den Anfangszuständen vorgegeben und damit eingegeben werden.

3.2.2 Darstellung der Ergebnisse oder Ausgabe der Daten der Simulation

Die Ergebnisse der Simulation lassen sich direkt im Modellfenster graphisch mit Hilfe des *Data Viewers* darstellen, z.B. durch:

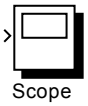

Bild 3.5 Block zur graphischen Darstellung des Ergebnisses der Simulation

oder indirekt als Datensatz einem *file* *.*mat* übergeben bzw. an die *Workspace* ausgegeben werden. Die dazu erforderlichen Blöcke *To File* und *To Workspace* sind in der Unterbibliothek *Sinks* enthalten. Sie verfügen nur über einen Eingangsvektor:

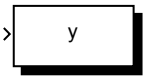

Bild 3.6 Ausgabeblock für Daten an einen file *.*mat* oder an die Workspace

Auch die Ausgabe von Daten an die Workspace kann über einen Block, genannt *Out*:

Bild 3.7 Ausgabeblock für Daten an die Workspace

erfolgen. Voraussetzung ist, dass im Modellfenster unter *Simulation → Configuration Parameters → Data Import/Export* im Fenster *Save to Workspace* die Felder *Time* und *Output* aktiviert und dahinter die entsprechenden Bezeichnungen der Datenvektoren, in die die Daten geschrieben werden sollen, eingetragen sind. Neben dem Vektor für die Zeit *t* – *Time*, den Vektoren für die Ausgangsdaten – *Output* – $y_1, \ldots, yr$ und den Vektoren für die Zustände – *States* – $x_1, \ldots, xn$ kann auch ein Vektor mit den Endwerten der Zustände – *Final states* – $xa_1, \ldots, xan$ mit ausgegeben werden. Die Vektoren ergeben sich aus den für jeden Rechen- bzw. Zeitschritt ermittelten Werten. Das Datenformat kann eingestellt werden:

$$\text{als Array mit } \begin{cases} t_a = [t_1; \cdots; t_k] & x_a = [x_{a1}; \cdots; x_{an}] \\ y_1 = [y_{11}; \cdots; y_{1k}] & x_1 = [x_{11}; \cdots; x_{1k}] \\ \vdots & \vdots \\ y_r = [y_{m1}; \cdots; y_{mk}] & x_n = [x_{n1}; \cdots; x_{nk}] \end{cases} \text{ als Spaltenvektoren} \qquad (3.3)$$

bzw.

$$\text{als Struktur mit } \begin{cases} t_a = [t_{a1}; \cdots; t_{an}] \\ y_1 = \begin{cases} \text{time}: [\ ] \\ \text{signals}: [1 \times 1 \text{ struct}] \end{cases} \quad x_a = \begin{cases} \text{time}: [\ ] \\ \text{signals}: [1 \times m \text{ struct}] \end{cases} \\ \vdots \qquad\qquad x = \begin{cases} \text{time}: [\ ] \\ \text{signals}: [1 \times m \text{ struct}] \end{cases} \\ y_r = \begin{cases} \text{time}: [\ ] \\ \text{signals}: [1 \times 1 \text{ struct}] \end{cases} \quad m: \text{Anzahl der Systeme} \end{cases} \tag{3.4}$$

Der Eintrag "*time*: [ ]" bedeutet, dass es sich um einen Leer-Vektor bzw. eine Leer-Matrix – Empty-Matrix – handelt.
Wird als Format für den Ausgang *Structure with time* gewählt, so wird in Gleichung (3.4) an Stelle der Empty-Matrix "*time*: [ ]", "*time*: [$k \times 1$ double]" ausgegeben, wenn k die Anzahl der Rechenschritte bedeutet.
Bei dem Format *Array* ergeben sich die Werte durch den Aufruf des Funktionsnamens in der *Workspace*. Erfolgte dagegen die Ausgabe mit dem Format *Structure* bzw. *Structure with time*, so kann wie folgt auf die Daten zugegriffen werden, z.B.

bei y_1 mit *signals*: [1 × 1 struct]:
    y_1.time
    y_1.signals.values
und bei x mit *signals*: [1 × m struct] auf das *i-te* von m Systemen:
    x.time
    x.signals(i).values

Neben dem Feld *values* gibt es noch die Felder *dimensions*, *label*, *blockName* und *inReferencedModel*.

3.3 Signalverbindungen – Informationsaustausch

Die Informationen bzw. Daten werden als Signale über die Signalleitungen zwischen den einzelnen Funktionsblöcken ausgetauscht. Die Darstellung der Signalleitungen erfolgt in der einfachsten Form durch eine gerichtete Linie. Mit Hilfe des Summiergliedes *Sum* können mehrere Signale – Skalare oder Vektoren – vorzeichengerecht addiert werden:

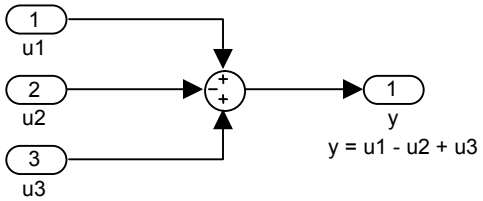

Bild 3.8 Vorzeichengerechte Summation von Signalen

Signalverzweigungen sind, wie in Signalflussplänen, durch einen Punkt gekennzeichnet, von dem die Signale, welche gleich dem zufließenden Signal sind, abführen. Einen besonderen Block gibt es dafür nicht.

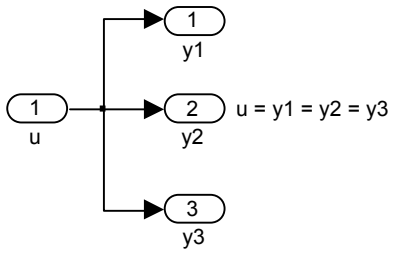

Bild 3.9 Signalverzweigung

Mit dem *Mux*-Block werden mehrere Signale, deren Anzahl unter *Number of inputs* in den *Function Block Parameters* eingetragen wird, zu einem virtuellen[1] Signal zusammengefasst bzw. gebündelt, was der übersichtlicheren Darstellung in einem Simulink-Signalflussplan dient. Dieses virtuelle Signal kann dann einem Funktions- oder Ausgabe-Block zur Verarbeitung bzw. dem *Demux*-Block zwecks Zerlegung in seine Bestandteile übergeben werden. Die Anzahl der Ausgänge muss in *Number of outputs* in die *Function Block Parameters* eingetragen werden. Mit Hilfe eines *Demux*-Blockes kann ein virtuelles Signal in mehrere virtuelle Signale, die Teil des Gesamtsignals sind, zerlegt werden.

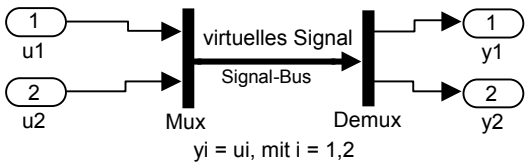

Bild 3.10 *Mux* zum Bilden eines Signalbusses und Demux zum Zerlegen in die Einzelsignale

Steht z.B. an einem *Demux*-Block ein aus *n* Einzelsignalen gebildetes virtuelles Signal an, dann wird dieses in seine *n* Einzelsignale zerlegt, in dem unter *Number of outputs* in die *Function Block Parameters* der Wert *n* eingetragen wird. Sollen dagegen am Ausgang *m* virtuelle Signale ausgegeben werden, so ist das Eingangssignal entsprechend auf die *m* Ausgangssignale aufzuteilen und dafür folgendes unter *Number of outputs* in die *Function Block Parameters* einzutragen:

$$\begin{bmatrix} n_1 & n_2 & \cdots & n_{m-1} & n_m \end{bmatrix} \tag{3.5}$$

Hierbei muss gelten:

$$\sum_{i=1}^{m} n_i = n \tag{3.6}$$

[1] virtuell: scheinbar, nur gedacht

Beispiel 3.1
Das einem *Demux*-Block zugeführte virtuelle Signal besteht aus 9 Signalen. Am Ausgang sollen die Signale 1 bis 3, 4 und 5, 6 sowie 7 bis 9, als 4 virtuelle Teilsignale ausgegeben werden. Was ist unter *Number of outputs* in die *Function Block Parameters* einzutragen?

Lösung:

Number of outputs: $\begin{bmatrix}3 & 2 & 1 & 3\end{bmatrix}$

Zwei weitere Blöcke, die ebenso, wie die soeben behandelten, der Vereinfachung und übersichtlicheren Darstellung in den Simulink-Signalflussplänen dienen, sind der *Bus Creator*-Block und der *Bus Selector*-Block, die nachfolgend kurz behandelt werden.
Mit dem *Bus Creator*-Block werden mehrere Signale, Einzelsignale oder bereits zu einem Bus zusammengefasste Signale, zu einem Bus, ähnlich wie bei dem *Mux*-Block, gebündelt, d.h. zu einer Gruppe von Signalen zusammengefasst und durch eine einzelne Linie im Signalflussplan dargestellt. Die Anzahl der zu einem Bus zu bündelnden Signale sind unter *Number of inputs* in die *Function Block Parameters* einzutragen. Über den Schalter *Find* in dem Fenster *Signals in bus* in den *Function Block Parameters* wird, nach dem Anklicken des betreffenden Signals, die Quelle dieses Signals im Signalflussplan farbig gekennzeichnet.
Um das Bussignal am Ausgang wieder zu zerlegen, ist es an einen *Bus Selector*-Block anzuschließen.
Der *Bus Selector*-Block stellt eine vom Anwender festgelegte Teilmenge seiner Eingangssignale an seinem Ausgang bereit. Die ausgewählten Einzelsignale können einzeln oder als ein neues Bussignal – *Output as bus* – ausgegeben werden. Diese beiden Blöcke sind durch ihre Variabilität sehr leistungsfähig.

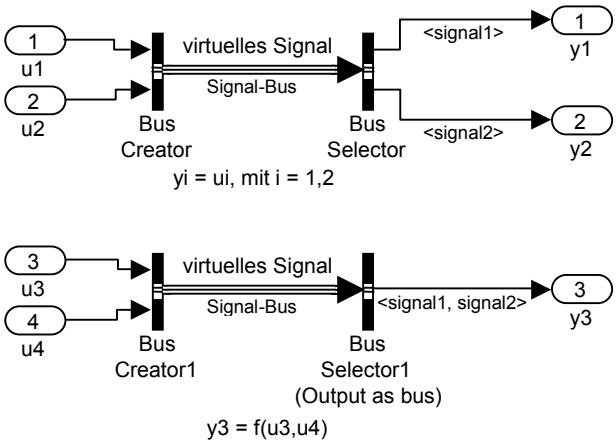

Bild 3.11 *Bus Creator* zur Bündelung und *Bus Selector* zur Zerlegung von Signalen

Zu bemerken ist noch, dass bei dem Anklicken einer Signallinie im Signalflussplan mit der rechten Maustaste ein Fenster geöffnet wird, mit dem es u.a. möglich ist, über *Highlight To Source* den Verlauf zur Quelle bzw. mit *Highlight To Destination* den Verlauf zum Ziel des Signals farbig darzustellen.

3.4 Algebraische Schleifen – Algebraic Loops

3.4.1 Systeme mit proportionalem sprungfähigem Verhalten

Es gibt Systeme, deren Eingangssignale direkt auf den Ausgang durchgeschaltet werden. Dies sind zum einen Systeme mit einem statischen Übertragungsverhalten, d.h. deren Ausgangssignal wird ohne Zeitverzögerung vom Eingangssignal bestimmt. Zum Anderen sind es die sprungfähigen Systeme, die dadurch gekennzeichnet sind, dass in Differenzialgleichungen, die ihr dynamisches Verhalten beschreibenden, die Ordnung der höchsten zeitlichen Ableitung des Eingangssignals gleich der Ordnung der höchsten zeitlichen Ableitung des Ausgangssignals ist. Im Zeitbereich, bei der Beschreibung durch Vektor-Matrix-Differenzialgleichungen, ist in diesem Fall die Durchgangsmatrix $\mathbf{D} \neq \mathbf{0}$. Im Frequenzbereich, bei linearen Systemen, ist der Grad des Zählerpolynoms gleich dem Grad des Nennerpolynoms.
Unter Simulink werden Blöcke mit diesem Verhalten, als Blöcke mit *direct feedthrough* Verhalten bezeichnet. Es sind, neben den oben genannten sprungfähigen Systemen, die mathematischen Funktionsblöcke, der *Gain*-, *Product*- und *Sum*-Block sowie die direkte Verbindung zwischen dem Ausgang eines *Integrator*-Blocks mit seinen Anfangswert-Eingang.

3.4.2 Algebraische Schleifen

Befindet sich in einem Signalflussplan eine Rückkopplung, deren Vorwärts- und Rückwärtszweig nur aus einem oder mehreren dieser *direct feedthrough*-Blöcke aufgebaut ist, so erscheint bei der Simulation im Command Window folgende Meldung – *Warning: Block diagram 'Name' contains* 1 *algebraic loop(s)*.
Durch eine Rückkopplung wird bekanntlich das Ausgangssignal des Vorwärtszweiges über den Rückführzweig auf den Eingang des Vorwärtszweiges zurückgeführt. Da zwischen dem Eingangssignal und dem Ausgangssignal des Vorwärtszweiges und des Rückführzweiges keine Zeitverzögerung auftritt, gilt während eines Rechenschrittes:

Eingangssignal gleich Ausgangssignal gleich Eingangssignal oder
Ursache gleich Wirkung gleich Ursache!

Diese Konstruktion stellt eine *algebraische Schleife* bzw. einen *algebraic loop* dar. Die *algebraische Schleife* ist entweder durch Umgestaltung des Signalflussplanes aufzulösen oder durch Einfügen des *Algebraic Constraint*-Blockes zu unterbrechen und zu lösen.

3.4.3 Auflösen einer algebraischen Schleife

Grundsätzlich sind für das Auflösen von algebraischen Schleifen innerhalb eines Signalflussplanes für ein dynamisches System die Regeln zur Vereinfachung bzw. zur Umformung von Signalflussplänen zu verwenden. Es sollen dazu einige Hinweise gegeben werden.
Befinden sich im Vorwärts- und Rückführzweig einer Rückkopplung nur statische Übertragungsglieder, z.B. je ein *Gain*-Block, so kann diese algebraische Schleife wie folgt aufgelöst

werden, wenn K_v der Übertragungsfaktor im Vorwärtszweig und K_r der Übertragungsfaktor im Rückführzweig ist:

$$K_{res} = \frac{K_v}{1 \pm K_v K_r} \text{ mit } \begin{cases} + : \text{ Gegenkopplung} \\ - : \text{ Mitkopplung} \end{cases} \tag{3.7}$$

Beispiel 3.2
Simulink gibt bei der Simulation des folgenden Signalflussplans, siehe Beispiel3_02.m:

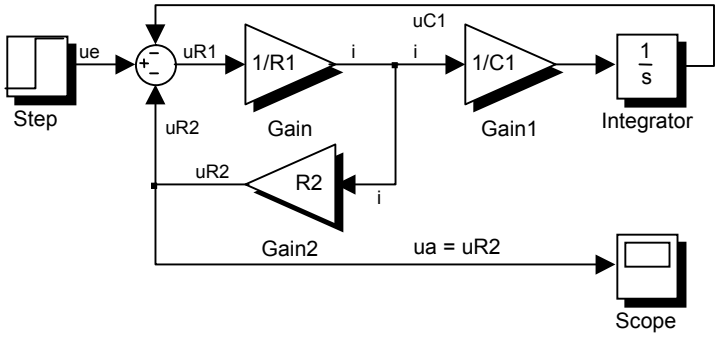

Bild 3.12 System 1. Ordnung mit einer algebraischen Schleife

im Command Window folgendes aus:

> Warning: Block diagram 'Bild3_12' contains 1 algebraic loop(s). …
> Found algebraic loop containing:
> 'Bild3_12/Gain'
> "Bild3_12/Gain2'
> 'Bild3_12/Sum' (algebraic variable)

Gesucht ist die resultierende Funktion, mit der die algebraische Schleife aufgelöst wird.

Lösung:

Die algebraische Schleife ist eine Gegenkopplung mit je einem *Gain*-Block im Vorwärts- und Rückführzweig, so dass sich mit Gleichung (3.7) für den Verstärkungsfaktor des resultierenden *Gain*-Blocks ergibt:

$$R_{res} = \frac{1}{R_1 + R_2} \tag{3.8}$$

Mit diesem resultierenden *Gain*-Block nach Gleichung (3.8) ergibt sich das Systemmodell im Bild 3.13 . Besteht dagegen die algebraische Schleife im Vorwärtszweig aus dem sprungfähigen System mit der Übertragungsfunktion:

$$G_v(s) = \frac{Z_v(s)}{N_v(s)} \quad \text{mit } m_v = n_v \tag{3.9}$$

und im Rückführzweig auch aus einem sprungfähigen System mit der Übertragungsfunktion: