Table of Contents

Einleitung 1
Inititalisierung 2
Matlab Hilfe: 2
Zuweisungen in Matlab: Matlab verwendet standardmäßig Fließkommazahlen 2
Mathematische Funktionen und Operatoren 3
Eingabe von Vektoren und Matrizen 3
Verzweigungen und Schleifen 5
Matlabfunktionen: 8
Datenexport: 8
Löschen einzelner Variablen 9

Einleitung

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Einführung in Matlab
%
% Institut für Energiesysteme und elektrische Antriebe
% Arbeitsgruppe: Energy Economics Group (EEG)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Dieses Musterfile dient zum Erlernen der grundlegenden Matlabbefehle. Es
% basiert auf der Einführung von Matlab/Simulink des Institutes für
% Automatisierungstechnik (Gruppe für komplexe dynamische Systeme) siehe
% auf www.acin.tuwien.ac.at.

% Es handelt sich dabei um ein sogenanntes m-file, in welchem der Code zu
% einer Datei zusammengefasst und direkt ausgeführt werden kann. Neben
% diesem m-file kann der Code auch direkt im Command Window hinter der
% Eingabeaufforderung ">>" eingegeben und ausgeführt werden.

% Das Prozentzeichen dient in Matlab zum Einbinden von Kommentaren.
% Weiterhin ist es möglich durch zwei aufeinanderfolgende Prozentzeichen
% den Quelltext in Abschnitte zu unterteilen. Das Ausführen der Datei
% erfolgt entweder durch Eingabe des Dateinamens im Command Window oder
% durch Drücken der F5-Taste, einzelne Abschnitte können durch die
% Kombination STRG+Eingabe ausgeführt werden. Einzelne Zeilen im
% Programmcode können weiterhin markiert und durch Drücken der F9-Taste
% ausgeführt werden.
%
% Das Ergebnis der Auswertungen erscheint im Command Window. Es wird dabei
```

```
% zeilenweise ausgewertet. Wird die Programmzeile mit einem Semikolon
% abgeschlossen, so wird die Ausgabe unterdrückt.
```

Inititalisierung

```
% Beim Start eines neuen Programmes sollten zunächst einmal alle alten
% Variablen gelöscht und alle offenen Fenster gelöscht werden

clear all   % Es werden alle Variablen gelöscht
close all   % Es werden alle Fenster geschlossen
clc         % Das Command Window wird zurückgesetzt
```

Matlab Hilfe:

Es gibt zu allen Matlabbefehlen eine Hilfe, die entweder direkt im Command Window oder aber in einem zusätzlichen Hilfefenster angezeigt wird (die Hilfe enthält des Weiteren auch sehr gute Einführungs-beispiele

```
help sqrt   % Hilfe im Command Window
doc sqrt    % Hilfe in der Dokumentation (meist ausführlicher)
% Befehle können direkt im Command Window ausgeführt
% werden ohne "Run". Hilfreich bei Debug und Test
```

```
 SQRT   Square root.
    SQRT(X) is the square root of the elements of X. Complex
    results are produced if X is not positive.

    See also SQRTM, REALSQRT, HYPOT.

    Reference page for sqrt
    Other functions named sqrt

       codistributed/sqrt    gpuArray/sqrt    sym/sqrt
```

Zuweisungen in Matlab: Matlab verwendet standardmäßig Fließkommazahlen

(double: 64 Bit).

```
variable_1 = 2;          % Durch das Semikolon wird die Ausgabe
                         % unterdrückt. Ohne wird das Ergebnis im
                         % Command Window angezeigt
variable_2 = 2.5;        % Dezimalstelle mit Punkt
variable_3 = 4e5;
variable_3 = variable_1;% Überschreiben von Variablen.
                         % Ohne Semikolon wird
                         % das Ergebnis im Command Window angezeigt

% Vordefinierte Variablen

variable_4 = pi;         % pi = 3.14159
```

```
variable_5 = i;         % Komplexe Einheit
variable_6 = j;         % Komplexe Einheit
variable_7 = NaN;       % Not a number, ungültiges Ergebnis
```

Mathematische Funktionen und Operatoren

```
v1 = 2;
v3 = 3;

% Arithmetik

Add = v1+v3;   % Addition
Mul = v1*v3;   % Multiplikation
Sub = v1-v3;   % Subtraktion
Div = v1/v3;   % Division

% Funktionen

sqrt(v1);        % Wurzel
exp(v1);         % Exponentialfunktion
log(v1);         % Natürlicher Logarithmus
log10(v1);       % Zehnerlogarithmus

abs(v1);         % Betrag
sign(v1);        % Signum

real(2+i);       % Realteil
imag(2+i);       % Imaginärteil
conj(2+i);       % komplexe Konjugation
angle(2+i);      % Phase einer komplexen Zahl

sin(v1);         % Sinus
cos(v1);         % Cosinus
tan(v1);         % Tangens
cot(v1);         % Cotangens
```

Eingabe von Vektoren und Matrizen

```
zv  = [1,2,3];          % Zeilenvektor
sv  = [1;2;3];          % Spaltenvektor
zv2 = 1:1:5;            % Zeilenvektor [1 2 3 4 5]
                        % Verwendung start:(schrittweite:)ende
                        % Der Parameter schrittweite ist optional
zv3 = 1:2:5;            % Zeilenvektor [1 3 5]
zv4 = linspace(1,2,4);  % Erzeugen eines linearen Vektors
                        % zwischen 1 und 2 mit 4 Einträgen.
zv5 = [zv zv3];         % Zusammenfügen von Vektoren

% Eingabe von Matrizen

matr1 = [1 2;3 4];      % 2x2 Matrix
matr2 = ones(3);        % 3x3 Einsmatrix
matr3 = zeros(4,2);     % 4x2 Nullmatrix
```

```
matr4 = randn(4);       % 4x4 Zufallsmatrix
matr5 = eye(3);         % 3x3 Einheitsmatrix

% Zugriff auf Einträge von Vektoren und Matrizen

V = [5,8,13,21];                    % Zeilenvektor V mit 4 Einträgen
v_1 = V(1);                         % Erstes Element von V
v_4 = V(end);                       % Letztes Element von V
M = [1,2;3,4];                      % [2 x 2] Matrix
M_new(M(:,:) <= 2) = 0;             % Alle Einträge mit Bedingung auf
                                    % 0 setzen.
                                    % ":" Operator prüft alle Einträge
m_22 = M(1,2);                      % Element in der ersten Zeile
                                    % und zweiten Spalte
l = length(V);                      % Länge des Vektors V
s = size(M);                        % Größe der Matrix M
s_z = size(M,1);                    % Zeilenanzahl m bei [m x n]
Matrix_Ext = [M;M];                 % Zusammenfügen von zwei Matrizen
Matrix_Sub = Matrix_Ext(1:3,1:2);   % Auf Submatrize zurückgreifen
                                    % Zeile 1 bis 3 und Spalte 1 bis 2

M1 = reshape(Matrix_Ext,[8,1]);     % reshape ordnet die Matrix neu
                                    % Matrix als Argument
                                    % sowie die neue Zeilen und
                                    % Spaltenanzahl

sum(M);                             % Liefert einen Spaltenvektor
                                    % dessen Einträge die Summe der
                                    % jeweiligen Spalte ist


min(M);                             % Spaltenvektor mit Minimum
                                    % aus jeder Spalte

% Rechnen mit Matrizen und Vektoren

det(matr1);             % Determinante der Matrix
rank(matr1);            % Rang der Matrix
eig(matr1);             % Eigenwerte der Matrix
inv(matr1);             % Inverse der Matrix
matr1';                 % Transponierte der Matrix

z = [1,2,3];            % Zeilenvektor
SP = z*z';              % Skalarprodukt
z.*z;                   % Elementweises Ausführen der Multiplikation

matr1*[1;2];            % Matrix-Vektor-Produkt
matr1^2;                % Matrixquadrat

zv = [3,5,6,8,1];
[min1,min2] = min(zv);      % Minimum und Stelle in Vektor
[max1,max2] = max(zv);      % Maximum und Stelle in Vektor

mean(zv);                   % Mittelwert eines Vektors
```

```
std(zv);                    % Standartabweichung vom Mittelwert
sum(zv);                    % Summe der Vektorelemente
prod(zv);                   % Produkt der Vektorelemente

X = [6,9,4,1,7,8,2];        % Zeilenvektor definieren
Y = 2*X;
Y(X < 5) = 0;               % Falls Wert kleiner als 5 soll der Wert 0
                            % gesetzt werden
sum(Y);
sort(Y);                    % Einträge aufsteigend sortieren
```

Verzweigungen und Schleifen

```
% If-Verzweigungen

var = 3;

if var > 1
    {
        sprintf('Die Variable ist größer als 2') %Ausgabe im Command
 Window
    }
else
    {
        sprintf('Die Variable ist kleiner als 2')
    }
end

% Verschachtelte Verzweigungen

if ((var > 1) && (var<3))
    {
        sprintf('Die Variable liegt zwischen 1 und 3')
    }
elseif (var>3)
    {
        sprintf('Die Variable ist größer als 3')
    }
else
    {
        sprintf('Die Variable ist kleiner als 1')
    }
end

% Case Anweisung

switch var case 1; a=1, case{3,4,5}; a=5, otherwise a=10, end

% Schleifen

for i=1:2:10
    sprintf(strcat('Der aktuelle Index beträgt: ',num2str(i)))
    % Umwandlung von Zahlen in Strings mit dem Befehl num2str
```

```
    pause(1)     % Einführen von Pausen bei der Ausgabe
end

k=0;

while k<5
    sprintf(strcat('Der aktuelle Index beträgt: ',num2str(k)))
    % Umwandlung von Zahlen in Strings mit dem Befehl num2str
    pause(1)     % Einführen von Pausen bei der Ausgabe
    k=k+2;
end

% Das Abbrechen von Schleifen (und m-code allgemein) ist mit dem
% Befehl break möglich


while k>0
    sprintf(strcat('Der aktuelle Index beträgt: ',num2str(k)))
    % Umwandlung von Zahlen in Strings mit dem Befehl num2str
    pause(1)     % Einführen von Pausen bei der Ausgabe
    if k>10
        sprintf('Schleife unterbrochen')
        break
    end
    k=k+2;
end

% Anmerkung: Sollte unbeabsichtigt eine Endlosschleife entstanden sein
%            (oder eine andere Berechnung nicht terminieren), so kann
%            diese mittels STRG+C unterbrochen werden


ans =

    'Die Variable ist größer als 2'


ans =

    'Die Variable ist kleiner als 1'


a =

     5


ans =

Der aktuelle Index beträgt:1


ans =
```

```
Der aktuelle Index beträgt:3

ans =

Der aktuelle Index beträgt:5

ans =

Der aktuelle Index beträgt:7

ans =

Der aktuelle Index beträgt:9

ans =

Der aktuelle Index beträgt:0

ans =

Der aktuelle Index beträgt:2

ans =

Der aktuelle Index beträgt:4

ans =

Der aktuelle Index beträgt:6

ans =

Der aktuelle Index beträgt:8

ans =

Der aktuelle Index beträgt:10

ans =

Der aktuelle Index beträgt:12

ans =
```

```
Schleife unterbrochen
```

Matlabfunktionen:

Dies sind spezielle m-files, welchen Parameter übergeben werden können und dann das Ergebnis der Funktion zurückgeben. Der Name der Funktion muss immer gleich dem Namen des m-files sein Syntax: function [out] = name(in) Abschluss der Funktion mit end

```
help mittelwert   % Ausgabe der Hilfe

mittelwert(2,3);   % Ausgabe des Ergebnis

% Anmerkung: Variablen in einer Funktion sind lokal, sofern nicht
 anders
% deklariert. Allerdings sollte darauf wenn möglich verzichtet werden.

  Hier steht die Information zur function
  out = mittelwert(a,b) berechnet den Mittelwert der Zahlen a und b
```

Datenexport:

Daten werden in Matlab standardmäßig als sogenanntes mat-file exportiert

```
vasa = [1 2 3];

save save_data vasa % Abspeichern des Vektors vasa in der Datei
 save_data.mat

vasa2 = [4 5 6];

save save_data -append vasa2  %Anhängen des Vektors vasa2 an die
 existierende Datei
clear all
load save_data.mat   % Laden einer mat-Datei
vec = load('save_data.mat') % Direktes Zuweisen zu einer Variable

% Weitere Datenexporte und Importe

vasa = [1 2 3];
save save_data_asc.txt -ascii vasa   %Abspeichern in einer ascii-Datei
 (z.B. txt)
clear all
load -ascii save_data_asc.txt  % Laden der ascii-Datei
vasa = [1 2 3];
xlswrite('save_data_xls',vasa, 'my_data', 'B2');    % Abspeichern in
 einer Excel-Datei
clear all
var = xlsread('save_data_xls')
```

```
vec =

     vasa: [1 2 3]
    vasa2: [4 5 6]


var =

     1     2     3
```

Löschen einzelner Variablen

```
a = 1;
clear a
```

Published with MATLAB® R2015b