

# **Einführung in die Software MATLAB<sup>®</sup> - Simulink<sup>®</sup> und die Toolboxen CARNOT und Stateflow<sup>®</sup> zur Simulation von Gebäude- und Heizungstechnik**

Sandra Lohmann

Oktober 2013

## 1 Inhalt

1	Inhalt .....	2
2	Einleitung .....	4
3	MATLAB.....	4
3.1	MATLAB-Desktop.....	5
3.2	MATLAB-Dateiformate .....	6
3.3	MATLAB-Hilfe .....	6
3.4	Rechnen mit MATLAB.....	7
3.5	Programmieren mit MATLAB.....	9
3.5.1	M-Skripte.....	9
3.5.2	M-functions .....	12
3.6	Grafiken erstellen .....	12
4	Simulink .....	15
4.1	Simulink Library Browser.....	15
4.2	Programmieren unter Simulink .....	16
4.2.1	Daten ansehen und Speichern .....	18
4.2.2	Verwenden von Variablen und Rechnen in Masken .....	19
4.2.3	Subsysteme .....	19
4.3	Modelle editieren.....	20
4.4	Durchführen einer Simulation .....	21
5	MATLAB- und Simulink-Shortcuts .....	22
6	Stateflow .....	23
6.1	Grundelemente eines Charts.....	25
6.1.1	Zustand (State) .....	25
6.1.2	Verbindung (Transition), Bedingungen .....	27
6.1.3	Verbindungspunkte (Connective Junctions) und Zeitverzögerung .....	27
6.2	Ein- und Ausgänge, Variablen.....	28
7	CARNOT .....	29
7.1	CARNOT-Hilfe.....	30
7.2	CARNOT-Demos .....	30
7.3	Vektoren.....	32
7.3.1	THV-Vektor .....	32
7.3.2	Weather data vector .....	33
7.3.3	AIV-Vektor.....	34
7.3.4	S-Vektor .....	34
7.4	CARNOT-Blöcke .....	34
7.4.1	Wetterdaten .....	34

---

7.4.2	Flachkollektor .....	36
7.4.3	Speicher .....	36
7.4.4	Wärmeübertrager .....	38
7.4.5	Brauchwasser-Zapfung .....	38
7.4.6	Pumpen.....	38
7.4.7	Regelung.....	39
7.4.8	Messung der Leistung und Energie .....	39
7.4.9	Daten ansehen und speichern.....	40
8	Literaturverzeichnis .....	41
9	Abbildungsverzeichnis.....	41
10	Tabellenverzeichnis .....	42

## 2 Einleitung

Diese Einführung in das Arbeiten mit MATLAB, Simulink, Stateflow und CARNOT beschreibt die grundlegende Funktionalität der Software und ihre Anwendung. Die Einleitung bezieht sich auf die MATLAB-Version R2012a, sollte aber für etwas ältere und neuere Versionen auch anwendbar sein.

Der Leser soll damit in der Lage sein, sich selbstständig in die Bedienung einzuarbeiten und grundlegende Aufgaben zu bewältigen. Vertiefende Informationen können (Angermann, 2009) bzw. der MATLAB- und CARNOT-Hilfe (siehe Kap. 3.3 und 7.1) entnommen werden.

## 3 MATLAB

MATLAB ist eine weitverbreitete, umfangreiche Software der Firma Mathworks für numerische Mathematik, Datenverarbeitung und Simulation. Der Name leitet sich von MATrix LABoratory ab, wodurch klar wird, dass MATLAB auf der Matrizenrechnung basiert.

MATLAB liefert eine eigene Programmiersprache, aber auch Schnittstellen zu anderen gängigen Programmiersprachen, wie C, Fortran und Java. Außerdem ist die Kommunikation mit Hardware, z.B. Messkarten möglich.

Zur Erweiterung der grundlegenden Funktionalität bietet Mathworks Toolboxes, z.B. Simulink zur grafischen Modellierung und Simulation physikalischer Systeme und Stateflow (Erweiterung von Simulink) zur Modellierung von Zustandsautomaten, die man zur Abbildung von Regelungsabläufen verwenden kann.

Die Toolbox CARNOT - eine Erweiterung der Simulationsumgebung MATLAB/Simulink - wurde ursprünglich vom Solar-Institut am Standort Jülich der FH Aachen entwickelt. Das Solar-Institut Jülich hat sich entschieden, diese Toolbox öffentlich und somit jedem Interessenten kostenfrei zugänglich zu machen. CARNOT enthält Blöcke aus dem Bereich der Gebäude- und Heizungstechnik und ermöglicht somit Simulationen dieser Systeme. CARNOT wird von der CARNOT-Usergroup weiterentwickelt. Diese Anleitung bezieht sich auf die Version 5.3.

### 3.1 MATLAB-Desktop

Der MATLAB-Desktop (siehe Abbildung 3.1) gliedert sich in folgende Bereiche/ Fenster, die sichtbar sein können, aber nicht müssen. Die Fenster können über den Menüpunkt *Desktop* geschlossen und geöffnet werden und durch drag- and drop mit dem Mauscourser unterschiedlich angeordnet werden.

1. *Command Window* - Hier werden Befehle eingegeben, die sofort ausgeführt werden und Eingaben in den Workspace gemacht. Zusätzlich gibt MATLAB über dieses Fenster Statusmeldungen aus.
2. *Command History* - In diesem Fenster werden die schon ausgeführten Befehle angezeigt und können durch einen Doppelklick nochmals ausgeführt werden.
3. *Current Directory* - Dies ist das Arbeitsverzeichnis, welches über die Menüleiste eingestellt wird. In diesen Ordner schreibt MATLAB die Simulationsdateien und -ergebnisse. Daten (z.B. Simulationsmodelle, Diagramme, Daten), die hier abgelegt wurden, können von hier aus auch wieder mit einem Doppelklick auf die entsprechende Datei geöffnet werden.
4. *Workspace Browser* – Der Workspace Browser zeigt alle im *Workspace* existierenden Variablen mit Name und Wert an. Handelt es sich um Matrizen bzw. Arrays mit mehreren Werten werden auch das Minimum und Maximum angezeigt. Externe Daten, wie z.B. Wetterdaten, können aus einer Datei in den *Workspace* geladen werden und stehen dann für die Simulation zur Verfügung. Außerdem können während einer Simulation Ergebnisse in den Workspace geschrieben und anschließend von hier aus in eine Datei exportiert werden.
5. *Editor* – Im Editor können Matlab-Skripte und Funktionen erstellt und bearbeitet werden (siehe 3.5.1).
6. *Variable Editor* – Anzeige und Bearbeitung von Variablen im Workspace. Von hier aus ist ein direktes Kopieren zu anderen Programmen (z.B. Excel) möglich. MATLAB tauscht dabei automatisch den Dezimalpunkt durch ein Komma aus.
7. *Shortcut-Leiste* – In der Shortcut-Leiste können Shortcuts mit ein oder mehreren Befehlen belegt werden, die sehr oft benötigt werden, z.B. zum Löschen des Workspace.

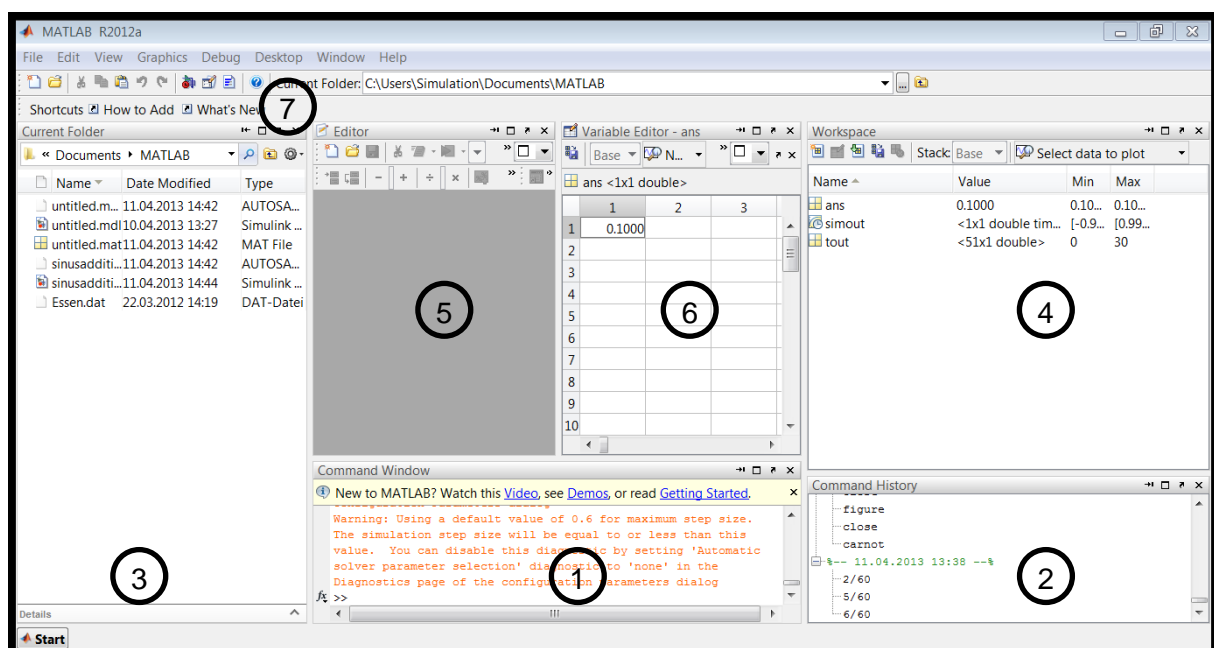


Abbildung 3.1 MATLAB-Desktop

### 3.2 MATLAB-Dateiformate

MATLAB verwendet eigene Dateiformate, die in Tabelle 3.1 aufgeführt werden.

Tabelle 3.1 MATLAB-Dateiformate

Dateiendung	Beschreibung
<b>mdl</b>	Simulink-Modell
<b>m</b>	M-Skript oder M-Funktion
<b>mat</b>	abgespeicherter Workspace oder einzelne Variable
<b>fig</b>	Grafikdatei

Bei der Vergabe von Dateinamen ist darauf zu achten, dass sie nur mit Buchstaben und nicht mit Ziffern beginnen dürfen, keine Umlaute (ä, ö, ü) enthalten und eine bestimmte Länge nicht überschreiten. Merkwürdige Fehlermeldungen, die keinen Grund zu haben scheinen, sind möglicherweise auf einen zu langen Dateinamen zurückzuführen.

MATLAB kann außerdem andere Dateitypen (z.B. \*.csv, \*.txt oder \*.xls(x) usw.) mit Hilfe von Importfunktionen oder des *Import Wizards* (Rechtsklick auf Datei > Import Data) einlesen, siehe Abbildung 3.2.

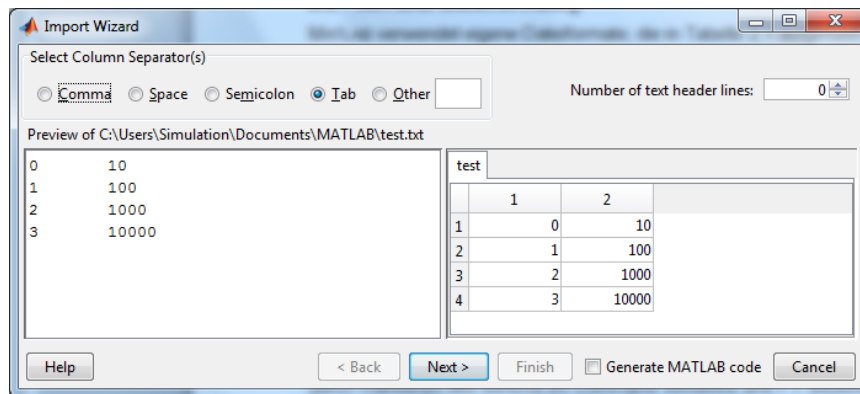


Abbildung 3.2 Import Wizard

### 3.3 MATLAB-Hilfe

Die MATLAB-Hilfe wird durch den Menüpunkt *Help* geöffnet. Hilfe zu einem Befehl erhält man durch markieren des Befehls im *Command Windows* und F1. Besonders hilfreich sind oft die Beispiele am Ende eines Hilfeeintrages.

### 3.4 Rechnen mit MATLAB

Rechnungen werden im *Command Window* durchgeführt. Beendet man den Befehl mit einem Semikolon wird das Ergebnis nicht im *Command Window* ausgegeben. Das Ergebnis der letzten Rechnung wird wie beim Taschenrechner unter der Variable *ans* im *Workspace* abgelegt. Berechnung können auch mit Variablen, hier *ans* durchgeführt werden (*ans+7*). Ergebnisse können unter einer Variablen, hier *a*, abgespeichert werden. Ein Beispiel ist in Abbildung 3.3 dargestellt.

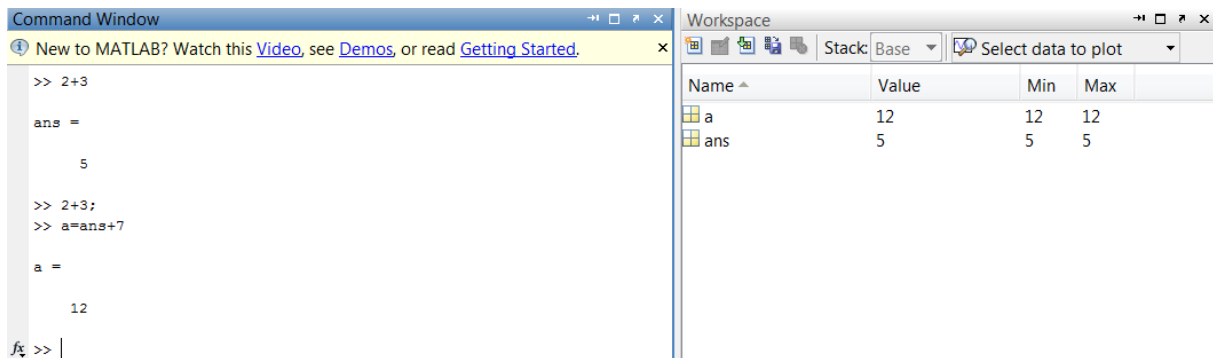


Abbildung 3.3 Grundlegende Berechnungen und Variablen

Variablen können auch Matrizen enthalten. Eine Matrix wird mit eckigen Klammern eingeleitet und abgeschlossen. Die Werte der Zeilen werden durch Leerzeichen oder Kommata, die Spalten durch Semikolon getrennt, siehe Abbildung 3.4. Der Zugriff auf einzelne Elemente einer Matrix erfolgt durch Indizes (beginnend bei 1). Zunächst wird die Zeile, dann die Spalte genannt.

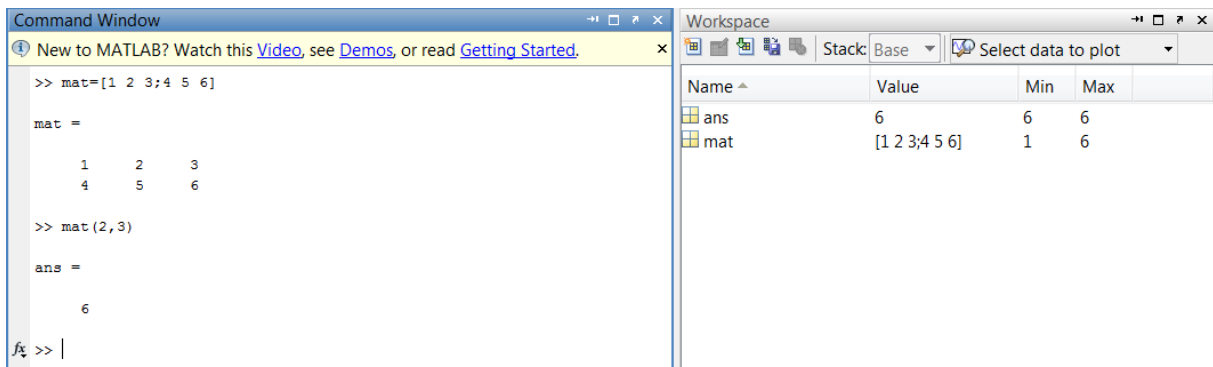


Abbildung 3.4 Matrizen erstellen

Elementweise Operationen werden durch den Punkt vor dem Rechenzeichen eingeleitet. Abbildung 3.5 zeigt die Addition zweier Zeilenvektoren ( $c=a+b$ ), die Matrixmultiplikation ( $d=a*b'$ ,  $b$  muss durch „ $'$ “ transponiert werden) und die elementweise Multiplikation ( $e=a.*b$ ).

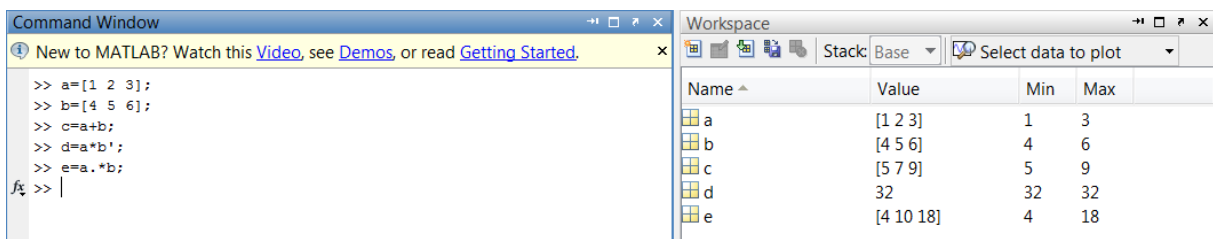
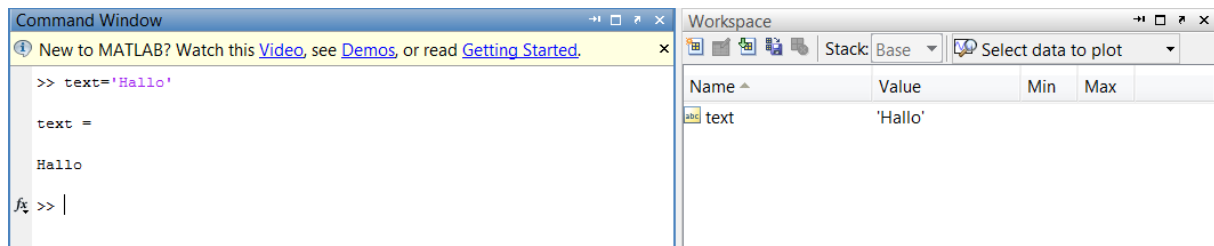


Abbildung 3.5 Matrixoperationen

Außer Zahlen unterstützt MATLAB noch diverse weitere Datentypen, z.B. Text (Strings). Ein String wird mit einem Hochkomma eingeleitet und abgeschlossen, siehe Abbildung 3.6



**Abbildung 3.6 String erstellen**

Wichtige Befehle zum Arbeiten mit dem *Command Window* und *Workspace*:

`clear`            löscht alle Variablen im *Workspace*

`clc`             löscht alle Ein- und Ausgaben im *Command Window*



## 3.5 Programmieren mit MATLAB

### 3.5.1 M-Skripte

Häufig verwendete Befehlskombinationen können mit dem *Editor* als M-Skripte (\*.m) abgespeichert werden. Hier ist auch die Programmierung von Schleifen und Bedingungen möglich.

#### For-Schleife:

Syntax:

```
for index = values
    program statements
    :
end
```

#### Beispiel:

Das M-Skript for-schleife.m (siehe Abbildung 3.7) beinhaltet zwei Zellen, die durch %% getrennt sind. In der ersten Zelle wird i mit dem Wert 1 bis 3 belegt (i=1:3) und im *Command Window* angezeigt. In der zweiten Zelle wird k mit den Werten 1 bis 3 mit der Schrittweite 0,5 belegt (k=1:0.5:3) und der jeweilige Wert mit dem Befehl disp() angegeben. Abbildung 3.8 zeigt die Ausgabe des Skriptes im *Command Window*.

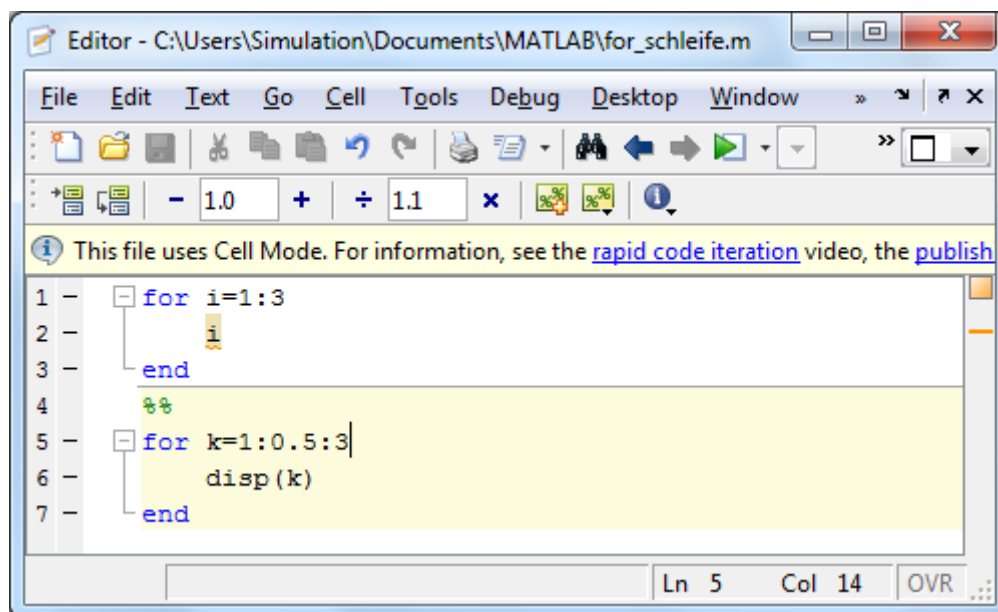
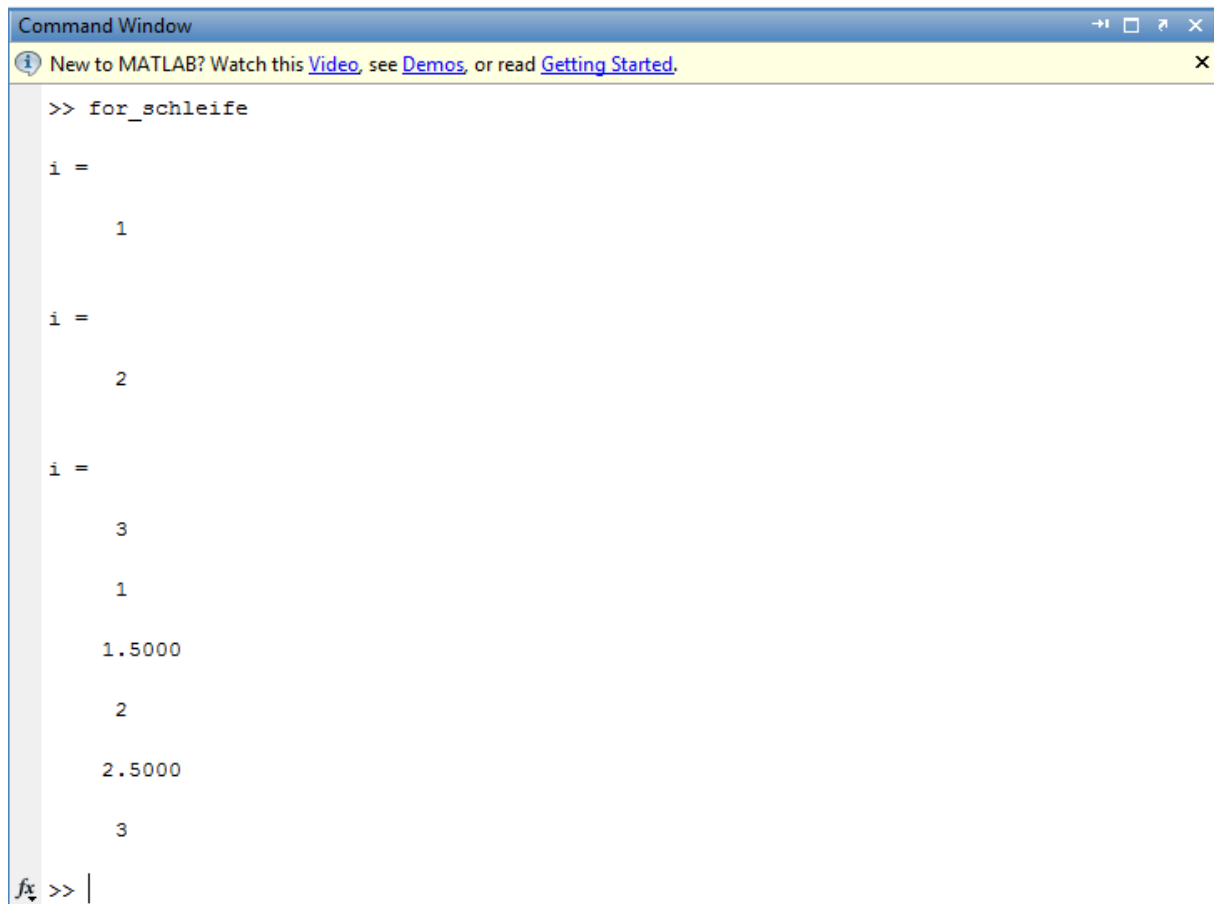


Abbildung 3.7 M-Skript for-schleife.m



```
>> for_schleife

i =

    1

i =

    2

i =

    3

    1

    1.5000

    2

    2.5000

    3

fx >> |
```

Abbildung 3.8 Ausgabe des M-Skript for-schleife.m

### If-Bedingung:

Syntax:

```
if expression
    statements
elseif expression
    statements
else
    statements
end
```

### Beispiel:

Das M-Skript if\_bedingung.m (siehe Abbildung 3.9) ist eine Erweiterung des M-Skript for-schleife.m. Wenn i größer als 1 ist wird der Text „i ist größer als 1.“ ausgegeben. Abbildung 3.10 zeigt die Ausgabe des Skriptes im *Command Window*.

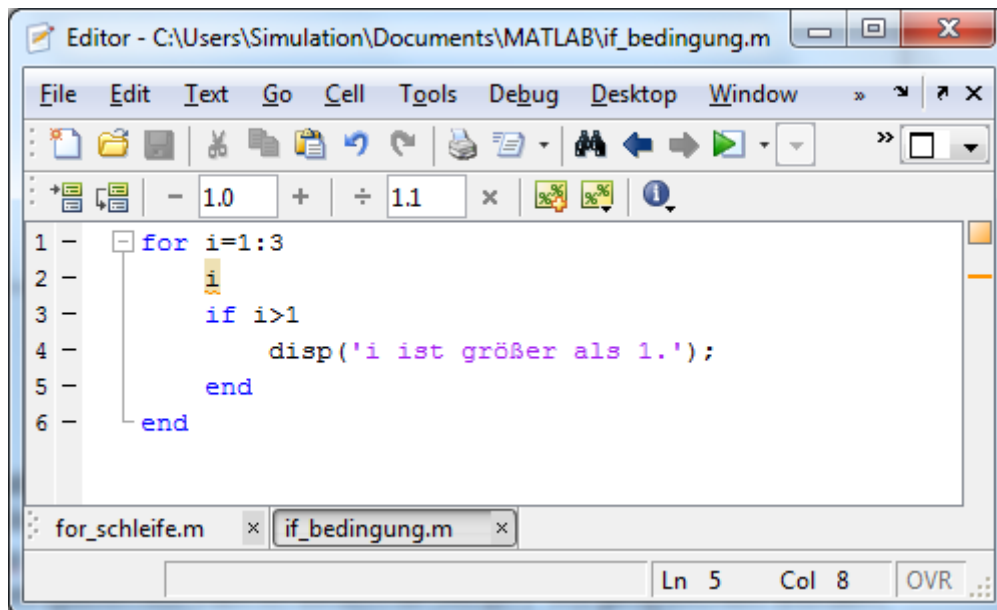


Abbildung 3.9 M-Skript if\_bedingung.m



Abbildung 3.10 Ausgabe des M-Skript if\_bedingung.m

Bei den meisten Simulationen ist es notwendig, Konstanten zu definieren oder Dateien (Wetterdaten, Kennlinien, o. a.) in den *Workspace* zu laden. Dies ist durch eine Befehlseingabe in das *Command Window* möglich, was bei mehreren Konstanten oder Dateien aufwendig ist. Die Befehlseingaben können in einem M-Skript gespeichert werden, so dass vor Simulationsbeginn nur das entsprechende M-Skript zu öffnen und auszuführen ist.

Ausgeführt wird ein M-Skript durch  im Editor, durch Drücken der Taste F5 bei geöffnetem Editor-Fenster oder durch Aufrufen des Skriptnamens im *Command Window*.

### 3.5.2 M-functions

MATLAB-Funktionen, sogenannte M-functions können auch selber programmiert werden.

Syntax:

```
function [out1, out2, ...] = myfun(in1, in2, ...)
    befehle
```

Es ist darauf zu achten, dass der Dateiname der M-Funktion exakt mit dem eingetragenen Funktionsnamen (hinter dem Gleichheitszeichen) übereinstimmt. In diesem Fall müsste der Dateiname also myfun.m heißen.

Vertiefende Informationen können (Angermann, 2009) bzw. der MATLAB- Hilfe (siehe Kap. 0) entnommen werden.

## 3.6 Grafiken erstellen

MATLAB beherrscht diverse Diagrammtypen in 2D und 3D. Der einfachste Befehl ist `plot(x,y)`, mit dem der Vektor `y` über dem Vektor `x` dargestellt wird.

Mit der Maus kann man plotten, indem man zunächst die Matrix `x` markiert, und dann Strg und die Matrix `y` markiert. Per Rechtsklick auf `y` und `plot(x,y)` wird die Grafik *Figure 1* erstellt und MATLAB gibt den verwendeten Befehl im *Command Window* aus.

Abbildung 3.11 zeigt die Erstellung zweier Vektoren `x` und `y` und den ausgegebenen Plot-Befehl. `x` enthält die Werte 0 bis 20 in 1er-Schritten, `y` ist  $x^2$ .

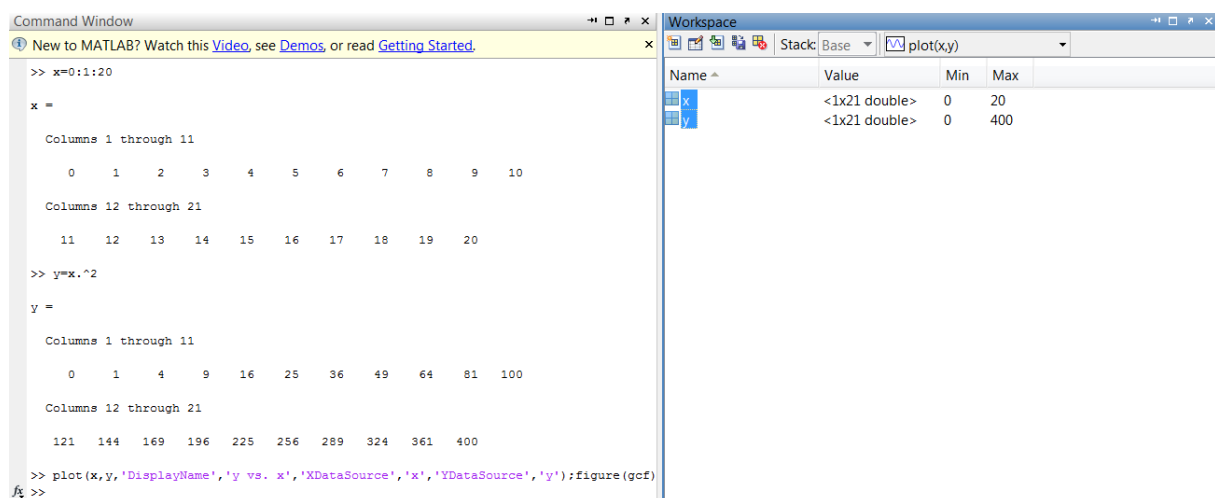


Abbildung 3.11 Plotten y über x

Abbildung 3.12 zeigt die erstellte Grafik *Figure 1*.

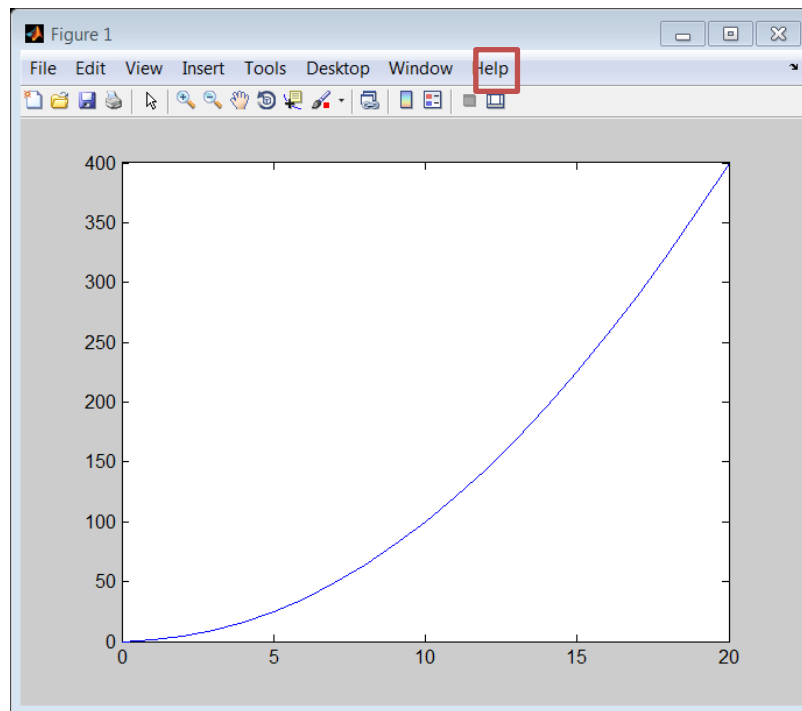


Abbildung 3.12 Figure 1

Die Veränderung von Grafiken, z.B. Achsenbeschriftungen und Titel hinzufügen, ist sowohl über die Menüs des *Figures*, als auch über Befehle im *Command Window* bzw. aus einem *M-Skript* möglich. Durch das Icon *Show Plot Tools and Dock Figure* (rot markiert in Abbildung 3.12) wird der *Property Editor* und der *Plot Browser* geöffnet, siehe Abbildung 3.13. Durch Klicken auf die Achsen oder den Grafen kann man die Einstellungen der einzelnen Objekte ändern. Enthält das *Figure* mehrere Graphen kann man diese im *Plot Browser* einzeln verstecken (Hide) oder anzeigen (Show).

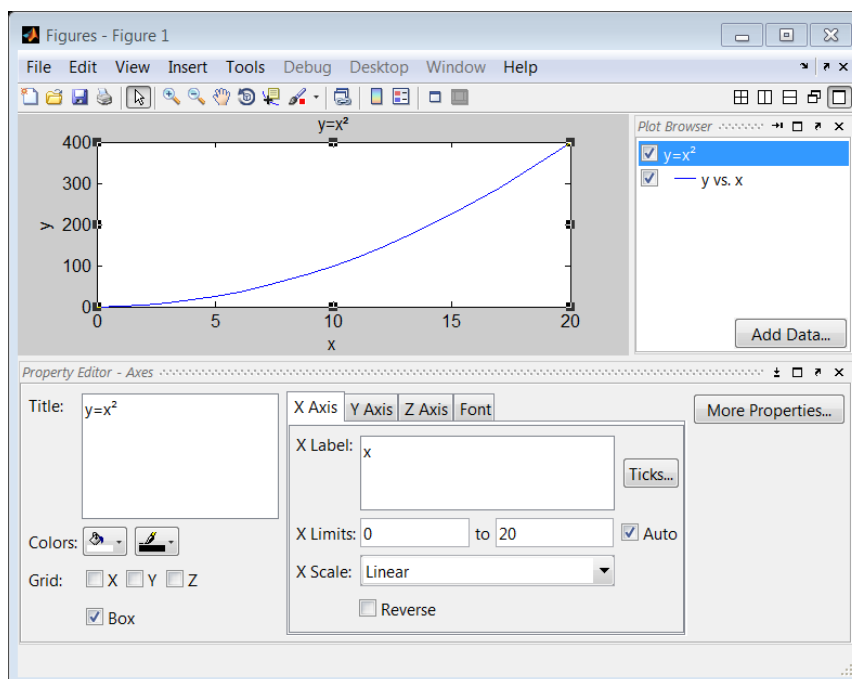
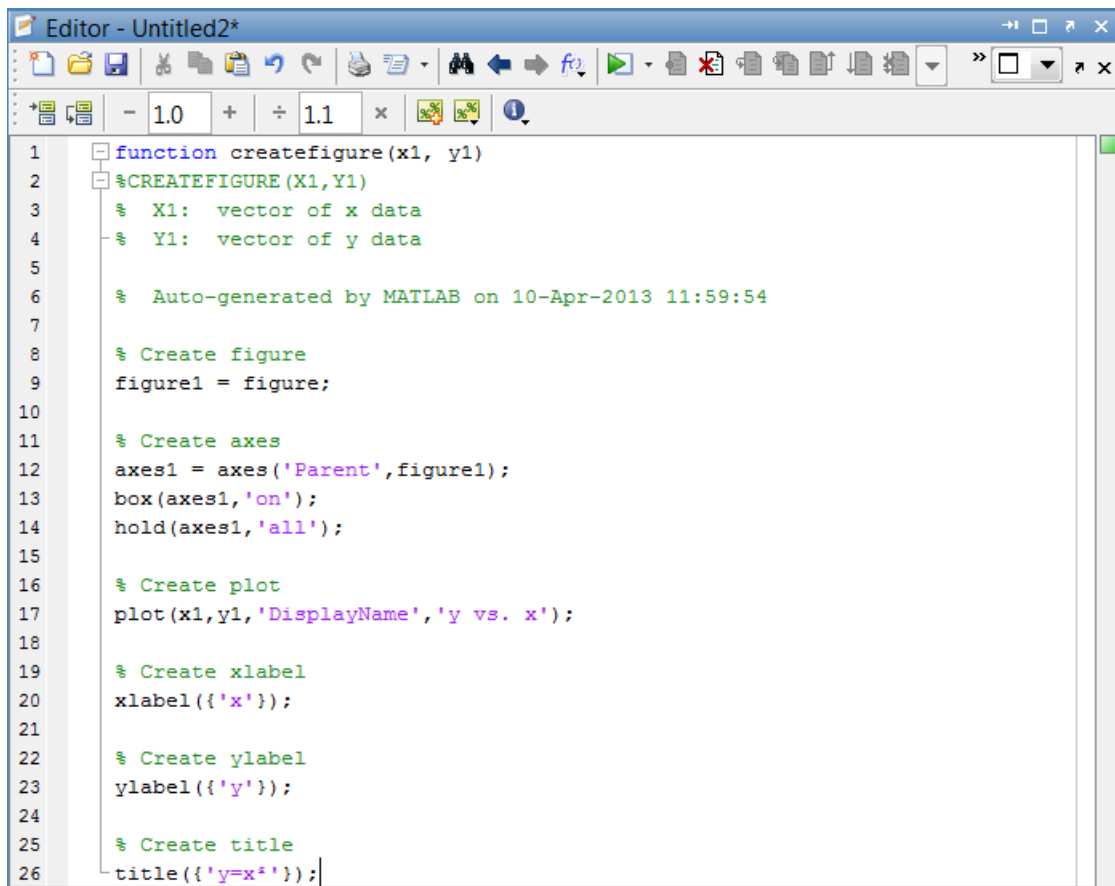


Abbildung 3.13 Figure 1 mit Achsenbeschriftungen, Property Editor und Plot Browser

In Abbildung 3.13 wurden manuell Achsenbeschriftungen und Titel hinzugefügt. Durch *File>Generate Code* wird eine M-Funktion erstellt, die alle manuell über die Menüs ausgeführten Befehle enthält, siehe Abbildung 3.14. Diese kann man z.B. verwenden und erweitern, wenn öfter die gleiche Grafik auf Basis neuer Daten erstellt werden muss.



```
1 function createfigure(x1, y1)
2 %CREATEFIGURE(X1,Y1)
3 % X1: vector of x data
4 % Y1: vector of y data
5
6 % Auto-generated by MATLAB on 10-Apr-2013 11:59:54
7
8 % Create figure
9 figure1 = figure;
10
11 % Create axes
12 axes1 = axes('Parent',figure1);
13 box(axes1,'on');
14 hold(axes1,'all');
15
16 % Create plot
17 plot(x1,y1,'DisplayName','y vs. x');
18
19 % Create xlabel
20 xlabel({'x'});
21
22 % Create ylabel
23 ylabel({'y'});
24
25 % Create title
26 title({'y=x^2'});
```

Abbildung 3.14 M-Funktion createfigure

Zur Erstellung einer Grafik mit mehreren Plots über- oder nebeneinander wird der Befehl `subplot(m,n,p)` verwendet, siehe Abbildung 3.15. Es wird ein  $m \times n$  Raster erstellt und der nächste Plot wird an die Position  $p$  geplottet.

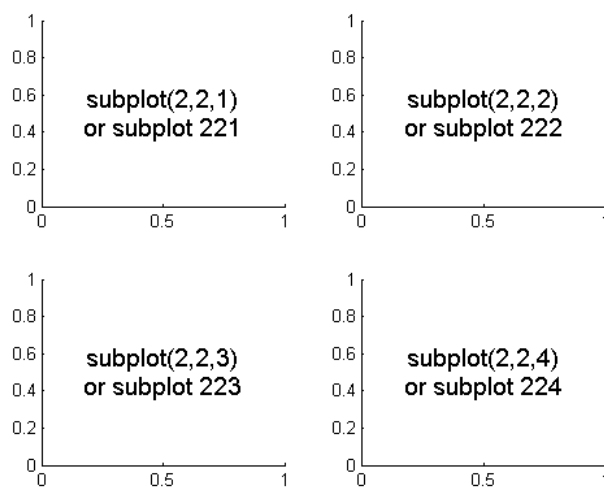


Abbildung 3.15 Subplots (The Mathworks, Inc., 2012)



Weitere Befehle für Grafiken:

<i>figure</i>	erstellt eine neue Grafik
<i>hold on</i>	erstellt den nächsten Plot mit einer anderen Farbe in der letzten Grafik
<i>hold all</i>	erstellt den nächsten Plot mit der gleichen Farbe in der letzten Grafik
<i>hold off</i>	setzt den Befehl <i>hold on</i> zurück
<i>close all</i>	schließt alle Grafiken

## 4 Simulink

Simulink ist eine MATLAB-Toolbox und dient der grafischen Modellierung und Simulation physikalischer Systeme. Simulink stellt eine umfangreiche Bibliothek mit vorgefertigten Blöcken für lineare, nichtlineare, diskrete und hybride Systeme zur Verfügung (Angermann, 2009 S. 273). Simulink kann auf den Matlab *Workspace* zugreifen, Variablen lesen und schreiben.

### 4.1 Simulink Library Browser

Der Button Simulink  öffnet den *Simulink Library Browser*, siehe Abbildung 4.1. Die Simulink Library enthält unterschiedliche Blöcke, z.B. mathematische Operationen (Math Operations), Signalquellen (Sources), Anzeige- und Aufzeichnungsblöcke (Sinks) und vieles mehr. Der Simulink Browser enthält auch die Bibliotheken anderen Toolboxes, wie z.B. CARNOT oder Stateflow. Ein neues Simulink-Modell wird durch das Icon *New Model*  erstellt.

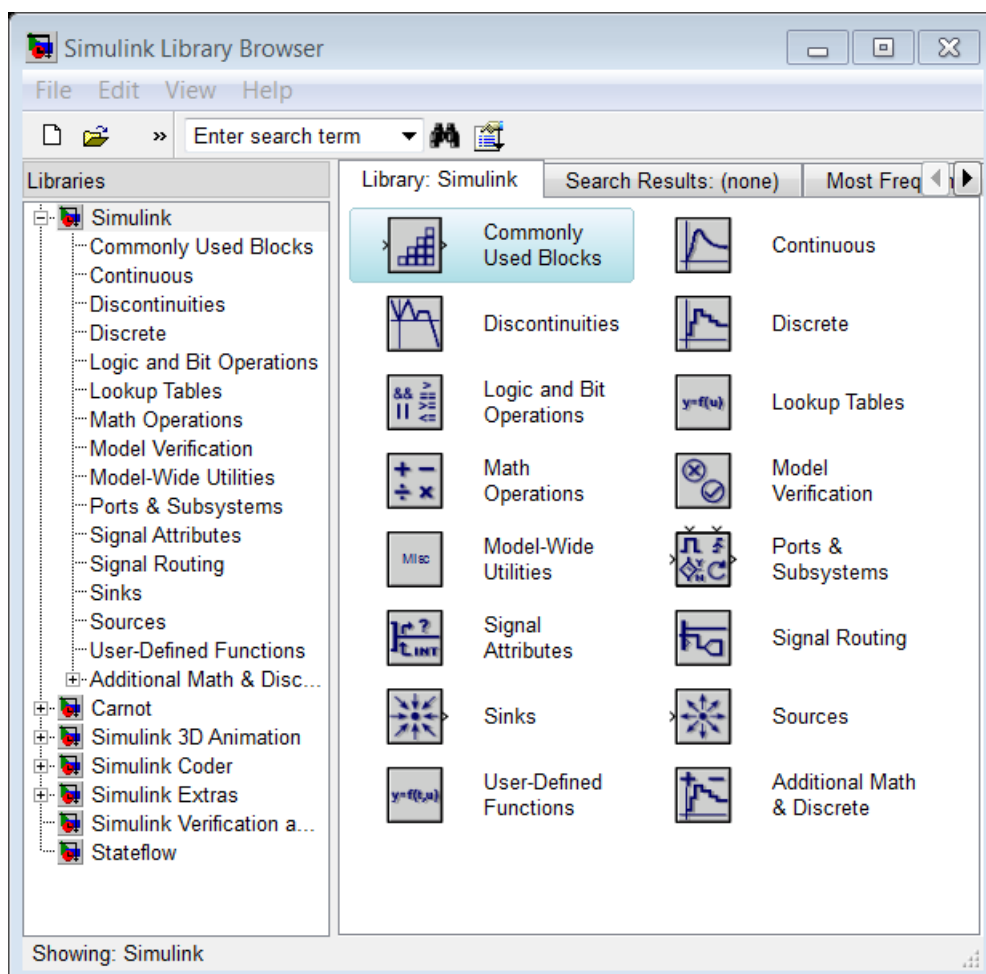


Abbildung 4.1 Simulink Library Browser

## 4.2 Programmieren unter Simulink

Ein Simulink-Modell wird durch Kombination mehrerer Blöcke aus der Bibliothek aufgebaut. Blöcke können per drag & drop bzw. copy & paste in das Modellfenster eingefügt werden. Sie werden durch Signalleitungen verbunden, indem man mit dem Mauszeiger über den kleinen Pfeil rechts am Quellblock geht. Wenn der Kreuzmauszeiger erscheint, kann man eine Verbindung zum Eingang (kleiner Pfeil links) des nächsten Blockes ziehen. Endet eine Verbindung im Leeren wird sie rot gestrichelt dargestellt, siehe Abbildung 4.2.

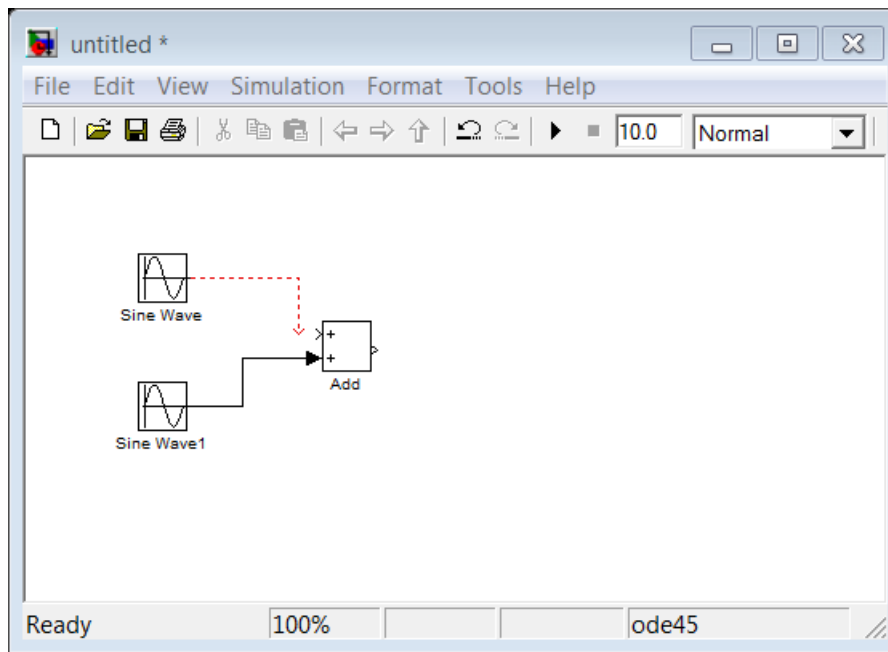


Abbildung 4.2 Verbindung von Blöcken

Abzweigungen von bestehenden Signalen kann man mit der rechten Maustaste von der Verbindung aus ziehen. Signale können durch Doppelklick auf die Linie beschriftet werden.

Abbildung 4.3 zeigt das Simulink-Modell „sinusaddition.mdl“ mit zwei Quellblöcken *Sine Wave*, einer Addition *Add*, einem *Mux*-Block und einem *Scope*. Die *Sine Wave* Blöcke geben zwei unterschiedliche sinusförmige Signale aus. Die Parameter der Blöcke kann man ändern, indem man doppelt auf den jeweiligen Block klickt, die Maske *Block Parameters* öffnet sich, siehe Abbildung 4.4. Die beiden Sinussignale werden addiert. Der *Mux*-Block fasst die drei Signale zu einem Vektor zusammen, sodass sie auf einer Achse im *Scope* (siehe Abbildung 4.5) angezeigt werden können.



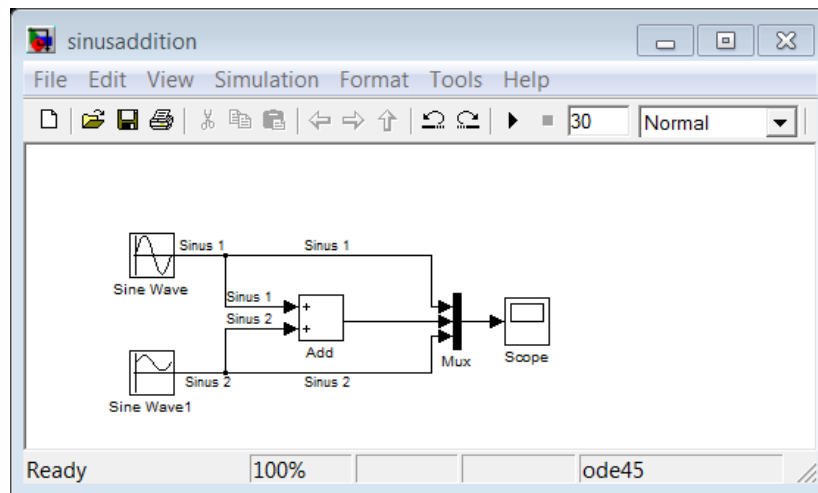


Abbildung 4.3 Simulink-Modell Sinusaddition

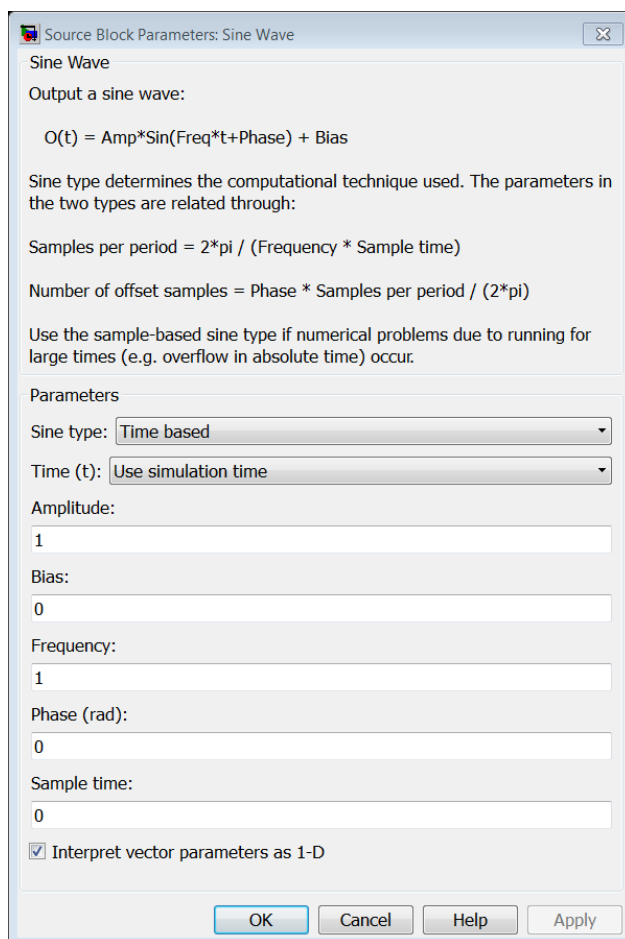


Abbildung 4.4 Source Block Parameters: Sine Wave

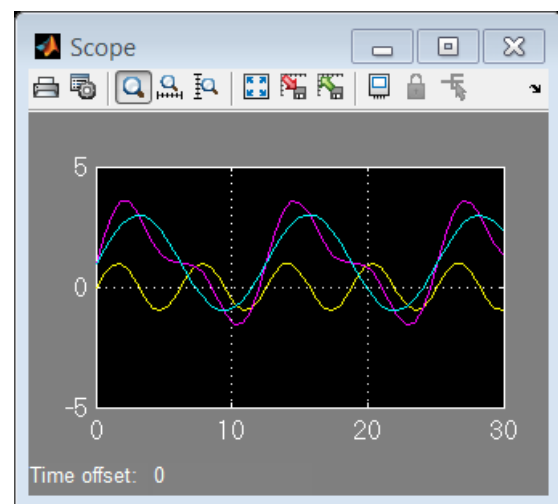



Abbildung 4.5 Scope

Die Eigenschaften des Scopes können über  verändert werden, z.B. eine Legende anzeigen, die Farben der Grafen ändern. Die Skalierung der y-Achse wird durch Rechtsklick auf das Scope > Axes Properties... verändert. Dort kann auch ein Titel definiert werden.

### 4.2.1 Daten ansehen und Speichern

Zur Ansicht und Ablage von Simulationsdaten stehen verschiedene Blöcke zur Verfügung.

Neben dem *Scope* (Abbildung 4.5) gibt es das *Display*, das die Momentanwerte eines Vektors anzeigt, siehe Abbildung 4.6.

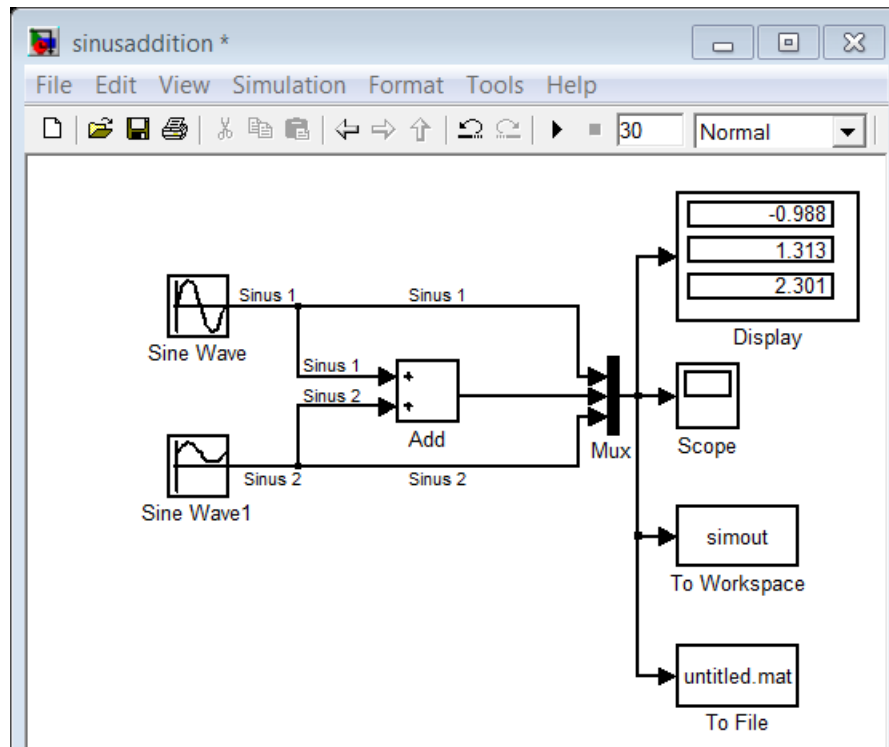


Abbildung 4.6 Simulink-Modell Sinusaddition mit Display und Speicherblöcken

Zum Abspeichern der Daten kann zum einen direkt das *Scope* verwendet werden. Dazu wird die Funktion *Save data to workspace* in den *Scope parameters* unter *History* aktiviert, siehe Abbildung 4.7.

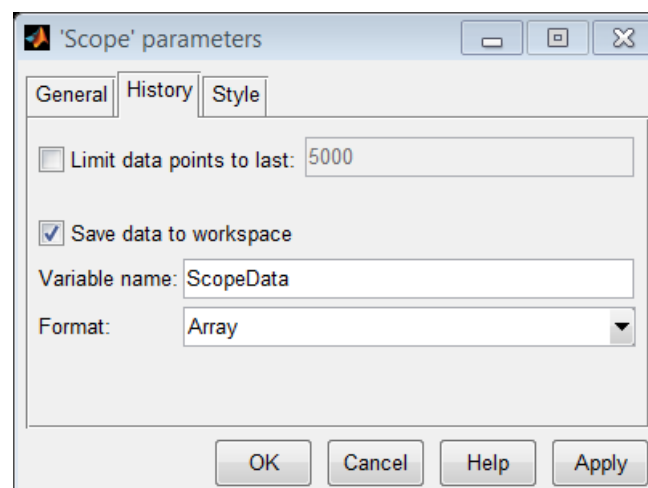
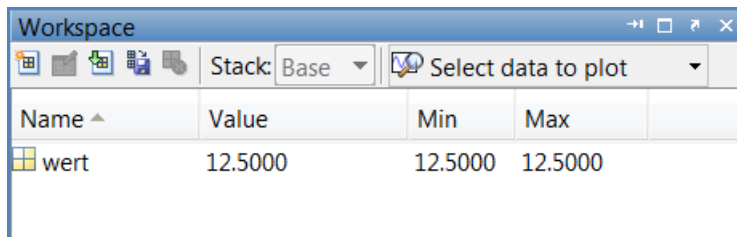


Abbildung 4.7 Scope parameters

Die Blöcke *To Workspace* und *To File* schreiben die Daten direkt in den Workspace bzw. in eine Datei. Als Format ist in den meisten Anwendungsfällen ein Array zu wählen.

### 4.2.2 Verwenden von Variablen und Rechnen in Masken

Simulink-Blöcke können auf Variablen im Matlab-Workspace zugreifen. In einen Constant-Block, siehe Abbildung 4.9 kann man z.B. den Namen einer im Workspace liegenden Variablen (hier „wert“, siehe Abbildung 4.8 und Abbildung 4.10) anstatt einer Zahl eintragen.



Name	Value	Min	Max
wert	12.5000	12.5000	12.5000

Abbildung 4.8 Workspace mit variable wert

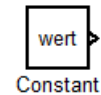


Abbildung 4.9  
Constant-Block

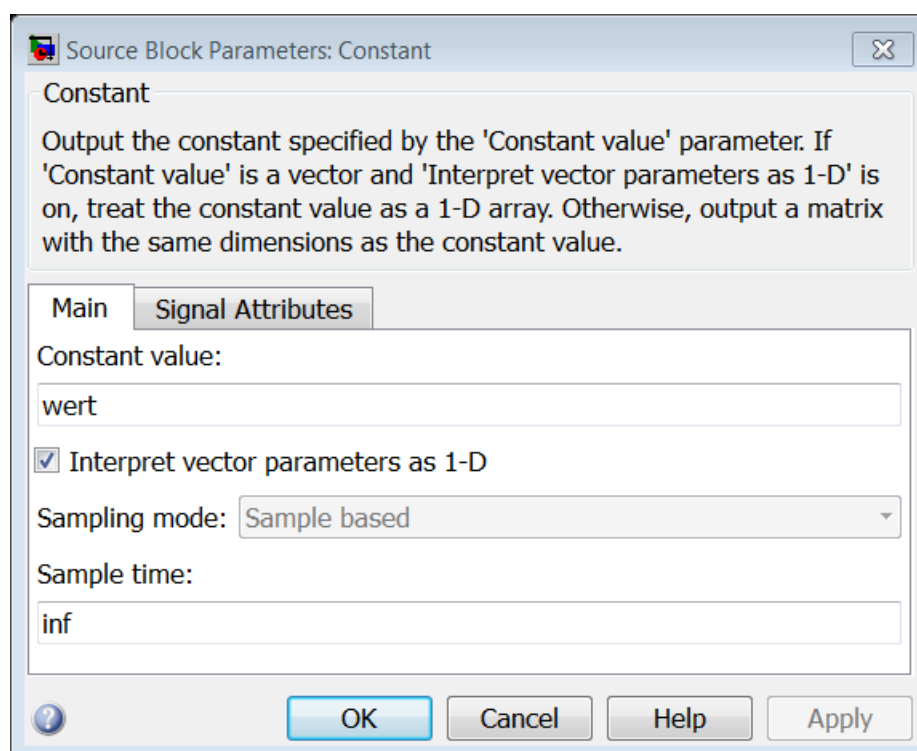



Abbildung 4.10 Source Block Parameters: Constant mit Variablenname wert

Simulink unterstützt auch die Durchführung von Rechnungen innerhalb der Masken, so kann man z.B. die Dauer eines Tages in Sekunden anstatt als 86400 s auch als  $24 \cdot 3600$  s eingeben.

### 4.2.3 Subsysteme

Bei großen, komplexen Modellen kann es nötig sein, mehrere Blöcke zu einem sogenannten Subsystem zusammenzufassen. Subsysteme findet man zum einen in der Simulink Library oder man erstellt sie, indem man die zusammenzufassenden Blöcke markiert und per Kontextmenu *Create Subsystem* ausführt.

Die untere Ebene eines Subsystems wird durch Doppelklicken geöffnet. Mit dem Icon  oder der Esc-Taste schließt man das Subsystem wieder und öffnet die obere Modellebene.

Ein Subsystem kann, wie die Simulink-Blöcke, maskiert werden. Dazu dient der Kontextmenüeintrag *Create Mask*. Bei maskierten Subsystemen öffnet sich bei Doppelklick die Maske.

Um das Subsystem zu öffnen muss der Kontextmenüeintrag *Look Under Mask* ausgeführt werden.

Eine Subsystem hat meist mindestens einen Ein- und Ausgang, siehe Abbildung 4.11. Diese können auch umbenannt werden. Dazwischen kann nach Bedarf programmiert werden.

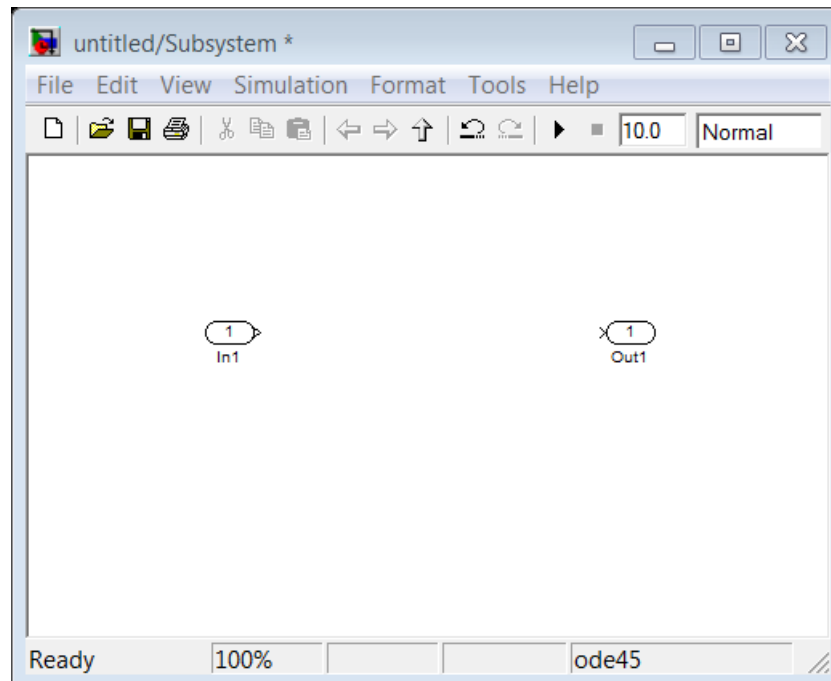


Abbildung 4.11 Ein- und Ausgang Subsystem

### 4.3 Modelle editieren

Zum Editieren von Modellen stehen folgende Möglichkeiten zur Verfügung:

Durch einen Doppelklick auf einen Block öffnet sich die **Block Parameters Dialogbox**, in der Blockparameter gesetzt werden können. Gibt es diese Dialogbox (Maske) nicht, öffnet sich durch einen Doppelklick das Subsystem. Eine Maske erstellt oder editiert man unter *Create Mask ...* bzw. *View Mask...* des Kontextmenüs. Gegebenenfalls muss vorher die Verknüpfung zur Bibliothek unterbrochen werden. Dies geschieht unter *Link Options/Disable Link* des Kontextmenüs.

**Block Properties Dialogbox:** Unter *Edit/Block Properties* im Simulationsfenster oder unter *Block Properties* im Kontextmenü öffnet sich die BLOCK PROPERTIES Dialogbox, in der z.B. Angaben zur Beschreibung des Blocks gemacht werden können.

**Erstellen von Verbindungsleitungen:** Mit gedrückter linker Maustaste einen Ein- und Ausgang verbinden. Eine Verzweigung kann erstellt werden, indem man mit der rechten Maustaste auf eine bestehende Linie klickt.

**Formatieren eines Blocks:** Unter *Format* des Kontextmenüs der rechten Maustaste können Einstellungen wie z.B. Schriftart und -größe, Drehen des Blocks, etc. vorgenommen werden.

**Kopieren/Einfügen:** Mit dem Kontextmenü der rechten Maustaste bzw. *Ctrl.+C* und *Ctrl.+V* können Blöcke / Leitungen kopiert und eingefügt werden.

**Löschen:** Markierte Blöcke/Leitungen können mit dem Kontextmenü der rechten Maustaste oder der *Entf*-Taste gelöscht werden. Unter *Edit/Undo* bzw. *Ctrl.+Z* können einzelne Bearbeitungsschritte rückgängig gemacht werden.

**Markieren:** Mit der linken Maustaste können Blöcke durch Anklicken markiert werden, mehrere mit gedrückter *Strg*-Taste oder durch Auswahlrahmen.

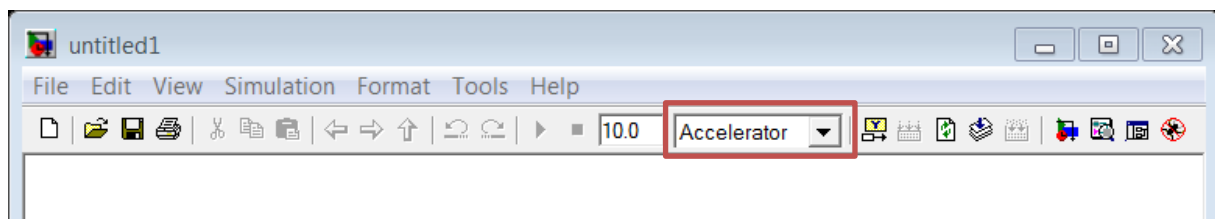
**Subsysteme:** Zusammenhängende Blöcke / Leitungen können markiert und unter dem Menüpunkt *Edit/Create Subsystem* zu einem Subsystem (Modul) zusammengefasst werden.

**Vergrößern/Verkleinern:** Markierte Blöcke können durch Ziehen an einer Ecke in ihrer Größe verändert werden.

**Verschieben:** Mit linker Maustaste können Blöcke/Leitungen innerhalb eines Fensters positioniert werden.

#### 4.4 Durchführen einer Simulation

Bevor die Simulation gestartet werden kann, müssen die dafür notwendigen Simulationsparameter angegeben werden. Zunächst sollte der Accelerator-Mode aktiviert werden, da dieser die Rechenzeit für die Simulation erheblich verkürzen kann. Dies geschieht im Auswahlfenster der Menüleiste.



Über *Simulation > Configuration Parameters* oder Strg-E gelangt man zu den Simulations-Einstellungen, siehe Abbildung 4.12.

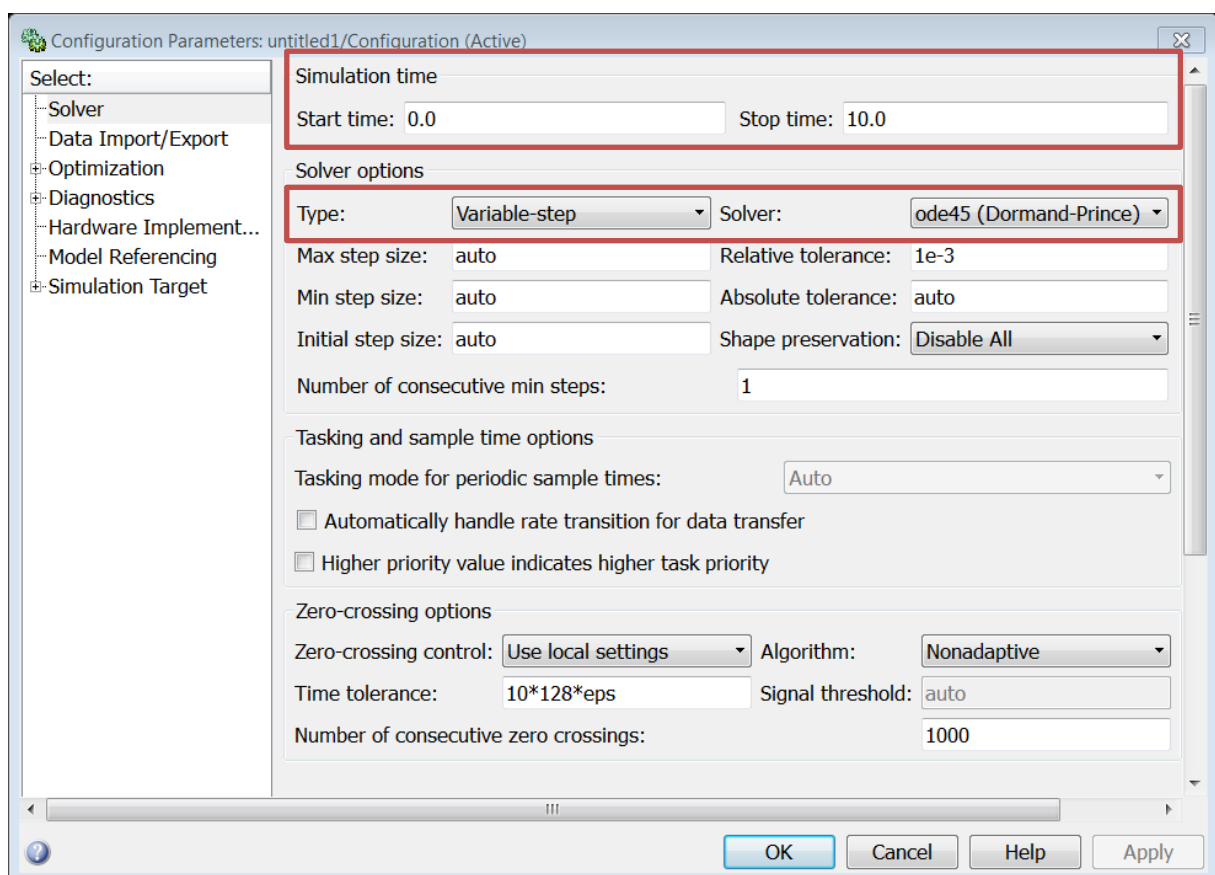


Abbildung 4.12 Simulink Configuration Parameters

Unter *Simulation time* wird der Simulations-Zeitraum eingestellt. Die *Start* bzw. *Stop time* werden in Sekunden angegeben. Soll ein ganzes Jahr simuliert werden, entspricht der 1. Januar 0:00 Uhr der Start-Zeit 0, der 31. Dezember 24:00 Uhr der End-Zeit  $365 \times 24 \times 3600$ .

Unter *Solver options* wird der *Typ Variable-step* gewählt, d.h. SIMULINK passt die Größe der zeitlichen Simulationsschritte während der Simulation selbst an. Die anderen Parameter sollten unverändert auf den Standardeinstellungen verbleiben.

Anschließend kann die Simulation durch einen Klick auf den *Start simulation* – Button  in der Werkzeugleiste gestartet werden.

## 5 MATLAB- und Simulink-Shortcuts

Tabelle 5.1 MATLAB-/ Simulink-Shortcuts

Shortcut	Befehl
<b>Ctrl+C</b>	Kopieren, Abbruch der aktiven Matlab-Aktion, z.B. einer Simulation
<b>Ctrl+X</b>	Ausschneiden
<b>Ctrl+V</b>	Einfügen
<b>Ctrl+N</b>	Neues Modell (im Arbeitsfenster) Neue Variable (im MATLAB-Fenster)
<b>Ctrl+E</b>	Simulation Parameters
<b>Ctrl+T</b>	Start Simulation
<b>Ctrl+D</b>	Update Diagramm
<b>Ctrl+F</b>	Find...
<b>Ctrl+A</b>	Select all
<b>Ctrl+S</b>	Save...
<b>Ctrl+O</b>	Open...
<b>Ctrl+W</b>	Close
<b>Ctrl+Q</b>	Quit
<b>Ctrl+P</b>	Print...
<b>Ctrl+I</b>	Flip Block
<b>Ctrl+R</b>	Rotate Block
<b>Esc</b>	Close Subsystem
<b>F5</b>	Run M-File (im M-File)
<b>F1</b>	Hilfe

## 6 Stateflow

„Stateflow ist eine grafische Erweiterung von Simulink zur Modellierung und Simulation endlicher Zustandsautomaten (Finite State Machines)“ (Angermann, 2009). Regelungsabläufe lassen sich mit Stateflow einfach und intuitiv abbilden.

Die Stateflow-Library *sflib* wird durch den Befehl *sf* geöffnet, siehe Abbildung 6.1. Die Stateflow-Library ist auch über den Simulink Library Browser zugänglich.

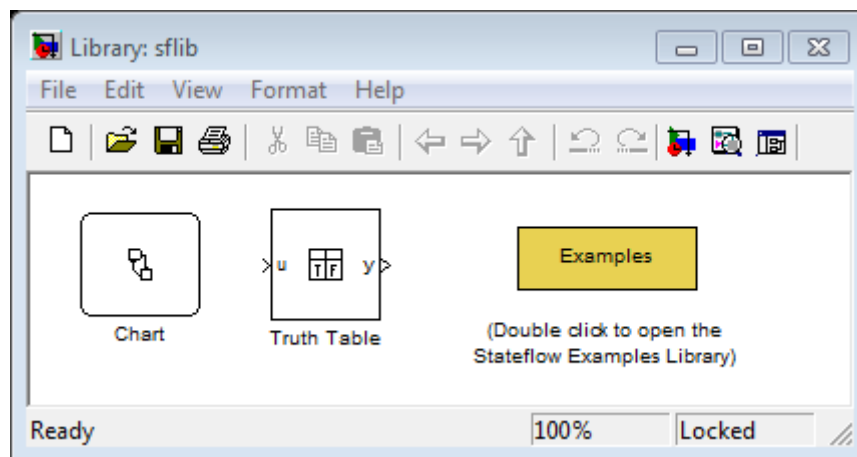


Abbildung 6.1 Stateflow-Library

Es gibt zwei Stateflow-Blöcke, *Chart* und *Truth Table*. Ein Stateflow-Block kann wie ein Simulink-Block in ein Modell eingefügt werden. Abbildung 6.2 zeigt das Modell *Beispiel\_Stateflow.mdl* mit einem Stateflow-Chart.

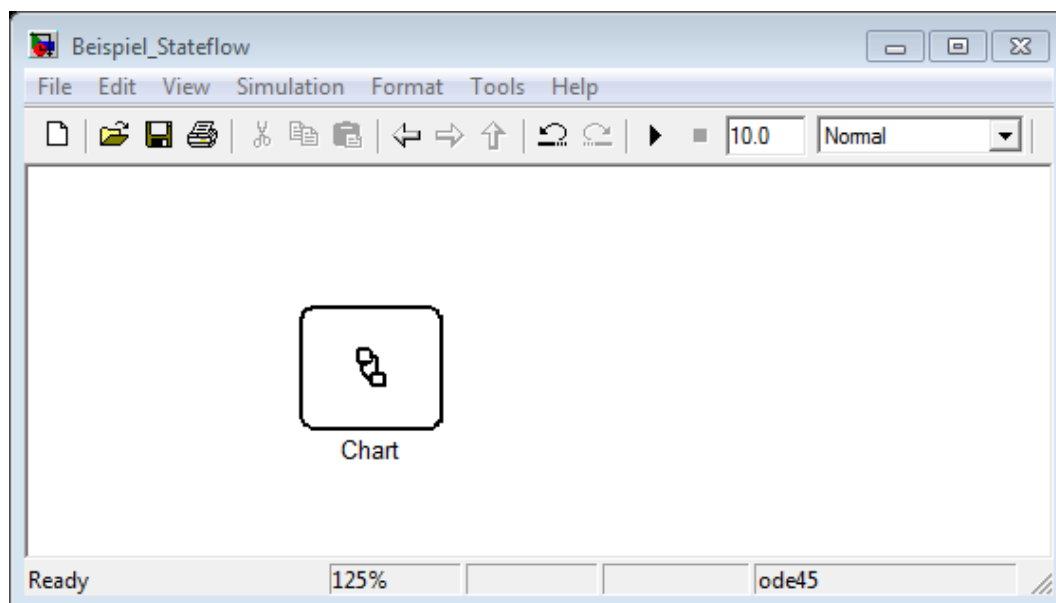


Abbildung 6.2 Stateflow-Chart

Durch Doppelklicken auf das Chart öffnet sich der Stateflow-Editor, siehe Abbildung 6.3. Die linke Werkzeugleiste des Editors enthält alle Elemente, die man zur grafischen Programmierung benötigt.

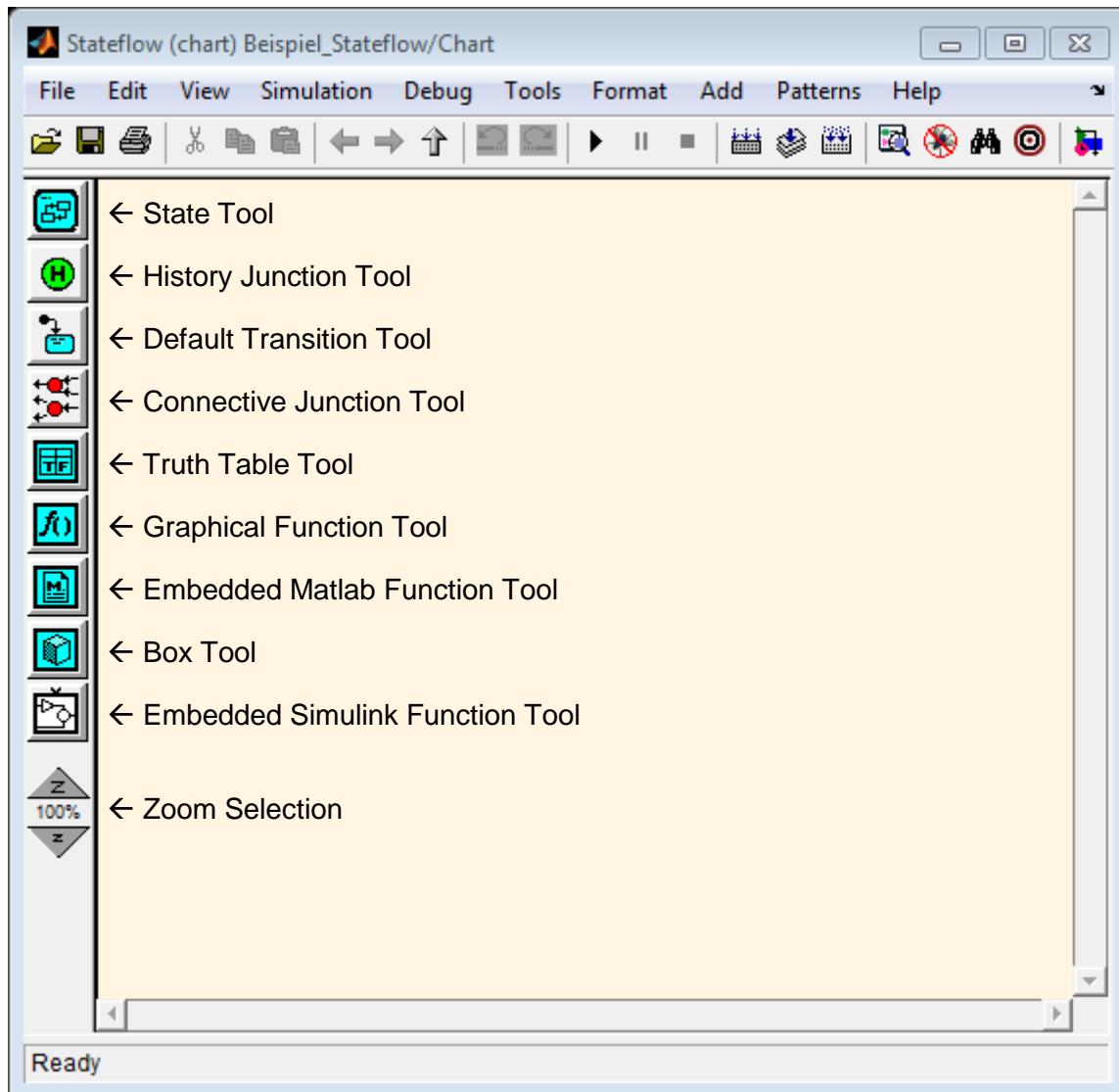


Abbildung 6.3 Stateflow-Editor



## 6.1 Grundelemente eines Charts

Ein Chart besteht aus mindestens zwei Zuständen (States), die durch Verbindungen (Transition) mit einander verbunden sind.

### 6.1.1 Zustand (State)

Abbildung 6.4 zeigt zwei Zustände (Zustand1 und Zustand2), die verbunden sind. Es kann nur einer der beiden Zustände aktiv sein, d.h. es sind exklusive Zustände.

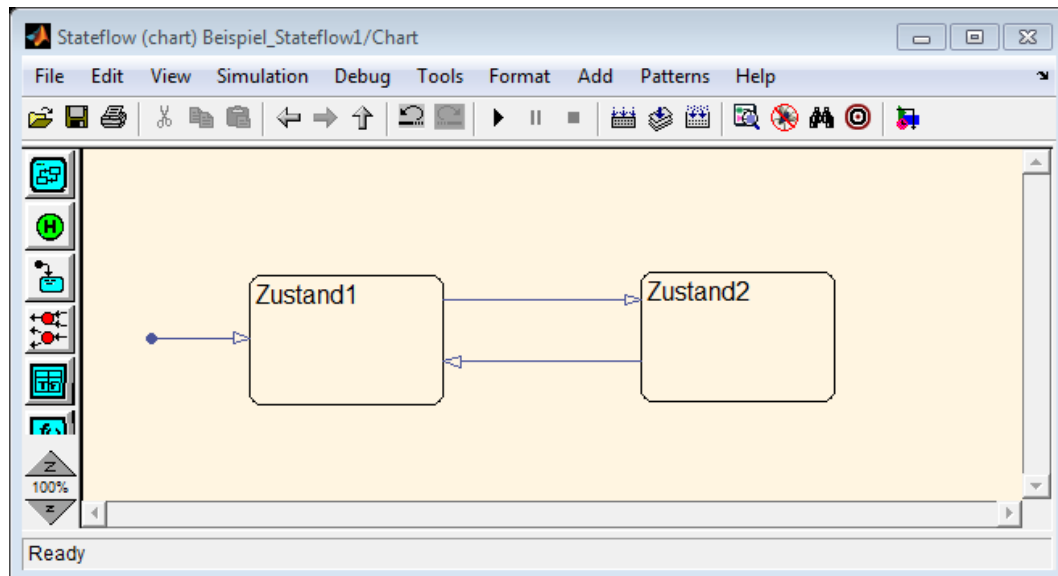


Abbildung 6.4 Exklusive Zustände

Abbildung 6.5 zeigt zwei Zustände (Zustand1 und Zustand2), die gleichzeitig aktiv sein können (Parallel-Zustände). Dies wird durch den gestrichelten Rahmen gekennzeichnet.

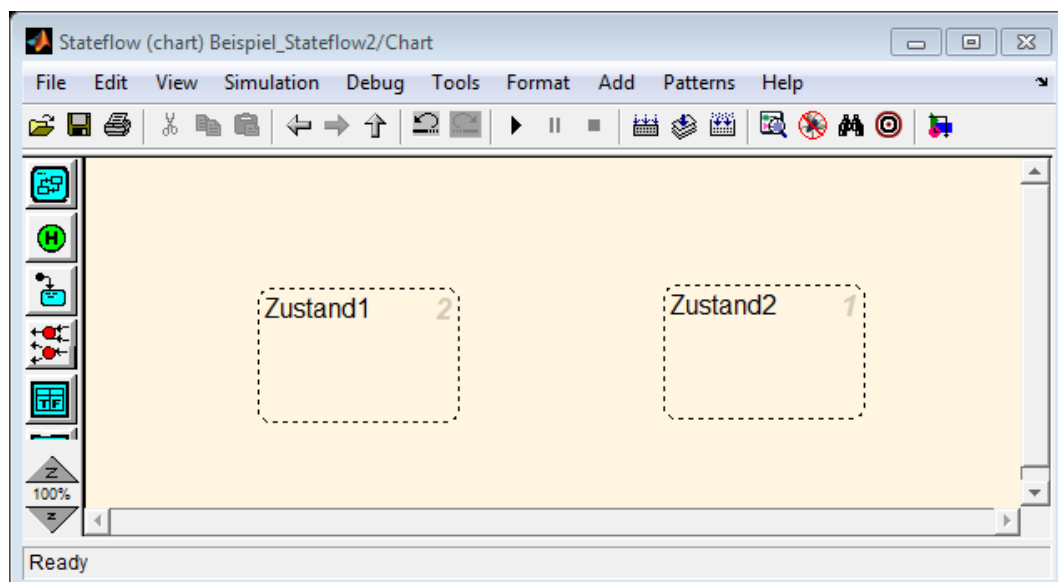


Abbildung 6.5 Parallel-Zustände

Abbildung 6.6 zeigt zwei sogenannte Superstates (Zustand1 und Zustand2). Ein Superstate beinhaltet Zustände und ermöglicht so die Strukturierung. Superstates können auch wie Subsysteme arbeiten, d.h. sie werden als Subcharted definiert. Der Inhalt öffnet sich durch Doppelklicken, siehe Abbildung 6.7. So kann man komplexe Regler strukturiert und übersichtlich aufbauen.

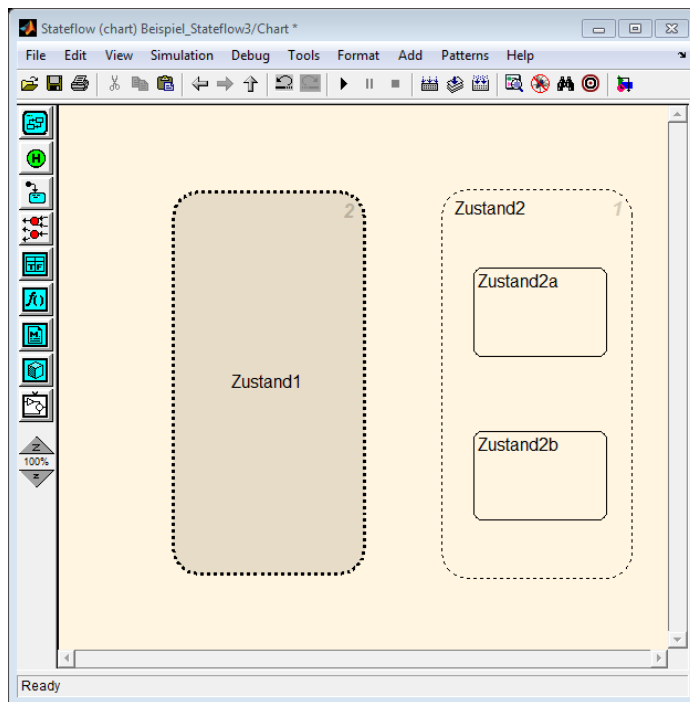


Abbildung 6.6 Subchart

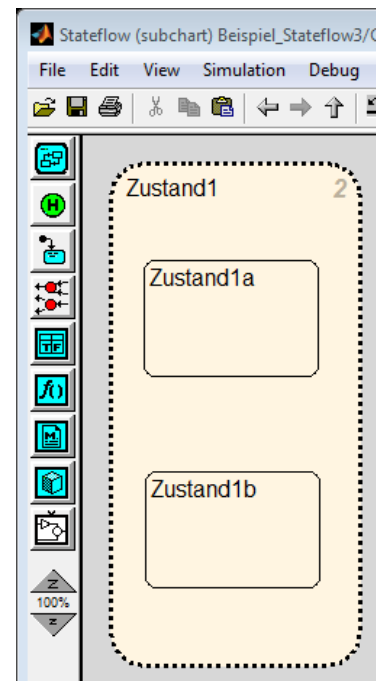


Abbildung 6.7 Inhalt eines Subcharts

States können neben der Bezeichnung (z.B. Zustand1) verschiedene Aktionen beinhalten. Diese werden unter anderem durch die Schlüsselwörter entry und during eingeleitet.

- entry: Aktion wird bei Eintritt in den Zustand einmalig ausgeführt. Entry kann als en abgekürzt werden.
- during: Aktion wird während der Aktivität des Zustandes ausgeführt. Wenn das Chart getriggert ist, wird die Aktion getriggert (also zu definierten Zeitpunkten) ausgeführt, sonst mit jedem Abtastschritt des Simulink-Modells.

Abbildung 6.8 zeigt Zustand1 und Zustand2, in denen die Variable x bei Eintritt mit dem Wert 1 bzw. 2 belegt wird.

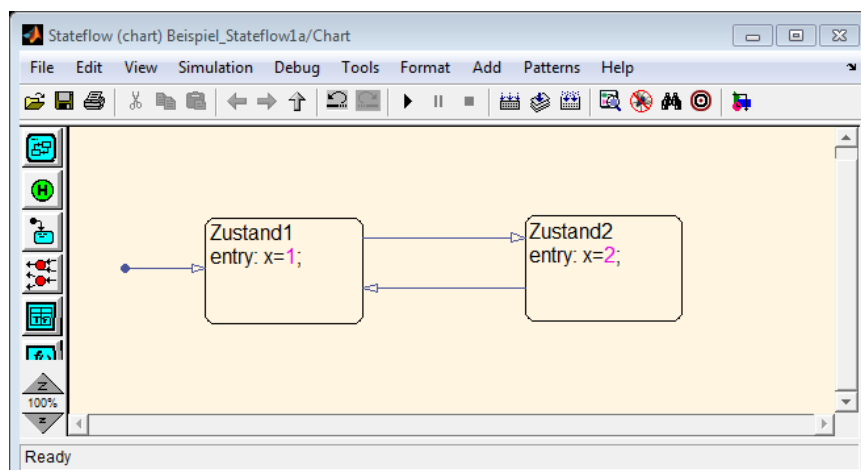


Abbildung 6.8 Aktionen in Zuständen

### 6.1.2 Verbindung (Transition), Bedingungen

Verbindungen werden per Maus von einem Element zu einem anderen gezogen. Ein Chart hat immer eine Default Transition, die als Einstiegspunkt dient. Das State, an das die Default Transition angeschlossen ist, wird als erstes aktiv, in Abbildung 6.8 wäre das Zustand1.

Bedingungen werden in eckigen Klammern per Doppelklick als Label einer Transition hinzugefügt [Bedingung]. Abbildung 6.9 zeigt die Verbindungen der beiden Zustände mit Bedingungen. Zustand2 wird aktiv, wenn die Variable y den Wert 1 annimmt. Zustand1 wird wieder aktiv, wenn y den Wert 2 annimmt.

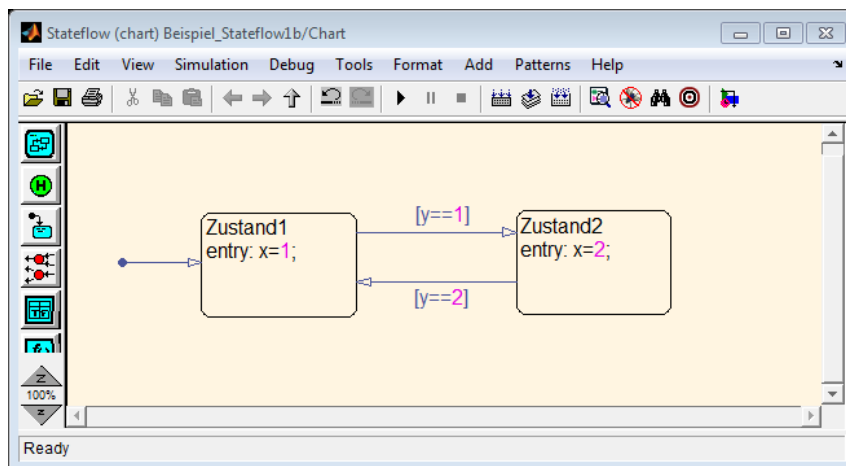


Abbildung 6.9 Bedingungen

Mehrere Bedingungen können mit UND (&&) oder ODER (||) verknüpft werden. Dabei ist zu beachten, dass die einzelnen Bedingungen mit Klammern () versehen werden. In Bedingungen kann auch gerechnet werden, z.B.  $[y \leq a + b]$ , dies ist für Hysterese-Regler praktisch.

Bei mehreren ausgehenden Verbindungen müssen die Bedingungen so definiert sein, dass immer nur eine gültig sein kann.

### 6.1.3 Verbindungspunkte (Connective Junctions) und Zeitverzögerung

In Abbildung 6.10 wurde die Verbindung zwischen Zustand2 und Zustand1 durch einen Verbindungspunkt getrennt. Die Verbindung vom Punkt zu Zustand1 besitzt eine Zeitverzögerung (after(10,secs)), die Zustand1 10 s später aktiviert.

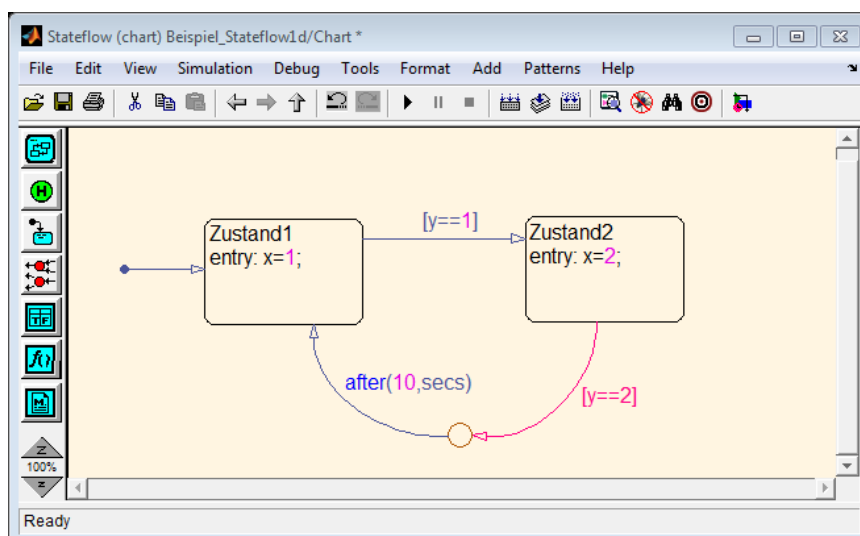


Abbildung 6.10 Chart mit Verbindungspunkt und Zeitverzögerung

## 6.2 Ein- und Ausgänge, Variablen

Zur Definition von Ein- und Ausgängen des Charts und verwendeten Variablen wird der Model Explorer verwendet. In diesem Fall wurde durch den Button Add Data ein Eingang (Input) y und ein Ausgang (Output) x definiert, siehe Abbildung 6.12.

Das Stateflow-Chart hat nun Ein- und Ausgänge, die direkt mit Simulink-Blöcken verbunden werden können, siehe Abbildung 6.11.

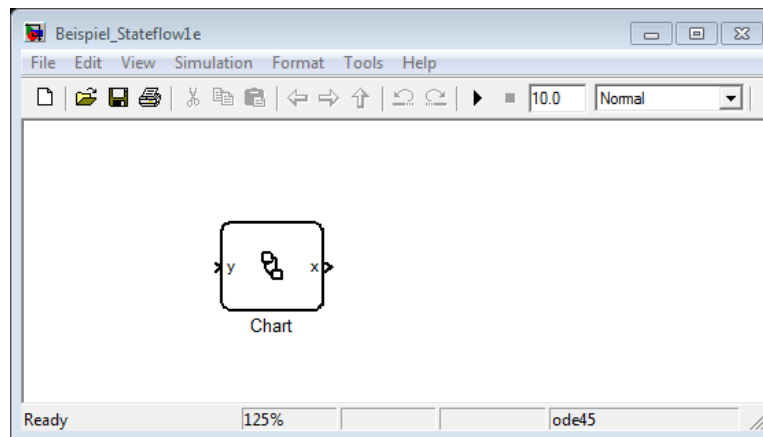


Abbildung 6.11 Stateflow-Chart mit Ein- und Ausgang

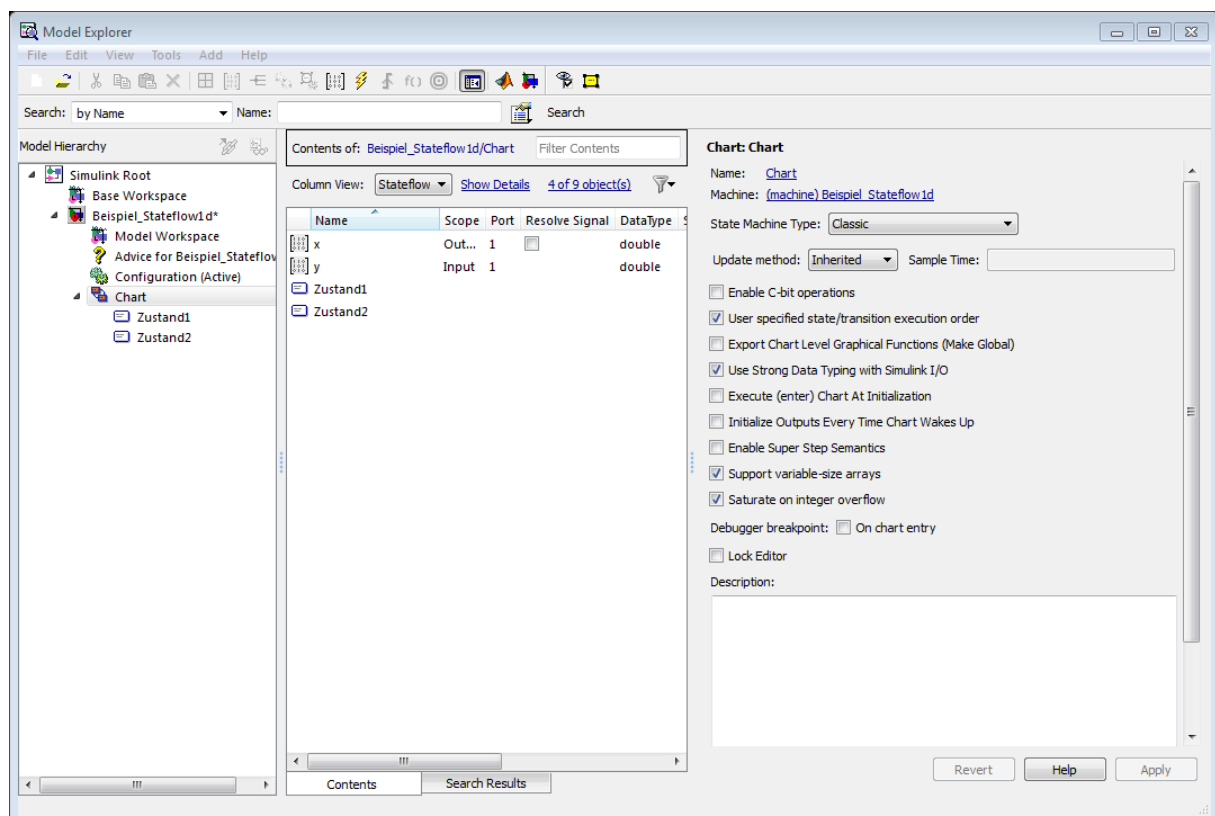


Abbildung 6.12 Model Explorer

Neben Ein- und Ausgängen können noch folgende Datentypen definiert werden:

- Local: Variable, die nur in diesem Chart benutzt werden kann.
- Constant: Konstante, die durch darüber liegende Charts benutzt werden kann.
- Parameter: Konstante, die im Matlab-Workspace definiert ist.

## 7 CARNOT

CARNOT steht für **C**onventional **A**nd **R**enewable **e**nergy systems **O**ptimization **T**oolbox. Diese Anleitung bezieht sich auf die CARNOT-Version 5.3.

CARNOT enthält Blöcke aus dem Bereich der Gebäude- und Heizungstechnik und ermöglicht somit Simulationen dieser Systeme. Unter MATLAB wird CARNOT durch den Befehl *car-not* geöffnet. Abbildung 7.1 zeigt das Library-Fenster. Die CARNOT-Bibliothek enthält einige Demos.

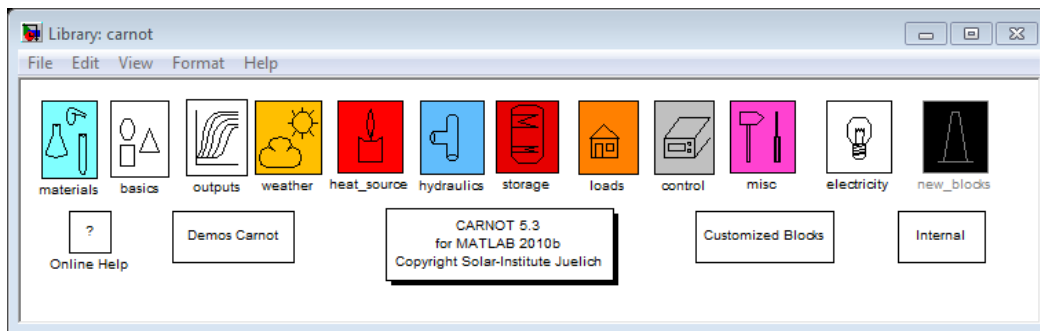


Abbildung 7.1 CARNOT-Library

Der Zugriff auf die CARNOT-Bibliothek ist auch über den *Simulink Library Browser* möglich, siehe Abbildung 7.2, jedoch ohne die Demos.

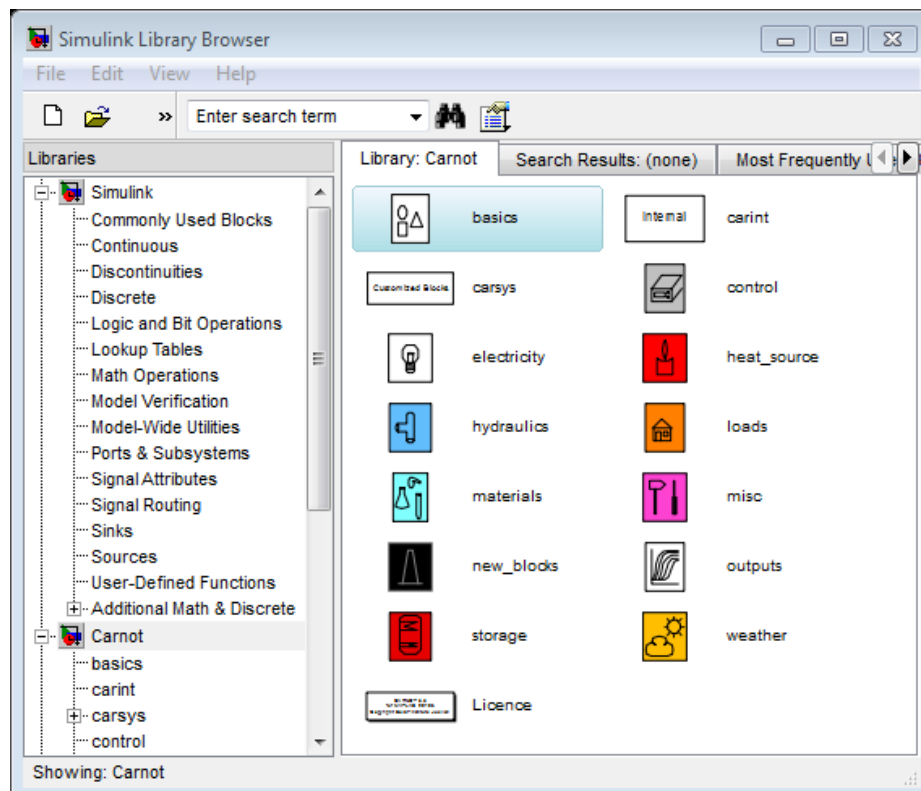


Abbildung 7.2 CARNOT im Simulink Library Browser

Die CARNOT-Bibliothek enthält folgende Register:

- basics: Grundlegende strömungstechnische Berechnungen
- carint: Unternehmensinterne Blöcke
- carsys: mit Herstellerdaten parametrisierte CARNOT-Blöcke
- control: Messstellen und Regelung
- electricity: Photovoltaik
- heat\_source: Wärmequellen, wie Kollektoren, elektrische Heizung, Heizgerät, Wärmeübertrager, Wärmepumpen, Heizkörper
- hydraulics: Rohrleitungen, Pumpen, Ventile
- loads: Lasten, wie Hausmodelle, Brauchwasserzapfung
- materials: Stoffdaten
- misc: alles Mögliche
- new\_blocks: neue Blöcke in der CARNOT-Bibliothek, die noch in der Testphase sind
- outputs: Auswahl- und Schreibe-/Speicher-Blöcke
- storage: Wärmespeicher
- weather: Einlesen und Erstellen von Wetterdaten

## 7.1 CARNOT-Hilfe

Die CARNOT-Hilfe (Hafner, 2012) kann durch Klicken auf Help in der Maske eines CARNOT-Blockes geöffnet werden. Es handelt sich um ein html-Handbuch, welches Grundlegendes zur CARNOT sowie Einzelheiten zu jedem Block erläutert.

## 7.2 CARNOT-Demos

Durch Doppelklicken auf Demos Carnot in der CARNOT-Library, siehe Abbildung 7.1, öffnet sich ein Auswahlfenster für die Demos, siehe Abbildung 7.3.

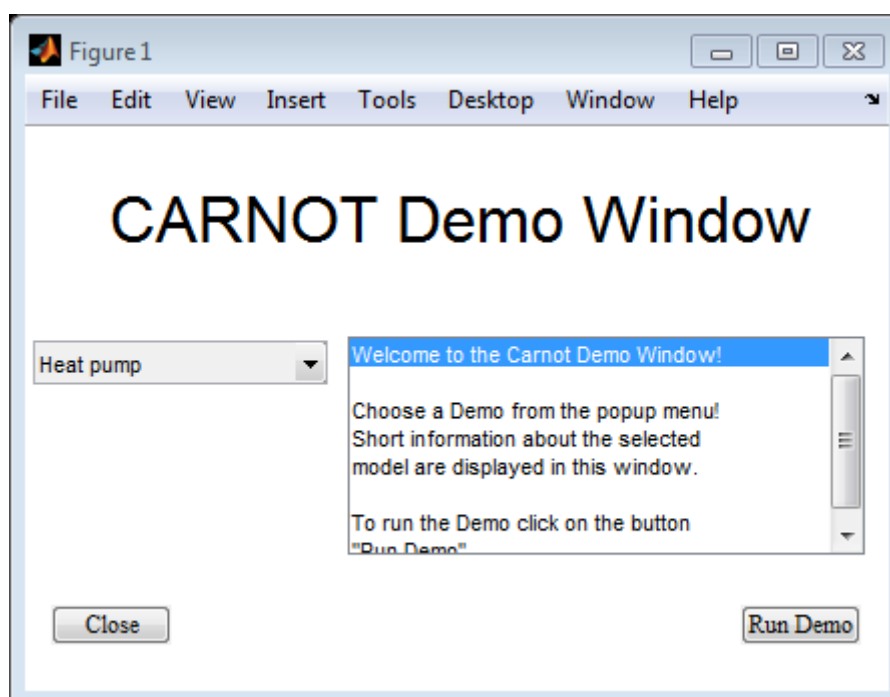


Abbildung 7.3 Auswahl-Fenster CARNOT Demos

Es stehen lauffähige Modelle zu folgenden Systemen zur Verfügung:

Tabelle 7.1 CARNOT Demos

Demo	Beschreibung
Heat exchanger	Wärmeübertrager <i>heat_exchanger</i>
Heat pump	Wärmepumpe <i>heatpump</i>
Furnace	Heizgerät <i>furnace</i>
House heating	Raummodell <i>room_radiator</i> mit Heizgerät und Regelung
House heating, real vs. Ideal	Raummodell <i>simple house</i> , 2 Varianten: ideal, real
Parallel flow diversion	Parallele Rohrleitungen mit <i>flow_diverter</i> und <i>flow_mixer</i>
Symetric flow diversion	3 Parallele Rohrleitungen mit <i>flow_diverter</i> und <i>flow_mixer</i>
Tempering valve	Thermostatischer Mischer <i>thermostatic flow mixer</i>
Thermostatic valve	Thermostatventil <i>thermostatic valve</i>
Ground storage	Erdspeicher <i>ground storage</i>
U-tube ground storage	Erdsonde <i>ground_storage_U-tube</i>
Simple storage	Speicherblock <i>storage</i>
Storage with heat exchanger	Speicherblock <i>storage_multiport</i>
Storage with tank in tank	Speicherblock <i>storage_tank-in-tank</i>
Simple combi-storage	Solarsystem mit Speicherblock <i>storage_tank-in-tank</i>
Combi-storage with hydraulics	Solarsystem mit Speicherblock <i>storage_multiport</i> , Raummodell <i>simple house</i> (Raumheizung) und Brauchwasserzapfung
Solar hot water thermosiphon system	Solares Warmwassersystem mit Thermosyphon
Solar system with control	Solarsystem mit Regelung
Solar system with backup heating	Solarsystem mit Backup
PV generator	Photovoltaik-Block <i>PV_generator</i>
Validation	Modell zur Blockvalidierung

### 7.3 Vektoren

Zusätzlich zu den Simulink-Verbindungen gibt es CARNOT-spezifische Vektoren zur Übertragung bestimmter Daten. Wichtige Vektortypen zur Verbindung von Blöcken der Toolbox CARNOT sind:

- THV (Thermo-Hydraulic-Vector),  
der die aktuellen Zustandsgrößen eines Fluids überträgt. Der THV-Vektor beschreibt einen physikalisch existierenden Massenstrom.
- Weather data Vector,  
der die aktuellen Wetterdaten an einem bestimmten Standort wie z.B. Einstrahlung und Außentemperatur enthält.
- AIV (Air-Infiltration-Ventilation),  
der den Raumzustand beschreibt und z.B. das Modul *room node*, in dem der Raumzustand berechnet wird, mit korrelierenden Modulen (*wall*, *radiator*, *ventliation*, etc.) verbindet.
- S-Vektor,  
der genutzt wird, um Energieflüsse darzustellen.

Die Vektoren haben je nach Typ unterschiedliche Elemente, die in den Blöcken deren Funktion entsprechend verrechnet werden. In folgender Tabelle sind die möglichen Elemente der drei Vektoren aufgeführt.

#### 7.3.1 THV-Vektor

Tabelle 7.2: Elemente des THV-Vektors

Nr.	Beschreibung	Abkürzung	Einheit	Bemerkung
1	flow identifier	ID	none	is set by the simulation, not by the user
2	temperature	T	[°C]	
3	massflow	$m_{\text{dot}}$	[kg/s]	
4	pressure	p	[Pa]	
5	fluid type	Fluid_ID	none	is set in the pump
6	mixture of fluid	Fluid_mix	[0..1]	Is set in the pump
7	diameter last piece	d_last	[m]	
8	constant coefficient of pressure drop	c	None	
9	linear coefficient of pressure drop	l	[s/kg]	
10	quadratic coefficient of pressure drop	q	[s <sup>2</sup> /kg <sup>2</sup> ]	
11	not used			
to	...			
20	not used			



Tabelle 7.3 Beschreibung von Element Nr. 5 des THV-Vektors, fluid type

Fluid_ID	Fluid	Bemerkung
1	water	fluid mix is the vapour content in the 2-phase region
2	air	fluid mix is kgwater/kgair, i.e. the absolute fraction of water
3	cotton oil	
4	silicone oil	
5	water-glycol	fluid mix gives the volume percentage of glycol the properties are for Typhocor L, Typhorop Hamburg
6	Tyfocor LS	fluid mix is the vapour content

### 7.3.2 Weather data vector

Es gibt ein altes und ein neues Format des Wetterdatenvektors. Der alte Vektor enthält nur 17 Spalten; die Werte für die Einstrahlung in Kollektorebene fehlen. Die Funktion *convert\_weather* konvertiert alte Datensätze in das neue Format.

Tabelle 7.4 Elemente des Weather data Vektors

Nr.	Beschreibung	Einheit
1	Time	[s]
2	timevalue (comment line) format YYYYMMDDHH Y is the year, M the month, D the day, H the hour	[-]
3	zenith angle of sun (at time, not averaged) (continue at night to get time of sunrise by linear interpolation)	[°]
4	azimuth angle of sun (0°=south, east negative) (at time, not averaged in timestep)	[°]
5	direct beam solar radiation on a normal surface	[W/m²]
6	diffuse solar radiation on a horizontal surface	[W/m²]
7	ambient temperature	[°]
8	radiation temperature of sky	[°C]
9	relative humidity	[%]
10	Precipitation	[m/s]
11	cloud index (0=no cloud, 1=covered sky)	[-]
12	station pressure	[Pa]
13	mean wind speed	[m/s]
14	wind direction (north=0° west=270°)	[°]
15	incidence angle on surface (0° = vertical) (= -9999, if surface orientation is unknown)	[°]
16	incidence angle in plane of vertical and main surface axis (the main axis is parallel to heat collecting pipes in a collector, it is pointing to the center of the Earth)	[°]
17	incidence angle in plane of vertical and second surface axis (the second axis is in the surface and a vertical on the heat collecting pipes in a collector, it is pointing to the horizon)	[°]
18	direct solar radiation on surface	[W/m²]
19	diffuse solar radiation on surface	[W/m²]

### 7.3.3 AIV-Vektor

Tabelle 7.5 Elemente des AIV-Vektors

Nr.	Beschreibung	Einheit
1	sensitive room temperature	[°C]
2	convective room temperature (air temperature)	[°C]
3	radiative room temperature	[°C]
4	mass of water per mass of air	[kg/kg]
5	mass of CO <sub>2</sub> per mass of air	[kg/kg]
6	density of air in the room	[kg/m <sup>3</sup> ]
7	heat capacity of air in the room	[J/(kg·K)]
8	air pressure in the room	[Pa]
9	air exchange rate (flow rate per room-volume)	[1/h] (no SI-unit)

### 7.3.4 S-Vektor

Tabelle 7.6 Elemente des S-Vektors

Nr.	Beschreibung	Einheit
1	incoming solar power	[W]
2	Power input by equipment, i.e. computers	[W]
3	Power input by light sources	[W]
4	Power input by persons	[W]
5	Power input by heating, i.e. radiator	[W]
6	Power input by ventilation	[W]
7	convective power from walls	[W]
8	radiative power from walls	[W]
9	massflow of air	[kg/s]
10	massflow of water (part of the moist air)	[kg/s]
11	massflow CO <sub>2</sub>	[kg/s]
12	flow rate of air from outside into room	[m <sup>3</sup> /s]
13	flow rate of air from room to outside	[m <sup>3</sup> /s]

## 7.4 CARNOT-Blöcke

CARNOT-Blöcke enthalten Simulink-Blöcke, die geeignet verschaltet wurden, kombiniert mit CARNOT-Blöcken, die z.B. Zustandsgrößen berechnen, und manchmal sogenannte S-Functions. S-Functions sind in C++ geschrieben und bilden das Verhalten thermischer Systeme durch Gleichungssysteme ab.

### 7.4.1 Wetterdaten

Die Bereitstellung von Wetterdaten und Erstellung des Wetterdatenvektors kann durch die Blöcke *weather\_from\_workspace*, *weather\_from\_file* oder *weather\_simple\_model* erfolgen. Für die Blöcke *weather\_from\_workspace* und *weather\_from\_file* stehen die Wetterdaten in einer Datei mit der Endung \*.dat, welche im Arbeitsverzeichnis gespeichert sein muss, zur Verfügung.

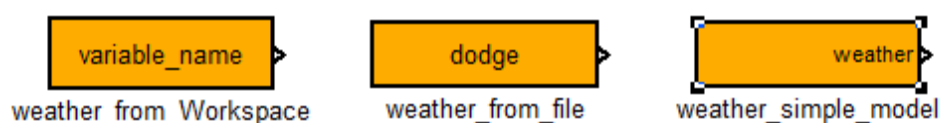


Abbildung 7.4 Weather-Blöcke

Das Laden von Wetterdaten am Standort Essen in der *Workspace* erfolgt durch die Eingabe des Befehls **load essen.dat** in das *Command Window*. Im *Workspace* erscheint anschließend das Element *essen*. Bei den Modulparametern des Moduls *weather\_from\_workspace* muss dann noch unter Data *essen* eingetragen werden.

*weather\_simple\_model* stellt alle Eingabemöglichkeiten für einen einfachen Wetterdatenvektor zur Verfügung, siehe Abbildung 7.5.

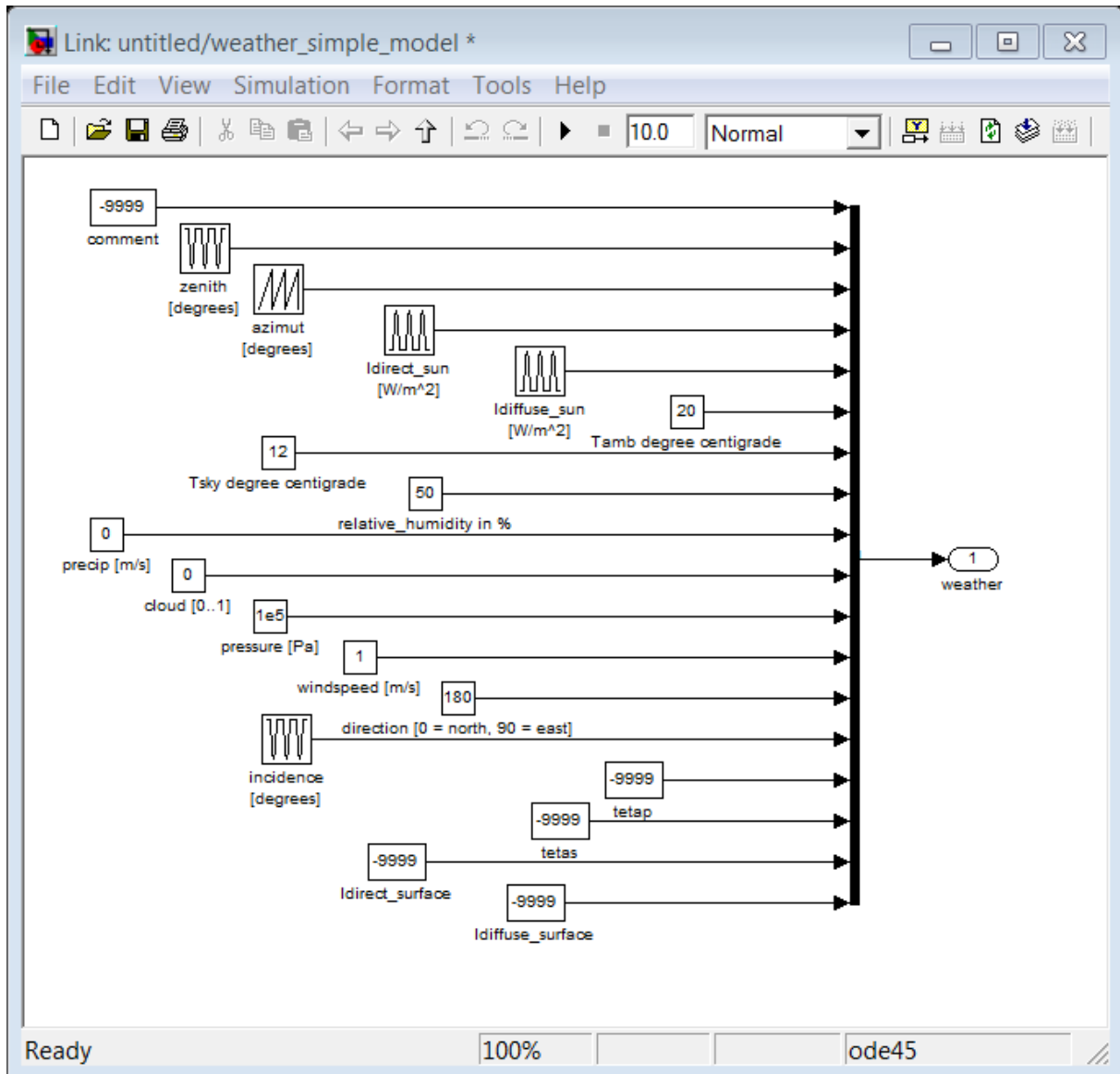


Abbildung 7.5 weather\_simple\_model

### 7.4.2 Flachkollektor

Der Block des Flachkollektors befindet sich in der CARNOT-Bibliothek -> *heat\_source* -> *collector\_flat\_plate*. Der Flachkollektor aus der Toolbox CARNOT besitzt drei Eingänge und einen Ausgang. Über die Anschlüsse *THVin* und *THVout* werden die Werte des Arbeitsmediums (Wasser bzw. Wasser-Glykol-Gemisch) übertragen. Diese Anschlüsse sind üblicherweise mit Modulen wie Speicher, Pumpen etc. verbunden. Über den Eingang *position* wird die Ausrichtung des Kollektors festgelegt. Durch den Eingang *weather* erhält der Kollektor die Wetterdaten.

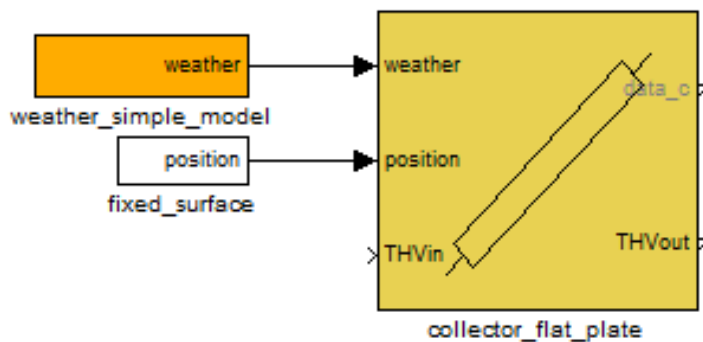


Abbildung 7.6: Block *collector\_flat\_plate* mit den Modulen für Wetterdaten und Position

Außerdem gibt es noch den Block *flat\_plate\_collector\_EN12975*, der nach EN 12975 modelliert wurde und der mit Hilfe einer Kollektorprüfberichtes nach EN 12975 leicht parametrierbar werden kann.

### 7.4.3 Speicher

Es stehen mehrere Speicher zur Auswahl, die nachfolgend erläutert werden.

**Achtung:** In der Eingabemaske der Speicher wird zwischen *number of nodes* (Knoten) und *number of measurement points* (Messpunkten) unterschieden. Die Temperaturen der verschiedenen Schichten des Speichers werden am Anschluss *Tnodes* ausgegeben und entsprechen den Messpunkten. Ein Knoten entspricht einer Zone mit gleichen Zustandsbedingungen. 3 Knoten unterteilen den Speicher beispielsweise in 3 gleichgroße Temperaturbereiche. In der Paramettermaske der Speichermodelle muss jeweils eine Starttemperatur *initial temperature* für alle Knoten oder ein Vektor mit Starttemperaturen für jeden einzelnen Knoten (v. unten n. oben) definiert werden. Es ist bei der Auswahl der Messpunkte, v.a. bei geringerer Messpunktanzahl als Knotenanzahl also darauf zu achten, dass die Messpunkte im richtigen Temperaturbereich liegen. Die Position eines Sensors wird durch die relative Höhe (0..1 rH) definiert, wobei 0 dem Boden und 1 dem Deckel entspricht. Bei einem Speicher mit 10 Knoten haben die Schichten also eine Höhe von 0,1 rH. Ein Temperatursensor in Schicht 5 kann mit den Werten von 0,4..0,49 rH definiert werden. 0,5 rH bezieht sich auf Schicht 6.

### 7.4.3.1 Pufferspeicher

Beim Pufferspeicher *storage* handelt es sich um einen einfachen Wasserbehälter. Dieser wird mittels der Anschlüsse *THVsource* mit einem heißen Wasserstrom verbunden. Über den Eingang *THVload* kann durch den Anschluss eines kalten Wasserstroms die Energieentnahme durch Brauchwasserzapfungen simuliert werden.

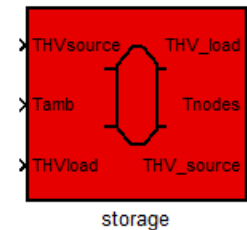


Abbildung 7.7 Block storage

### 7.4.3.2 Multiport-Speicher

Der *Multiport-Speicher* besitzt zwei integrierte Rohr-Wärmeübertrager mit jeweils einem Ein- und einem Ausgang für Wärmequellen. Außerdem verfügt er über zwei direkte Speicheranschlüsse mit je einem Ein- und Ausgang. Über die Anschlüsse *THVsolar* wird der Solarkreis, über *THVbackup* ein Zusatzheizgerät (z.B. Gaskessel) an den Speicher angeschlossen. Am Eingang *Tamb* wird mittels des Moduls *Constant* eine konstante Umgebungstemperatur (in °C) angelegt. Durch die Anschlüsse *THVheating* und *THVother* wird der Speicher mit dem Wärmeabnehmer (Wasserzapfung, Gebäudeheizung, etc) verbunden.

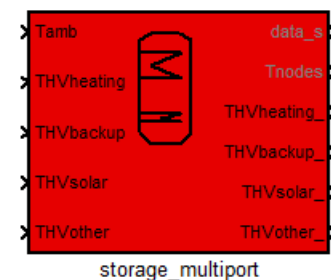


Abbildung 7.8 Block storage\_multiport

Der Ausgang *data\_s* gibt Speichertemperaturen und Energieflüsse aus (siehe subsystem).

Die Blöcke *storage\_combisystem1* und *storage\_combisystem2* sind Varianten des *storage\_multiport*.

### 7.4.3.3 Tank-in-Tank-Speicher

Der *tank-in-tank*-Speicher besitzt zwei integrierte Rohr-Wärmeübertrager mit jeweils einem Ein- und Ausgang für Wärmequellen. Über *THVsolar* wird der Solarkreis, über *THVbackup* ein Zusatzheizgerät (z.B. Gaskessel) an den Speicher angeschlossen. Am Eingang *Tamb* wird mittels des Moduls *Constant* eine konstante Umgebungstemperatur (in °C) angelegt.

Die Anschlüsse *THVdhw* führen zum innen liegenden Warmwasserbehälter. Hier kann das Modul für Brauchwasser-Zapfungen angeschlossen werden (siehe 3.7). Die Anbindung einer Heizungsanlage findet über die Anschlüsse *THVheating* statt. Der Ausgang *data\_s* dient der Datenerfassung.

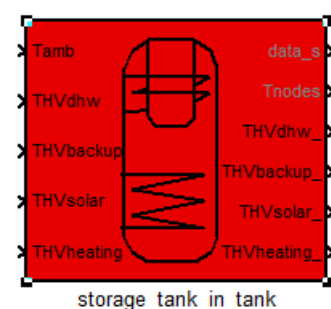


Abbildung 7.9 Block storage\_tank\_in\_tank

#### 7.4.4 Wärmeübertrager

An den Anschlüssen *THVsource\_in* wird der wärmeabgebende Fluidstrom, über *THVload\_in* der wärmeaufnehmende Fluidstrom des Wärmeübertragers angeschlossen.

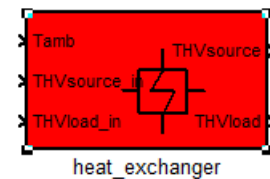


Abbildung 7.10 Block heat\_exchanger

#### 7.4.5 Brauchwasser-Zapfung

Zur Simulation von Brauchwasser-Zapfungen werden zwei Module benötigt (siehe Abbildung 7.11). Das Modul *water\_tab* erzeugt einen THV, dessen Massenstrom und Temperatur vorgegeben werden kann. Der erzeugte THV wird mit dem entsprechenden Anschluss am Brauchwasser-Speicher verbunden (z.B. *THVload*; *THVdhw*). Der entsprechende Speicher-Ausgang kann dann mit dem Modul *Terminator* geschlossen werden, da er nicht weiter benötigt wird. Da die Zapfungen nur zu bestimmten Zeiten stattfinden sollen, muss mit Hilfe des Blocks *DHW\_40\_20\_40* ein „Zapfverlauf“ erstellt werden. Dieser aktiviert, bzw. deaktiviert das Modul *water\_tab* zu den vorgegebenen Zeiten. In den Blockparametern des Blocks *DHW\_40\_20\_40* werden die Periode (Time Period, meistens 1 Tag =  $24 \cdot 3600$  (s)), die Zeiten (Time Values), die Dauer der Aktivierung (Duration) und der Output definiert. Ein Zapfprogramm von 8:00 bis 8:06 und 12:00 bis 12:06 hätte folgende Parameter, siehe Abbildung 7.12.

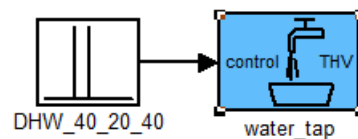


Abbildung 7.11 Brauchwasser-Zapfung mit den Blöcken DHW\_40\_20\_40 und water\_tab

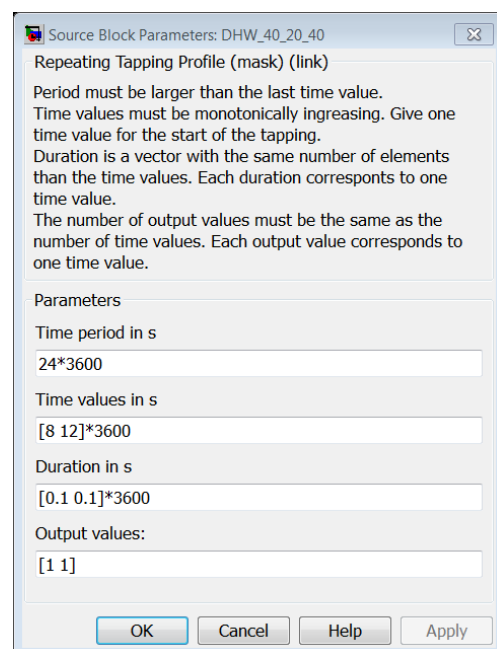


Abbildung 7.12 Parameter des Blocks DHW\_40\_20\_40

#### 7.4.6 Pumpen

Zur einfachen Simulation von Pumpen kann das Modul *pump\_const* benutzt werden. Es wird ein definierter Massenstrom ohne Berücksichtigung von Druckverlusten in Leitungen oder Einbauten simuliert. Zur Ansteuerung der Pumpen siehe Kapitel 7.4.6.

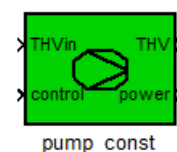


Abbildung 7.13 Block pump\_const

### 7.4.7 Regelung

Der Regler *controller\_bang\_bang* bestimmt, wann Pumpen oder andere Komponenten aktiviert werden. Realisiert wird die Hysterese-Regelung in Verbindung mit Temperatursensoren (siehe Abbildung 7.14). An den Eingängen des Controllers werden zwei Temperaturen angelegt (*Thot* und *Tcold*), z.B. die Temperatur am Kollektorausgang sowie die Temperatur an einer bestimmten Stelle des Speichers, welche miteinander verglichen werden. Am Controller werden die Einstellungen für *deltaT-on*, *deltaT-off* und die *maximum temperature* festgelegt. Wird die Differenz von *Thot-Tcold* größer als *deltaT-on*, so gibt der Controller eine 1 aus (Pumpe An). Wird die Differenz kleiner als *deltaT-off*, beträgt der ausgegebene Wert 0 (Pumpe Aus). Außerdem wird das Ausgangssignal 0, wenn *Thot* oder *Tcold* größer sind als die *maximum temperature*.

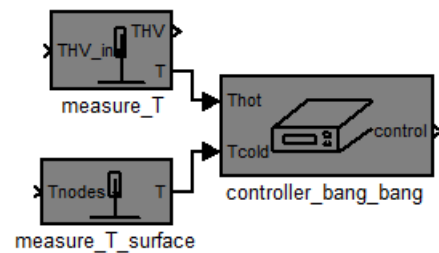


Abbildung 7.14 Block *controller\_bang\_bang* mit Temperatursensoren

Als Temperatur-Sensoren stehen zwei verschiedene Module zur Verfügung. Das Modul *measure\_T* misst die Temperatur eines THVs, z.B. zwischen Kollektor und Speicher, während das Modul *measure\_T\_surface* die Temperatur an einer beliebigen Höhe des Speichers messen kann (Anschluss am Speicher über *Tnodes*).

Verwendet man kein Sensor-Modul, entsteht ein *Algebraic Loop*, der als Warnung von Simulink ausgegeben wird. Das bedeutet, dass eine Eingangsgröße eines Blockes direkt (ohne zeitliche Verzögerung) von einer Ausgangsgröße desselben Blockes abhängig ist.

Beispiel: Eine Pumpe ist an einen Speicher angeschlossen; die Pumpe wird von einem *controller\_bang\_bang* in Abhängigkeit von der Speichertemperatur gesteuert. Die Speichertemperatur ist aber wiederum davon abhängig, ob die Pumpe gerade an oder aus ist und somit der Speicher Energie aufnimmt oder abgibt. Dies ist in einem Zeitschritt nicht lösbar. Die Sensor-Blöcke beinhalten eine thermische Trägheit und einen Startwert und brechen den *Algebraic Loop*.

### 7.4.8 Messung der Leistung und Energie

Mit dem Modul *energy\_meter* lässt sich die übertragene Leistung und Energiemenge eines Fluidstroms berechnen. An den Ein- und Ausgang des Moduls müssen THV-Vektoren eines Fluidstroms angeschlossen werden. Die Outputs sind die übertragene Leistung in Watt und die übertragene Energie in Joule. Um die Energiemenge in Kilowattstunden anzugeben, muss der Output *energy* durch  $3,6 \cdot 10^6$  dividiert werden. Hierfür kann der Block *gain* (gain:1/3600000) verwendet werden. (Der Übersicht wegen kann dies auch im Subsystem des *energy\_meter* geschehen.)

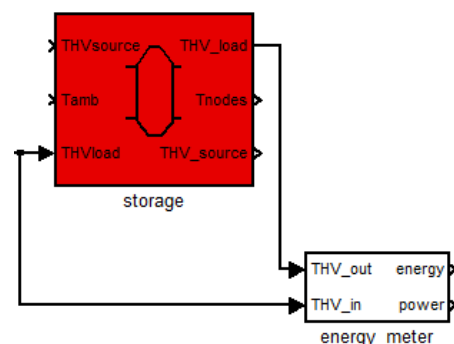


Abbildung 7.15 Block *energy\_meter* am Speicher angeschlossen

### 7.4.9 Daten ansehen und speichern

Zum Anzeigen und Speichern der Messdaten werden Simulink-Blöcke verwendet, siehe Kapitel 4.2.1.

Zur Auswahl eines bestimmten Wertes aus einem THV, AIV oder S-Vektor stehen die Blöcke *THV\_select*, *AIV\_selector* und *S\_selector* zur Verfügung (Simulink-Browser: CAR-NOT/outputs).

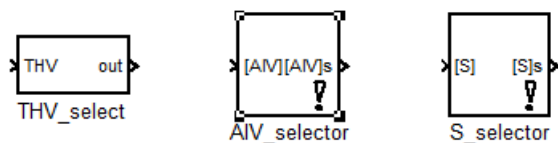


Abbildung 7.16 Selector-Blöcke

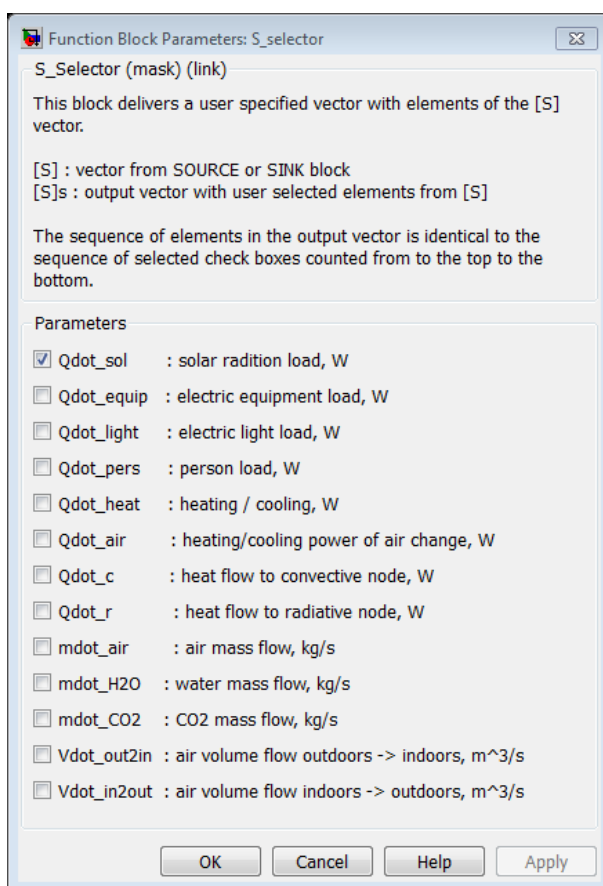


Abbildung 7.17 Function Block Parameters: S\_selector

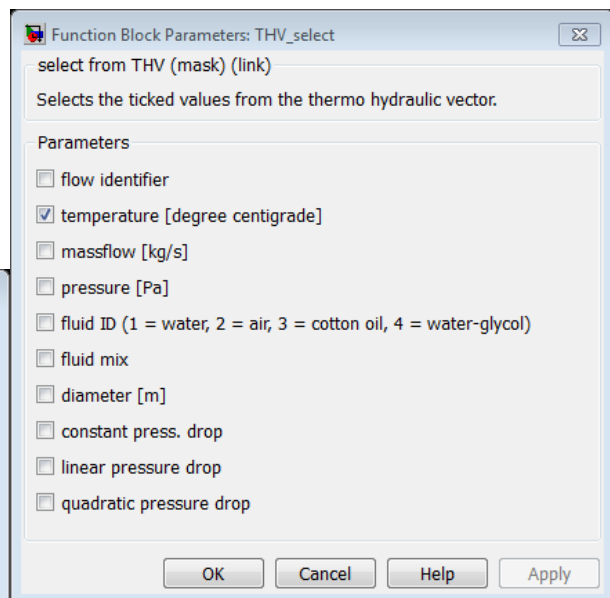


Abbildung 7.18 Function Block Parameters: THV\_select

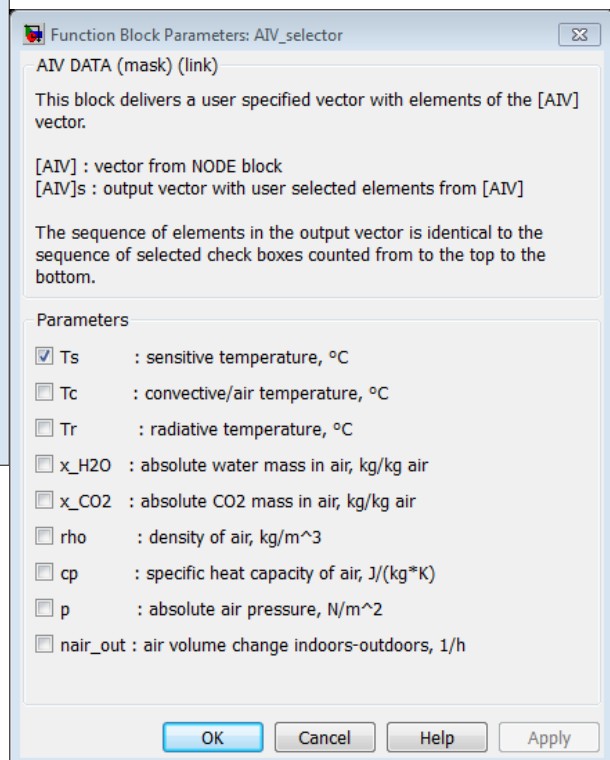


Abbildung 7.19 Function Block Parameters: AIV\_selector



## 8 Literaturverzeichnis

**Angermann, Anne. 2009.** *MATLAB - Simulink - Stateflow, Grundlagen, Toolboxes, Beispiele.* München : Oldenbourg, 2009. 978-3-486-58985-6.

**Hafner, Bernd. 2012.** CARNOT 5.2 Manual. 2012.

**The Mathworks, Inc. 2012.** Matlab R2012a Help. 2012.

—. **2013.** Stateflow. [Online] 2013. [Zitat vom: 28. 03 2013.] <http://www.mathworks.de/products/stateflow/>.

## 9 Abbildungsverzeichnis

Abbildung 3.1 MATLAB-Desktop	5
Abbildung 3.2 Import Wizard	6
Abbildung 3.3 Grundlegende Berechnungen und Variablen	7
Abbildung 3.4 Matrizen erstellen	7
Abbildung 3.5 Matrixoperationen	7
Abbildung 3.6 String erstellen	8
Abbildung 3.7 M-Skript for-schleife.m	9
Abbildung 3.8 Ausgabe des M-Skript for-schleife.m	10
Abbildung 3.9 M-Skript if_bedingung.m	11
Abbildung 3.10 Ausgabe des M-Skript if_bedingung.m	11
Abbildung 3.11 Plotten y über x	12
Abbildung 3.12 Figure 1	13
Abbildung 3.13 Figure 1 mit Achsenbeschriftungen, Property Editor und Plot Browser	13
Abbildung 3.14 M-Function createfigure	14
Abbildung 3.15 Subplots (The Mathworks, Inc., 2012)	14
Abbildung 4.1 Simulink Library Browser	15
Abbildung 4.2 Verbindung von Blöcken	16
Abbildung 4.3 Simulink-Modell Sinusaddition	17
Abbildung 4.4 Source Block Parameters: Sine Wave	17
Abbildung 4.5 Scope	17
Abbildung 4.6 Simulink-Modell Sinusaddition mit Display und Speicherblöcken	18
Abbildung 4.7 Scope parameters	18
Abbildung 4.8 Workspace mit variable wert	19
Abbildung 4.9 Constant-Block	19
Abbildung 4.10 Source Block Parameters: Constant mit Variablenname wert	19
Abbildung 4.11 Ein- und Ausgang Subsystem	20
Abbildung 4.12 Simulink Configuration Parameters	21

Tabellenverzeichnis	42
Abbildung 6.1 Stateflow-Library	23
Abbildung 6.2 Stateflow-Chart	23
Abbildung 6.3 Stateflow-Editor	24
Abbildung 6.4 Exklusive Zustände	25
Abbildung 6.5 Parallel-Zustände	25
Abbildung 6.6 Subchart	26
Abbildung 6.7 Inhalt eines Subcharts	26
Abbildung 6.8 Aktionen in Zuständen	26
Abbildung 6.9 Bedingungen	27
Abbildung 6.10 Chart mit Verbindungspunkt und Zeitverzögerung	27
Abbildung 6.11 Stateflow-Chart mit Ein- und Ausgang	28
Abbildung 6.12 Model Explorer	28
Abbildung 7.1 CARNOT-Library	29
Abbildung 7.2 CARNOT im Simulink Library Browser	29
Abbildung 7.3 Auswahl-Fenster CARNOT Demos	30
Abbildung 7.4 Weather-Blöcke	34
Abbildung 7.5 weather_simple_model	35
Abbildung 7.6: Block collector_flat_plate mit den Modulen für Wetterdaten und Position	36
Abbildung 7.7 Block storage	37
Abbildung 7.8 Block storage_multiport	37
Abbildung 7.9 Block storage_tank_in_tank	37
Abbildung 7.10 Block heat_exchanger	38
Abbildung 7.11 Brauchwasser-Zapfung mit den Blöcken DHW_40_20_40 und water_tap	38
Abbildung 7.12 Parameter des Blocks DHW_40_20_40	38
Abbildung 7.13 Block pump_const	38
Abbildung 7.14 Block controller_bang_bang mit Temperatursensoren	39
Abbildung 7.15 Block energy_meter am Speicher angeschlossen	39
Abbildung 7.16 Selector-Blöcke	40
Abbildung 7.17 Function Block Parameters: S_selector	40
Abbildung 7.18 Function Block Parameters: THV_select	40
Abbildung 7.19 Function Block Parameters: AIV_selector	40

## 10 Tabellenverzeichnis

Tabelle 3.1 MATLAB-Dateiformate	6
Tabelle 5.1 MATLAB-/ Simulink-Shortcuts	22
Tabelle 7.1 CARNOT Demos	31
Tabelle 7.2: Elemente des THV-Vektors	32

Tabelle 7.3 Beschreibung von Element Nr. 5 des THV-Vektors, fluid type	33
Tabelle 7.4 Elemente des Weather data Vektors	33
Tabelle 7.5 Elemente des AIV-Vektors	34
Tabelle 7.6 Elemente des S-Vektors	34