



Kurzeinführung in MATLAB mit Symbolic Toolbox

MATLAB

MATLAB ist ein kommerzielles Numerik-Programmpaket, das Studenten der Universität Stuttgart zur Verfügung steht:

<https://www.stud.uni-stuttgart.de/dienste/software/matlab.html>

MATLAB verwendet eine numerische Darstellung von Vektoren und Matrizen und erlaubt eine einfache Programmierung von Vektor/Matrixoperationen und stellt eine Vielzahl von Methoden zur Systemanalyse zur Verfügung. Viele so genannte Toolboxes sind verfügbar, welche die Funktionalität von MATLAB erweitern.

Die Befehle können direkt in dem MATLAB-Fenster eingegeben und ausgeführt werden. Alternativ können die Befehle auch in einer Textdatei `<name>.m` zusammengefasst werden. Dieses so genannte m-file wird durch Eingabe von `<name>` in dem MATLAB-Fenster geladen und ausgeführt.

`helpdesk`

`a = [3 -4 5]`

`b = [3 -4 5]'`

`A = [1 2 4; 2 13 23; 4 23 77]`

`A'`

`A(2,3)`

`A(:,3)`

`A(1,:)`

`zeros(n,m)`

`ones(n,m)`

`diag([1 2 3])`

`c = A * b`

`c = A \ b`

`inv(A)`

`[EV, ew] = eig(A)`

`plot(t,x,'g'),hold on;`

`whos`

startet die Online-Hilfe

definiert einen Zeilenvektor

definiert einen Spaltenvektor

definiert eine Matrix

Transponierte der Matrix A

Matrizelement A_{23}

3. Spalte der Matrix A

1. Zeile der Matrix A

definiert eine Nullmatrix

definiert eine Matrix mit Einsen

definiert eine Diagonalmatrix

Matrix-Vektor Multiplikation

löst ein lineares Gleichungssystem

Inverse einer Matrix

Matrix EV der Eigenvektoren, Vektor

ew der Eigenwerte der Matrix A

Zeichnet Vektor x über Vektor t

Zeigt alle im Workspace belegten Variablen



Zur numerischen Zeitintegration stellt MATLAB zwei explizite Einschrittverfahren (`ode23`, `ode45`), ein implizites Einschrittverfahren (`ode23s`), ein Adams-Bashforth-Moulton Prädiktor-Korrektor-Verfahren (`ode113`) und eine implizite “numerical differentiation formula” (`ode15s`) zur Verfügung. Alle Verfahren haben eine eingebaute Schrittweitensteuerung.

```
[t,X] = ode23('<name>',tspan,x0)
```

simuliert ein System gewöhnlicher Differentialgleichungen der Form $\frac{dx}{dt} = f(t, x)$. Ausgabe ist der Vektor t der Zeitschritte und die Matrix X der Zustandstrajektorien. Die Anzahl der Spalten von X entsprechen der Anzahl der Zustände und die Anzahl der Zeilen entspricht der Anzahl der Zeitschritte.

Eingabe ist mit $\langle \text{name} \rangle$ eine Funktion dx/dt , ein (2×1) Vektor $t_{\text{span}} = [t_0, t_{\text{end}}]$ welcher die Anfangs- und Endzeit enthält und der Vektor der Anfangsbedingungen $x_0 = [x_1(t_0) \ x_2(t_0) \cdots \dot{x}_1(t_0) \ \dot{x}_2(t_0) \dots]$.

Die Funktion $\langle \text{name} \rangle$ muss in einer extra Datei $\langle \text{name} .m \rangle$ mit entsprechendem Dateiname und folgender Syntax gespeichert sein:

```
function dx = <name>(t, x)
...
dx = ...
```

Beispiel Doppelpendel

```
function dx = doublependulum(t, x)

% state vector x = [alp, bet, Dalp, Dbet]';
alp = x(1); bet = x(2);
Dalp = x(3); Dbet = x(4);
% mass and geometry parameters
m = 1.0; l = 1.0; g = 9.81;
M = zeros(2, 2);
M(1, 1) = l^2*m*(3+2*cos(bet));
M(1, 2) = l^2*m*(cos(bet)+1);
M(2, 1) = l^2*m*(cos(bet)+1);
M(2, 2) = l^2*m;
k = zeros(2, 1);
k(1, 1) = -l^2*m*Dbet*sin(bet)*(Dbet+2*Dalp);
k(2, 1) = l^2*m*Dalp^2*sin(bet);
q = zeros(2, 1);
q(1, 1) = -m*g*l*(2*cos(alp)+cos(alp+bet));
q(2, 1) = -cos(alp+bet)*l*m*g;

dx = [[Dalp; Dbet]; inv(M)*(q-k)];
```



MATLAB Symbolic Toolbox

Mit der Symbolic Toolbox können in MATLAB neben numerischen Berechnungen auch symbolische Untersuchungen durchgeführt werden. Bis zu der MATLAB-Version R2008b basiert die Symbolic Toolbox auf dem MAPLE-Kernel, ab R2009a basiert diese auf MUPAD. In beiden Versionen sind folgende Grundfunktionen enthalten:

<code>syms a b</code>	definieren symbolischer Variablen
<code>c=a+b</code>	Grundrechenarten
<code>d=a*b</code>	
<code>a=5; b=2</code>	Werte zuweisen
<code>e=subs(c)</code>	Ersetzen aller symbolischen Ausdrücke
<code>f=subs(c,b)</code>	Ersetzen von <code>b</code> in <code>c</code>

Beispiel: Berechnung von Jacobi-Matrix und Beschleunigungsvektor

```
syms l1 alpha alpha_ Dalpha D2alpha beta Dbeta D2beta t_  
  
% Vektor der verallgemeinerten Koordinaten und Ableitungen  
y=[alpha;beta]  
Dy=[Dalpha;Dbeta]  
D2y=[D2alpha;D2beta]  
  
% Orstvektor  
r1=[l1*cos(alpha);l1*sin(alpha);0]  
  
% Zeitabhängige Variablen als Taylorreihe darstellen  
alpha_ = alpha + Dalpha*t_+0.5*D2alpha*t_^2  
  
% Berechnung der Geschwindigkeit v1 und Beschleunigung a1  
r1_=subs(r1,alpha,alpha_)  
v1_=diff(r1_,'t_')  
v1=subs(v1_,t_,0)  
  
a1_=diff(v1_,'t_')  
a1=subs(a1_,t_,0)  
  
% Jacobi-Matrix und lokale Beschleunigung  
J1=jacobian(r1,y)  
a1q=a1-J1*D2y
```