Live Script Prüfungsvorbereitung INF TOOIS, AV21a - Fabian Beeler

SW01 - Einführung

MATLAB = **MAT**rix **LAB**ortatory

Command Window Clear Befehl = clc

Workspace Clear = clear all

π = pi, ∞ = inf, Not-a-Number = NaN, imaginäre Einheit = i und j

Zeilenvektor x = start:schrittweite:ziel

Linearer Vektor y = linspace(start, ziel, anzahl)

Logarithmischer Vektor z = logspace(start, ziel, anzahl)

Einheitsmatrix I = eve (Zeilen, Spalten)

Matrix mit Einträgen 1 L = ones(zeilen, spalten,typ)

Matrix mit Einträgen 0 Z= zeros(zeilen,spalten,typ)

Matrix mit Zufallswerten zwischen 0 und 1 R = rand(zeilen,spalten)

size(Matrix) gibt Zeilen und Spalten aus (Dimensionen mxn)

Befehl 'ver' gibt die Informationen zur vorliegenden Version an

Zeichen für ungleich: ~=

Mit dem Befehl 'save x y' wir die Variable y gespeichert!!

```
x = 1:0.01:10
```

```
x = 1×901
    1.0000    1.0100    1.0200    1.0300    1.0400    1.0500    1.0600    1.0700 ···
```

```
x1 = 5;2;30
```

```
ans = 30
```

```
y = linspace(10,20,10)
```

```
y = 1×10
   10.0000   11.1111   12.2222   13.3333   14.4444   15.5556   16.6667   17.7778 ···
```

```
z = logspace(10,20,10)
```

```
z = 1×10
10^20 ×
    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0005    0.0060 ···
```

```
I = eye(4)
```

```
I = 4×4
```

```
     1     0     0     0
     0     1     0     0
     0     0     1     0
     0     0     0     1
```

```
L = ones(3,3,"double")
```

```
L = 3×3
     1     1     1
     1     1     1
     1     1     1
```

```
Z = zeros(3,3,"double")
```

```
Z = 3×3
     0     0     0
     0     0     0
     0     0     0
```

```
Y = 3 * ones(3,3)
```

```
Y = 3×3
     3     3     3
     3     3     3
     3     3     3
```

```
R = rand(2,3)
```

```
R = 2×3
    0.2785    0.9575    0.1576
    0.5469    0.9649    0.9706
```

```
k = size(I)
```

```
k = 1×2
     4     4
```

```
l = mod(3,4) %Resten aus 3/4
```

```
l = 3
```

```
n=3
```

```
n = 3
```

```
factorial(n) %n!
```

```
ans = 6
```

```
numel(t)        %Gibt Anzahl Element zrück in t
```

```
Unrecognized function or variable 't'.
```

Mit Matrizen abreiten

Alle Werte einer Mateix die grösser sind als 5 sollen auf 0 gesetzt werden:

```
M = [10 1 10 5; 2 3 2 9; 10 6 10 2; 7 10 10 5]
idx = M > 5
M(idx) = 0
```

Logisches indexieren mit einem logical array

```
v = [1 2 3 4]
idx = logical([1 0 1 0]) %logical muss explizit angegeben werden
v2 = v(idx)

% Zeitvektor definieren
t = linspace(0,10,20);

% Mathematische Funktion wirkt elementweise
y = sin(t)

% Vegleichsoperation
y >= 0.5

% Elemente die Bedigung erfüllen verändern mit logischem idexieren
y2 = y;
y2(y2 > 0.5) = 0
```

Aus den Übungen

Sklarprodukt --> Ergebis ist eine Zahl, also braucht man einen Spalten und einen Zeilenvektor

```
skp = x*x1
```

Inverse berechen

```
Q = [1 2;3 4];
Qinv = inv(Q)
```

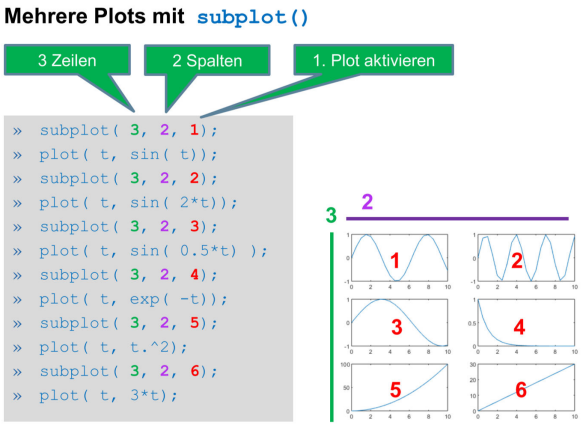
Zeilen und Spalten aus Matrizen löschen

```
Qneu = Q;
Qneu(1,:) = 5; %Zeilen umschreiben
```

```
Qneu(:,2) = 6; %Spalten umschreiben
Qneu(2,1) = 8
Qneu(:,1:2) = [] %Spalten 1-3 werden gelöscht

% Element ausgeben
a = Q(1, 2)

e = eig(Q) %Eigenwerte

% Element editieren
B(2, 2) = 7

% Grösse von Matrizen und Vektoren auslesen
[Q_row, Q_col] = size(Q)
x_len = length(x)

% Mathematische Operationen mit Matrizen
matrix_multi = Q * Q
element_multi = B .* B
Q_t = transpose(Q)  %Matrix transponieren
Q_t = Q.'           %Matrix transponieren
```

SW02 - Plotten und I/O Ops

Subplots

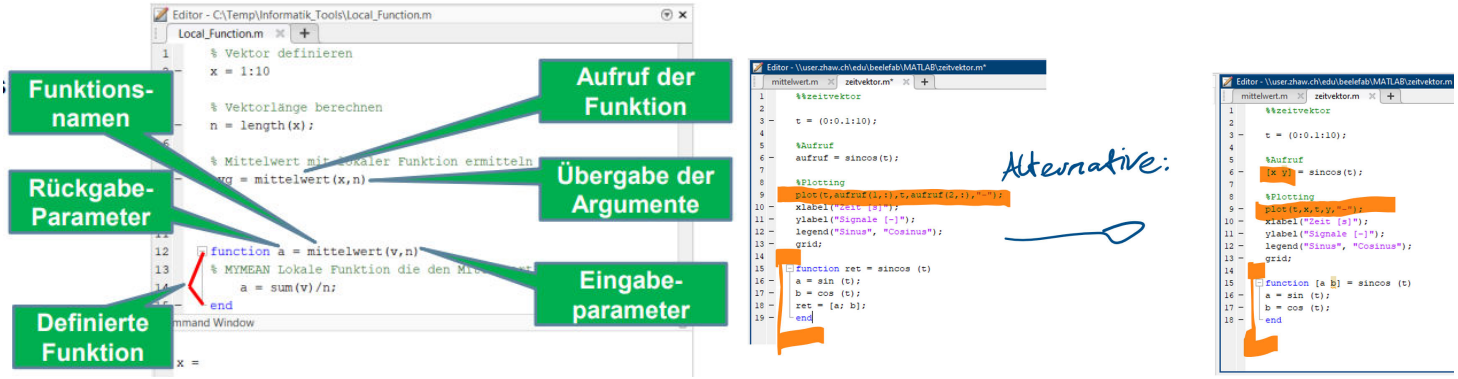


Imports mit dem Befehl: uiimport

```
% Daten plotten als Beispiel
figure(1)
set(gcf, 'Color', [1 1 1]);
plot(t, y, 'Linewidth', 2);
hold on
plot(t, y2, '--r')
xlabel('Time (s)');
ylim([-1 2]);
title('Plot');
grid on;
axis([0 10 -1 1])
```

SW03 Scripts und Functions

lokale Functions müssen immer <u>am Ende</u> vom Script stehen!




```
x = [1 3 5];
y = [6 8 2];

resultat = [];

% Schleife mit for...end
for i = 1:length(x)
    % Eigene Funktion "addNumbers" aufrufen
    %resultat(i) = addNumbers(x(i), y(i));

    % Abzweigung mit if...end
    if resultat > 10 %Korrekterweise müsste stehen if resultat(i) !!!!
        disp('Resultat ist grösser als 10');
    end
end

disp(resultat)
```

SW04 Symbolische Berechnungen - Symbolic Toolbox

```
% Ableiteng von symbolischen Ausdruecken
syms v x y
f = sin ( x * y^2 ) * cos ( v * x * y )

df_x = diff(f, x) %Differentiate / Ableiten
df_v = diff(f, v)

% Ersetzen von Variablen
subs(f,x,3) % Symbolic substitution eine sym Variable durch numerische Variable ersetzten

% Lösen von symbolischen Gleichungen
x = solve(6*x^2 -6*x^2*y + x*y^2 -x*y + y^3 -y^2 == 0, y) %lösen dsolve Diff. Equations solver

pretty(x) %Lesbarkeit
fplot(x) %plotten 2D
```

SW05 boolsche Algebra

```
% Logische Ausdruecke auswerten
A = [0 0 1 1];
B = [0 1 0 1];
```

UND / AND - Konjuktion --> wenn alle Eingangssignale 1 sind, wird Ausgangssignal 1

```
X = and(A, B)
X = A & B
```

ODER / OR - Disjuktion --> wenn mind. 1 Signal der Wert 1 hat, hat Ausgangssignal Wert 1

```
X = or(A, B)
X = A | B           % mit AltGr und 7
```

NICHT / NOT - Negation --> wenn Eingangssignal 0 ist, Ausgangssignal 1

```
X = not(A)
X = ~A
```

NAND / Not AND --> wenn nicht alle Eingangssignale 1 sind, wird Ausgangssignal 1

```
X = not( and(A,B))
X = ~(A&B)
```

NOR / Not OR --> wenn alle Eingangssignale 0 sind wird das Ausgangssignal 1

```
X = not(or(A,B))
X = ~(A|B)
```

XOR / exclusice OR / Antivalenz --> wenn Eingangssignale unterschiedlich sind, wird das Ausgangssignal 1

```
X = xor(A,B)
```

XNOR /exclusive Not OR / Äquivalenz --> wenn alle Eingangssignale gleich sind wird Ausgangssignal 1

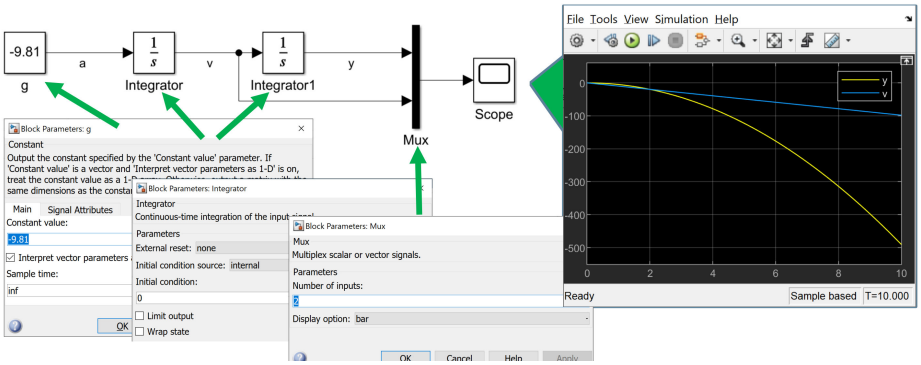
```
x = ~xor(A,B)

% Logische Ausdruecke vereinfachen
% In Octave muss das Symbolic-Packet zuerst geladen werden mit: pkg load symbolic
syms S1 S2 S3
X = simplify((~S1 & S2 & ~S3) | (S1 & ~S2 & ~S3) | (S1 & S2 & ~S3))

'C = A+B' %ist kein boolscher Ausdruck!
```

SW06 Simulink

Simulink – Beispiel 2

School of Engineering

Aufgabe: Berechnung der Funktion $\ddot{y} = g$



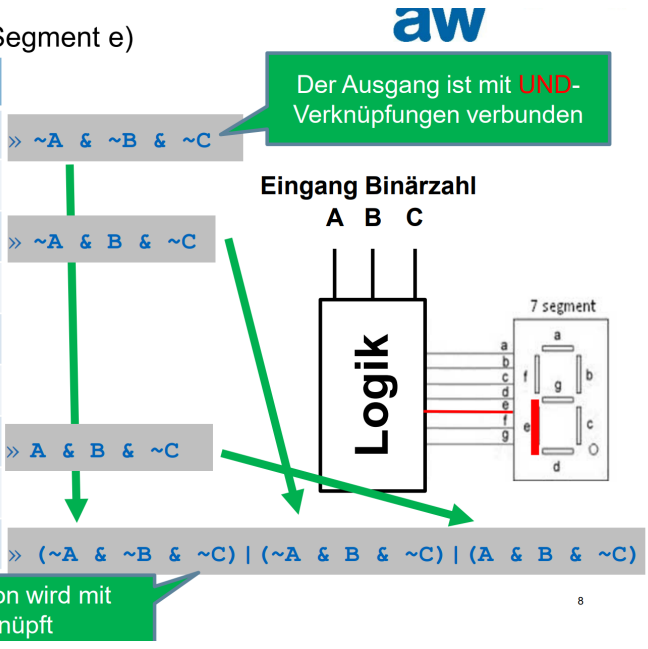
SW07 Logische Vereinfachnung und Logik mit Simulink

Nr	Gesetz	Rechenregel	
1	Identitätsgesetz	$A \cdot 0 = 0$	» A=1; B=1
2		$A \cdot 1 = A$	» A&0
3		$A \cdot A = A$	» A&1
4	Komplementgesetz	$A \cdot \bar{A} = 0$	» A&A
5	Identitätsgesetz	$A + 0 = A$	» A&~A
6		$A + 1 = 1$	» A|0
7		$A + A = A$	» A|1
8	Komplementgesetz	$A + \bar{A} = 1$	» A|A
9		$\bar{\bar{A}} = A$	» A|~A
10	Kommutativgesetz	$A \cdot B = B \cdot A$	» ~(~A)
11		$A + B = B + A$	» A&B; B&A
			» A|B; B|A

Nr	Gesetz	Rechenregel	
12	Assoziativgesetz	$(A \cdot B) \cdot C = A \cdot (B \cdot C) = A \cdot B \cdot C$	» A=1; B=1; C=1; D=1;
13		$(A + B) + C = A + (B + C) = A + B + C$	» (A&B)&C; A&(B&C); A&B&C
14	Distributivgesetz	$(A \cdot B) + (A \cdot C) = A \cdot (B + C)$	» (A|B)|C; A|(B|C); A|B|C
15		$(A + B) \cdot (A + C) = A + (B \cdot C)$	» (A&B)|(A&C); A&(B|C)
16	Allg. Distributivgesetz	$(A + B)(C + D) = AC + AD + BC + BD$	» (A|B)&(A|C); A|(B&C)
17		$AB + CD = (A + C)(A + D)(B + C)(B + D)$	» (A|B)&(C|D); A&C|A&D|B&C|B&D
18	Absorbtionsgesetz	$A \cdot (A + B + C) = A$	» A&C|C&D;(A|C)&(A|D)&(B|C)&(B|D)
19		$A + (A \cdot B \cdot C) = A$	» A&(A|B|C)
20	De Morgan	$\overline{A \cdot B} = \bar{A} + \bar{B}$	» A|(A&B&C)
21		$\overline{A + B} = \bar{A} \cdot \bar{B}$	» ~(A&B); ~A|~B
			» ~(A|B); ~A&~B

Beispiel: Ansteuerung einer 7 Segment Anzeige (Segment e)

Zahl	A	B	C	Segment e	
0	0	0	0	1	$\bar{A} \cdot \bar{B} \cdot \bar{C}$
1	0	0	1	0	
2	0	1	0	1	$\bar{A} \cdot B \cdot \bar{C}$
3	0	1	1	0	
4	1	0	0	0	
5	1	0	1	0	
6	1	1	0	1	$A \cdot B \cdot \bar{C}$
7	1	1	1	0	
Logik für Segment e					$\bar{A}\bar{B}\bar{C} + \bar{A}B\bar{C} + AB\bar{C}$



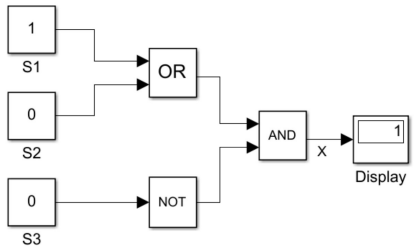
Zürcher Fachhochschule

8

In Simulink mit dem Block "Logical Operator" oder "Combinatorial Logic", bei Einsatz mit "Compinatorial Logic" immer noch einen Converter vorne hin setzen also zwischen Mux und Cominatorial (von double nach boolean).

Umsetzung mit Simulink eines logischen Ausdrucks:

```
» X = ~S3 & (S1 | S2)
```

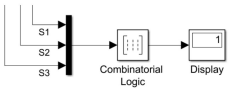


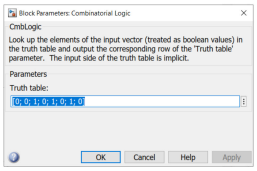
Einfaches Beispiel einer Logik mit Simulink

Combinatorial Logic Block für `» X = ~S3&(S1|S2)`

Wahrheitstabelle in Matlab:
[0 0 0 0; 0 0 1 0; 0 1 0 1; 0 1 1 0; 1 0 0 1; 1 0 1 0; 1 1 0 1; 1 1 1 0]

Der Combinatorial Logic Block liest S1, S2 und S3 ein und gibt anhand der letzten Spalte den Ausgabewert.





S1	S2	S3	X
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

SW08 LGS und lineare Regression

LGS

Ein weiteres Beispiel:

$$\begin{cases} 2x + y - z = 7 \\ x - 2y + 5z = -13 \\ 3x + 5y - 4z = 18 \end{cases}$$

```
A = [2 1 -1; 1 -2 5; 3 5 -4];
b = [7 -13 18]';
x = A\b
```

Alternativ:

```
A_inv = pinv(A);
disp(A_inv*b)
```

lineare Regression --> bedeutet überbestimmtes System, mehr GL als Unbekannte --> Lösung mit Methode der kleinsten Quadraten

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \epsilon = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} = \begin{bmatrix} 1 & x_{1,1} & \cdots & x_{1,r} \\ 1 & x_{2,1} & \cdots & x_{2,r} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{N,1} & \cdots & x_{N,r} \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_r \end{bmatrix} + \begin{bmatrix} \epsilon_0 \\ \epsilon_1 \\ \vdots \\ \epsilon_N \end{bmatrix} \quad \mathbf{X} = \begin{bmatrix} 1 & x_{1,1} & \cdots & x_{1,r} \\ 1 & x_{2,1} & \cdots & x_{2,r} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{N,1} & \cdots & x_{N,r} \end{bmatrix}$$

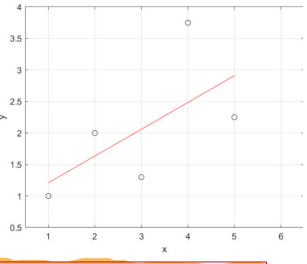
Unabhängige Variablen → (rechts)
Beobachtungen ↓ (unten)

↳ Fehler Korrektur

Beispiel: $y = \beta_0 + \beta_1 x$

```
y = [1, 2, 1.3, 3.75, 2.25]';
x = [1, 2, 3, 4, 5]';

X = [ones(size(x)), x];
b = X\y;

plot(x, y, 'ko', x, X*b, 'r')
grid on
```

↳ kleinste Quadrate
↳ X-Werte



If A is a rectangular m-by-n matrix with m ~= n, and B is a matrix with m rows, then A\B returns a least-squares solution to the system of equations A*x= B.

https://ch.mathworks.com/help/matlab/ref/mldivide.html

↳ Lösung vom kleinsten Quadrat

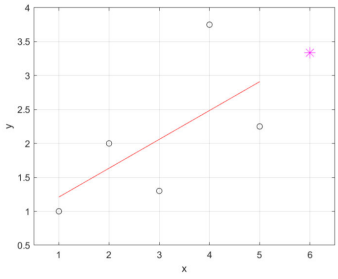
lineare Regression Vorhersagen:

Beispiel: Vorhersage an Punkt $x = 6$

$$\boldsymbol{\beta} = \begin{bmatrix} 0.7850 \\ 0.4250 \end{bmatrix} \qquad \boldsymbol{X_{pred}} = [\mathbf{1} \quad \mathbf{6}]$$

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} = 3.3350$$

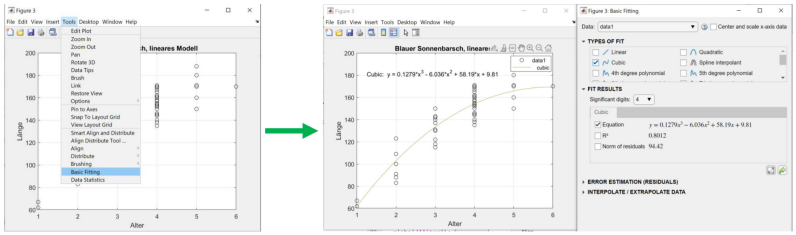
```
x_new = 6;
X_pred = [1, x_new];
plot(x, y, 'ko', x, X*b, 'r')
hold on;
plot(x_new, X_pred*b, 'm*', 'MarkerSize', 12);
hold off;
```



von linearen Termen zu quadratischen Funktionen:

- x elementweise quadrieren
- Darstellung mit neuer Matrize X_new
- in Plot einbauen
- Altrnative: Rohdaten plotten und dann im Plott Fenster unter Tools, Basic Fitting --> gewünschtes Modell auswählen

In MATLAB können auch lineare Regressionsmodelle interaktiv angepasst werden.



Beispiel, Blauer Sonnenbarsch (Fisch) Grösse:

$$y = \beta_0 + \beta_1 x + \beta_2 x^2$$

```
% ...
X = [ones(size(x)), x, x.^2];
b = X\y;

x_new = linspace(min(x), max(x))';
X_new = [ones(size(x_new)), x_new,
x_new.^2];

plot(x, y, 'ko', x_new, X_new*b, 'r');
% ...
```

SW09 ODE in Matlab und Simulink

bekannteste Methode für ODE: Runge-Kutta-Verfahren mit Solver

- ode23 --> Funktion der Syntax: *doc ode23*
- ode45 --> Funktion der Syntax: *doc ode45*

Beispiel 1:

Gegeben: $\dot{y} = 2yt$ mit dem Anfangswert $y(t = -1) = e$

Gesucht: Lösung für $y(t = [-1,1])$

Aufruf

```
[t, y] = ode45(@dgl1, [-1, 1], exp(1)); %dgl = Funktion, [-1, 1] O Interval, exp(1) = Anfangswe
plot(t, y)                              %Funktion ist am Ende des Dokumentes
```

Beispiel 2:

Gegeben: $\ddot{\alpha}(t) = -\frac{g}{l}\sin(\alpha(t))$, $g = 9.81\,{}^{m}/_{s^2}$, $\alpha(0) = \begin{pmatrix}\alpha(0)\\ \dot{\alpha}(0)\end{pmatrix}$

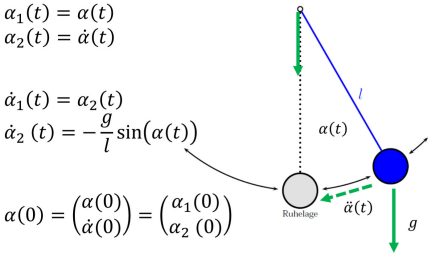
Dabei ist $\alpha(t)$ der Winkel, den das Pendel (der Länge l) zur Zeit t bezogen auf die Ruhelage einnimmt.

MATLAB erwartet, dass eine DGL höherer Ordnung als System von Gleichung erster Ordnung ausgedrückt wird.

$\alpha_1(t) = \alpha(t)$
$\alpha_2(t) = \dot{\alpha}(t)$

$\dot{\alpha}_1(t) = \alpha_2(t)$
$\dot{\alpha}_2(t) = -\frac{g}{l}\sin(\alpha(t))$

$\alpha(0) = \begin{pmatrix}\alpha(0)\\ \dot{\alpha}(0)\end{pmatrix} = \begin{pmatrix}\alpha_1(0)\\ \alpha_2(0)\end{pmatrix}$




```
[t, loesung] = ode23(@pendgl, [0, 25], [pi/3,0]);
plot(t, loesung(:,1), 'r-', t, loesung(:,2), 'g-'); %Funktion ist am Ende
```

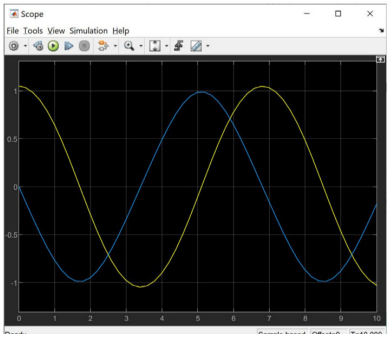
ODE in Simulink Tipps:

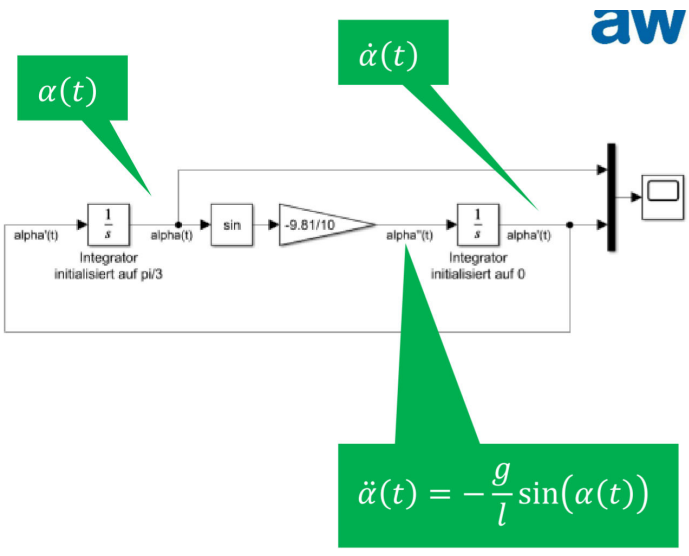
- Formen Sie die DLG so um, dass die Ableitung nach der höchsten Ordnung alleine links der Gleichung steht
- Starten Sie beim Modellieren von DGL in Simulink mit den Integranden

Beispiel 2 in Simulink:

$$\ddot{\alpha}(t) = -\frac{g}{l}\sin(\alpha(t))$$

$$\alpha(0) = \begin{pmatrix}\alpha(0)\\ \dot{\alpha}(0)\end{pmatrix}$$





SW10 Datenaustausch zwischen Matlab und Simulink

- Workspace = Matlab
- to Workspace-Block (Simu --> M/simout) und from Workspace-Block (M --> Simu/simin)
- mit "struct" eine Sammlung von Variablen erzeugen ***??????????????????????????????????***

```
a = struct
a.field1 = 1:5;
a.field2 = 'Hello'
```

- Block: in1 und out1 --> als Alternative zur Initialisierung mit Script
- Weitere Funktionen mit sim --> siehe "doc sim"
- get_param (Simu --> M) und set_param (M --> Simu)

```
%get_param
open('uebung_9_2_7_vorlage.slx');
valueGain = get_param('uebung_9_2_7_vorlage/Gain','Gain');      %Object, Parameter
disp(valueGain);
%set_param
g = -(1/20)
open('uebung_9_2_7_vorlage.slx');
set_param('uebung_9_2_7_vorlage/Gain','Gain',num2str(g));               %Object, Parameter, Va
```

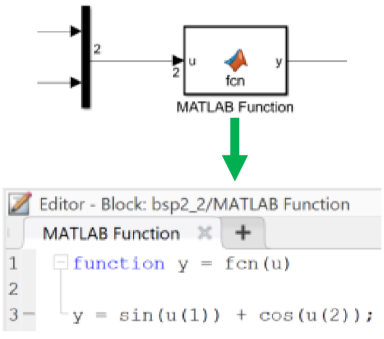
SW11 Simulationstechniken in Simulink

Es geht um Matlab Function Block --> ausführen von Matlab Code in Simulink und Subsystem erstellen.

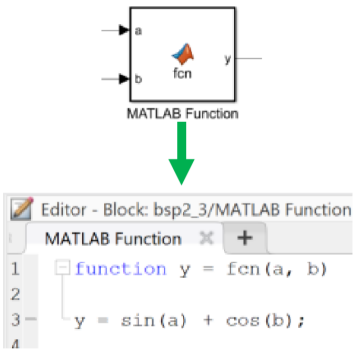
Zum Beispiel:

$$y = \sin(a) + \cos(b)$$

MATLAB-Function (1)



MATLAB-Function (2)



Um ein Subsystem zu erstellen, muss man alle gewünschten Blöcke markieren, rechtsklick und auf create Subysstem from Selcetion (oder Ctrl.+G). Alternativ gibt es auch den Block "Subsystem"

Lookups und Kennlinien

Beispiel Datei: uebung_07-slx / uebung_07_init.mlx --> C172 im Steigflug

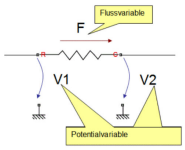
SW12 Simulink/Simscape

Simulink arbeitet mit Signalfluss --> definierte Richtigung

Simscpae arbeitet mit Leistungen oder Informationen --> in mehrere Richtungen

Simscape starten mit Befehl: *simscape*

Objektorientiert
Objekt mit Flussvariablen



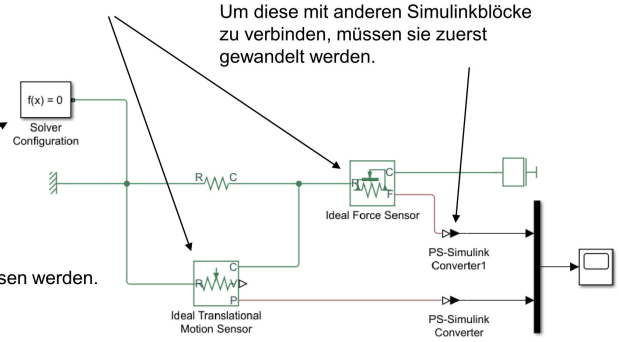
Disziplin	Potentialvariable	Flussvariable
Elektrotechnik	Spannung U	Strom I
Mechanik linear	Geschwindigkeit v	Kraft F
Mechanik rotierend	Winkelgesch. w	Drehmoment M
Wärmelehre	Temperatur T	Wärmefluss Q
Hydraulik	Druck p	Massestrom Q

Solver, Sensoren und Signalwandlung

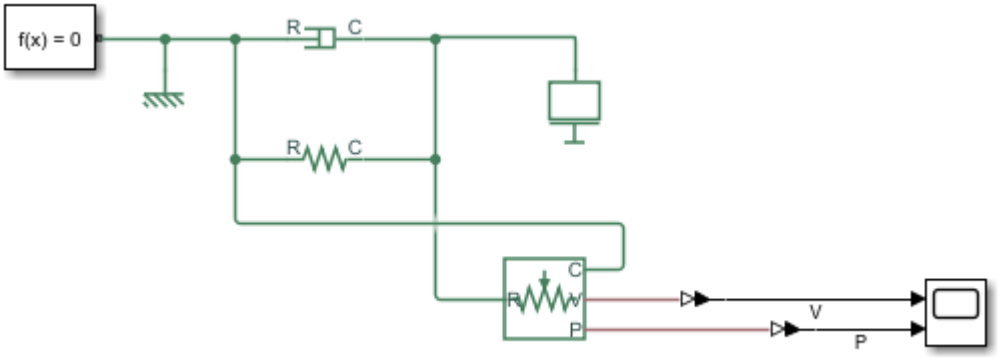
Sensoren
Physikalische Grössen können in Simscape nicht direkt ausgelesen werden. Das System benötigt Sensoren.

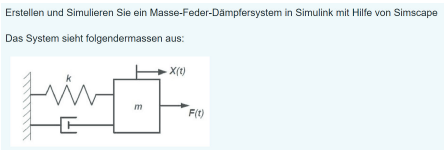
Signalwandlung
Die Ausgänge der Sensorblöcke haben ein spezielles Format (Physical Signal). Um diese mit anderen Simulinkblöcke zu verbinden, müssen sie zuerst gewandelt werden.

Solver
An das Modell muss ein Solver Configuration Block angeschlossen werden.



Dateien --> Übung 12-1-1

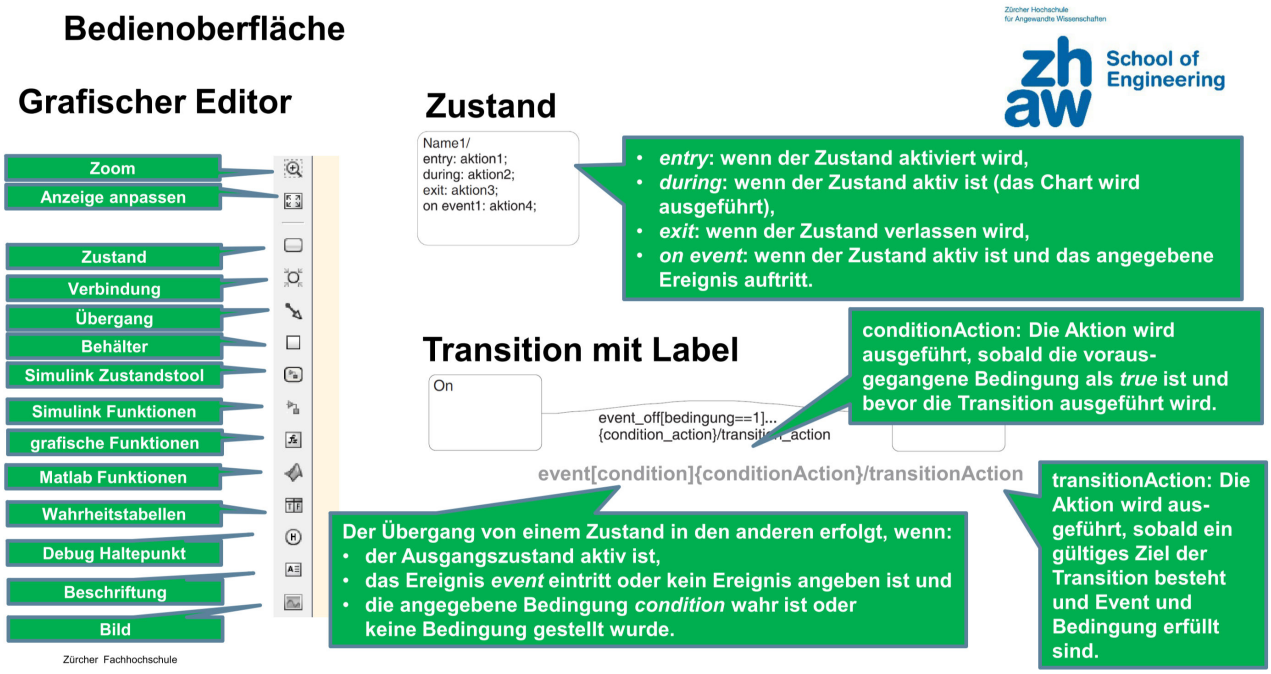




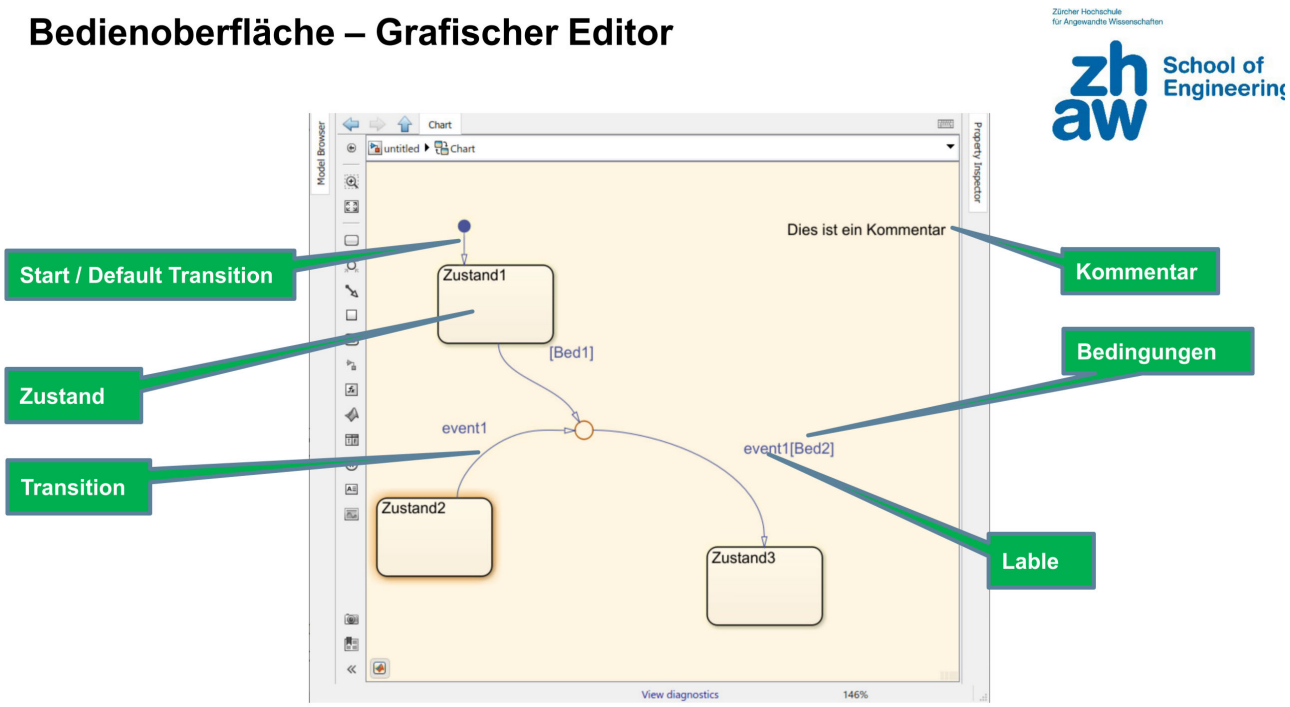
SW13 Stateflow und automatische Codegenerierung

- Modellierung und Simulation ereignisdiskreter Systeme
- Verwendet Zustandsdiatramme (state charts)
- Modelle werden graphisch programmiert
- Hierarchiebildung wird berücksichtigt
- Stateflow starten mit Befehl: *stateflow*

Bedienoberfläche



Bedienoberfläche – Grafischer Editor



automatische Code Generierung wird in den Folien von SW13 erläutert.

SW14 Open-Source Octave und Scilab

Ocatve Online: https://octave-online.net/

Scilab Online: https://cloud.scilab.in/

Unterschiede: https://en.wikibooks.org/wiki/MATLAB_Programming/Differences_between_Octave_and_MATLAB

Was ist Octave
- interaktive Skriptsprache z. B. für numerische Berechnungen, grösstenteils kompatibel zu Matlab
- freie Software für alle gängigen Systeme Kommandozeilenbasiert

Was ist Scilab
- interaktive Skriptsprache z. B. für numerische Berechnungen, kompatibel zu Matlab
- freie Software
- Xcos, ist die Alternative zu Simulink,
- Funktionen für
  2D- und 3D-Plots, numerische lineare Algebra, Polynom-Berechnungen, Statistik, Regelungstechnik, digitale Signalverarbeitung, I/O-Funktionen zum Daten Lesen und Schreiben
- Konverter von MATLAB nach Scilab

	MATLAB	Octave	Scilab
Ausgangslage	Matrix-Funktionen	numerische Berechnungen	numerische Berechnungen
verwendete Programmiersprachen	C, C++ und Java	C, C++ und Fortran	C, C++ und Fortran
Kompatibel zu Matlab	100%	~ 99%	~ 98%
grafische Modellierung	Simulink		Xcos
Schnittstelle	sehr umfangreich	eingeschränkt	umfangreich
Speicherbedarf (RAM) Installation	grosse ~10 GB	gering ~300 MB	gering
Kosten	2 300 CHF unb. 920 CHF / a *	Open Source	Open Source

Functions

Funtkion für ODE Beispiel 1:

```
function[dy] = dgl1(t,y)
dy = 2*t*y;
end
```

Funtkion für ODE Beispiel 2:

```
function[alphadot] = pendgl(t, alpha)
l = 10;
g = 9.81;
alphadot = [0;0];
alphadot(1) = alpha (2);            % Erste DLG 1. Ordnung
alphadot(2) = -(g/l)*sin(alpha(1)); % Zweite DLG 1. Ordnung
end
```

Thruthtable:

```
function T = truth_table(N)
% Truth Table Generator
% Mustafa U. Torun (Jan, 2010)
% ugur.torun@gmail.com
%
% T = truth_table(N);
%
% Inputs:
```

```
% N: Number of bits;
%
% Outputs:
% T: Truth Table;
%
% Example:
% T = truth_table(2)
% T =
%     0     0
%     0     1
%     1     0
%     1     1
L = 2^N;
T = zeros(L,N);
for i=1:N
   temp = [zeros(L/2^i,1); ones(L/2^i,1)];
   T(:,i) = repmat(temp,2^(i-1),1);
end
end
```