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 Toolboxen 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`	startet die Online-Hilfe
`a = [ 3 -4 5 ]`	definiert einen Zeilenvektor
`b = [ 3 -4 5 ]'`	definiert einen Spaltenvektor
`A = [ 1 2 4; 2 13 23; 4 23 77 ]`	definiert eine Matrix
`A'`	Transponierte der Matrix A
`A(2,3)`	Matrixelement A_{23}
`A(:,3)`	3. Spalte der Matrix A
`A(1,:)`	1. Zeile der Matrix A
`zeros(n,m)`	definiert eine Nullmatrix
`ones(n,m)`	definiert eine Matrix mit Einsern
`diag([1 2 3])`	definiert eine Diagonalmatrix
`c = A * b`	Matrix-Vektor Multiplikation
`c = A \ b`	löst ein lineares Gleichungssystem
`inv(A)`	Inverse einer Matrix
`[ EV, ew ] = eig(A)`	Matrix `EV` der Eigenvektoren, Vektor `ew` der Eigenwerte der Matrix A
`plot(t,x,'g'),hold on;`	Zeichnet Vektor `x` über Vektor `t`
`whos`	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 `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 `<name>` eine Funktion `dx/dt`, ein (2x1) Vektor `tspan = [t0, tend]` welcher die Anfangs- und Endzeit enthält und der Vektor der Anfangsbedingungen `x0` $= [x_1(t_0)\, x_2(t_0) \cdots \dot{x}_1(t_0)\, \dot{x}_2(t_0) \ldots]$.

Die Funktion `<name>` muss in einer extra Datei `<name.m>` 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:

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

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
```