

Matlab: eine kurze Einführung

Marcus J. Grote Christoph Kirsch
Mathematisches Institut
Universität Basel

4. April 2003

In dieser Einführung zu Matlab sind die im Praktikum I erworbenen Kenntnisse zusammengefasst. Einen Matlab-Primer findest du im Internet, z. B. unter <http://ise.stanford.edu/Matlab/matlab-primer.pdf>

1 Erste Schritte in Matlab

1.1 Wie fange ich an?

Starte Matlab mit dem Befehl `matlab`. Mit `help` Befehl kannst du Informationen zu jedem Matlab-Befehl erhalten. Erläuterungen zu den Matlab-Befehlen findest du auch in der Menüleiste unter «?». Der Befehl `clear all` löscht die gespeicherten Variablen.

1.2 Rechnen mit Zahlen

```
>1+2
```

```
ans =  
     3
```

```
>a=1+2;           % weist der Variablen a das Ergebnis der Rechnung 1+2 zu  
>a               % der Wert von a wird ausgegeben
```

```
a =  
     3
```

```
>2*7+3^4
```

```
ans =  
    95
```

```
>k=cos(3*pi/25)    % weist der Variablen k den Wert von cos(3*pi/25) zu
```

```
k =  
    0.9298
```

```
>format long      % Anzeige von 15 Stellen
```

```
>k
```

```
k =  
    0.92977648588825
```

```
>format short    % Anzeige von 5 Stellen
```

```
>k^3-3*k^2+sqrt(5)
```

```
ans =  
    0.4464
```

```
>sqrt(-1)        % imaginaere Einheit
```

```
ans =  
    0 + 1.0000i
```

```
>a=3+4i;         % komplexe Zahl
```

```
>real(a)         % Realteil
```

```
ans =  
    3
```

```
>imag(a)         % Imaginaerteil
```

```
ans =  
    4
```

```
>abs(a)          % Absolutbetrag
```

```
ans =  
    5
```

```
>round(5.4)     % runden
```

```
ans =  
    5
```

1.3 Vektoren

```
>x=[1 2 3] % Zeilenvektor
```

```
x =  
    1    2    3
```

```
>x(2) % 2. Eintrag von x
```

```
ans =  
    2
```

```
>y=[4;5;6] % Spaltenvektor
```

```
y =  
    4  
    5  
    6
```

```
>y(1:2) % Eintraege 1 und 2 von y
```

```
ans =  
    4  
    5
```

```
>y([1 3]) % Eintraege 1 und 3 von y
```

```
ans =  
    4  
    6
```

```
>length(x) % Anzahl Eintraege in x
```

```
ans =  
    3
```

```
>length(y)
```

```
ans =  
    3
```

```
>z=3*x % weist dem Vektor z das Ergebnis von 3*x zu
```

```
z =  
    3    6    9
```

```

>w=y+1          % zu jedem Eintrag von y wird 1 addiert

w =
     5
     6
     7

>x'             % transponieren: Zeilenvektor <-> Spaltenvektor

ans =
     1
     2
     3

>x*y           % Skalarprodukt

ans =
    32

>y*x           % Produkt einer 3x1-Matrix und einer 1x3-Matrix

ans =
     4     8    12
     5    10    15
     6    12    18

>x.*x         % komponentenweise Multiplikation

ans =
     1     4     9

>cos(x)       % cos wird auf jeden Eintrag von x angewendet

ans =
    0.5403   -0.4161   -0.9900

>z1=1:5       % Zahlen von 1 bis 5 mit Schrittweite 1

z1 =
     1     2     3     4     5

>z2=0:0.25:1  % Zahlen von 0 bis 1 mit Schrittweite 0.25

z2 =
     0    0.2500    0.5000    0.7500    1.0000

```

1.4 Matrizen und lineare Gleichungssysteme

```
>A=[1 2 3;4 5 6;7 8 9]
```

```
A =
```

```
    1    2    3
    4    5    6
    7    8    9
```

```
>det(A)      % Determinante
```

```
ans =
```

```
    0
```

```
>size(A,1)   % Anzahl Zeilen der Matrix A
```

```
ans =
```

```
    3
```

```
>size(A,2)   % Anzahl Spalten der Matrix A
```

```
ans =
```

```
    3
```

```
>B=[0 1 0;1 0 0;0 0 1]
```

```
B =
```

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

```
>A+B      % Addition von Matrizen
```

```
ans =
```

```
    1    3    3
    5    5    6
    7    8   10
```

```
>A*B      % Matrixmultiplikation
```

```
ans =
```

```
    2    1    3
    5    4    6
    8    7    9
```

```
>A.*B          % komponentenweise Multiplikation
```

```
ans =
```

```
    0    2    0
    4    0    0
    0    0    9
```

```
>A^3          % A*A*A
```

```
ans =
```

```
    468    576    684
   1062   1305   1548
   1656   2034   2412
```

```
>A'          % Transponierte
```

```
ans =
```

```
    1    4    7
    2    5    8
    3    6    9
```

```
>A(1,2)       % Eintrag von A an der Stelle (1,2)
```

```
ans =
```

```
    2
```

```
>A(2,1:3)     % 2. Zeile von A
```

```
ans =
```

```
    4    5    6
```

```
>A(1:3,3)     % 3. Spalte von A
```

```
ans =
```

```
    3
    6
    9
```

```
>A(1:2,2:3)   % Teilmatrix aus Zeilen 1 und 2, Spalten 2 und 3
```

```
ans =
```

```
    2    3
    5    6
```

```
>C=A([1 3],[1 3]) % Teilmatrix aus Zeilen 1 und 3, Spalten 1 und 3
```

```
C =  
    1    3  
    7    9
```

```
>A*C % Produkt einer 3x3-Matrix mit einer 2x2-Matrix
```

```
??? Error using ==> * Inner matrix dimensions must agree.
```

```
>b=rand(3,1) % Zufallsvektor der Laenge 3
```

```
b =  
    0.9501  
    0.2311  
    0.6068
```

```
>x=B\b % loest das lineare Gleichungssystem Bx=b
```

```
x =  
    0.2311  
    0.9501  
    0.6068
```

```
>D=zeros(2,3) % 2x3-Nullmatrix
```

```
D =  
    0    0    0  
    0    0    0
```

```
>E=ones(3,2) % 3x2-Einsmatrix
```

```
E =  
    1    1  
    1    1  
    1    1
```

```
>F=eye(3) % 3x3-Identitaetsmatrix
```

```
A =  
    1    0    0  
    0    1    0  
    0    0    1
```

```
>G=[C D;E F]
```

```
% Matrix zusammengesetzt aus Matrizen
```

```
G =
```

```
    1    3    0    0    0
    7    9    0    0    0
    1    1    1    0    0
    1    1    0    1    0
    1    1    0    0    1
```

1.5 Grafische Darstellungen 2-D

```
>x=0:0.01:2*pi;
```

```
% Zahlen von 0 bis 2*pi, Schrittweite 0.01
```

```
>plot(x,cos(x))
```

```
% zeichnet cos(x) gegen x
```

```
>plot(x,cos(x),'-',x,sin(x),'-.')
```

```
% zeichnet cos(x) und sin(x) gegen x
```

```
>axis([-0.1 2*pi+0.1 -1.1 1.1])
```

```
% legt den zu zeichnenden Bereich fest
```

```
>legend('Cosinus','Sinus')
```

```
% Legende
```

```
>xlabel('x')
```

```
% x-Achsen-Beschriftung
```

```
>ylabel('f(x)')
```

```
% y-Achsen-Beschriftung
```

```
>x=[1 2 3 4];
```

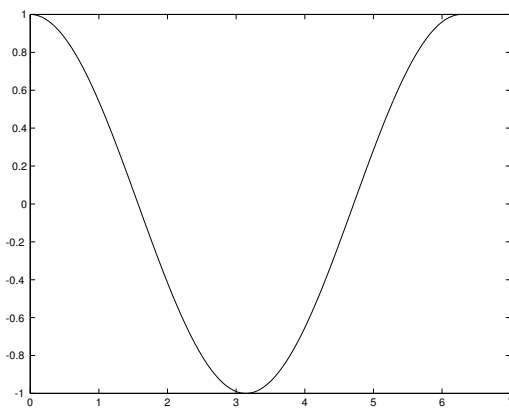
```
>y=[1 2.5 -1 0.5];
```

```
>plot(x,y,'o')
```

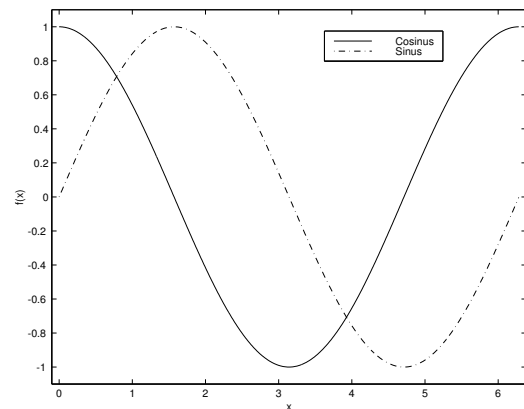
```
% zeichnet die Punkte (x,y)
```

```
>axis([0 5 -2 3])
```

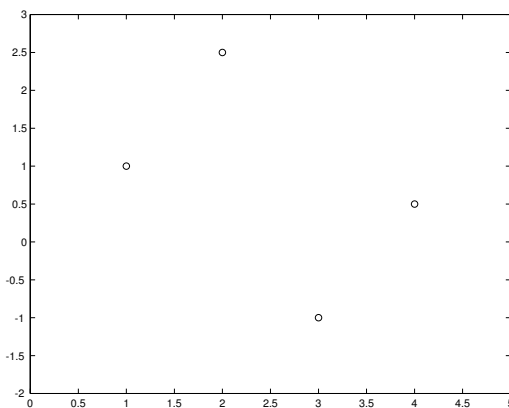
plot(x,cos(x))



plot(x,cos(x),'-',x,sin(x),'-.')

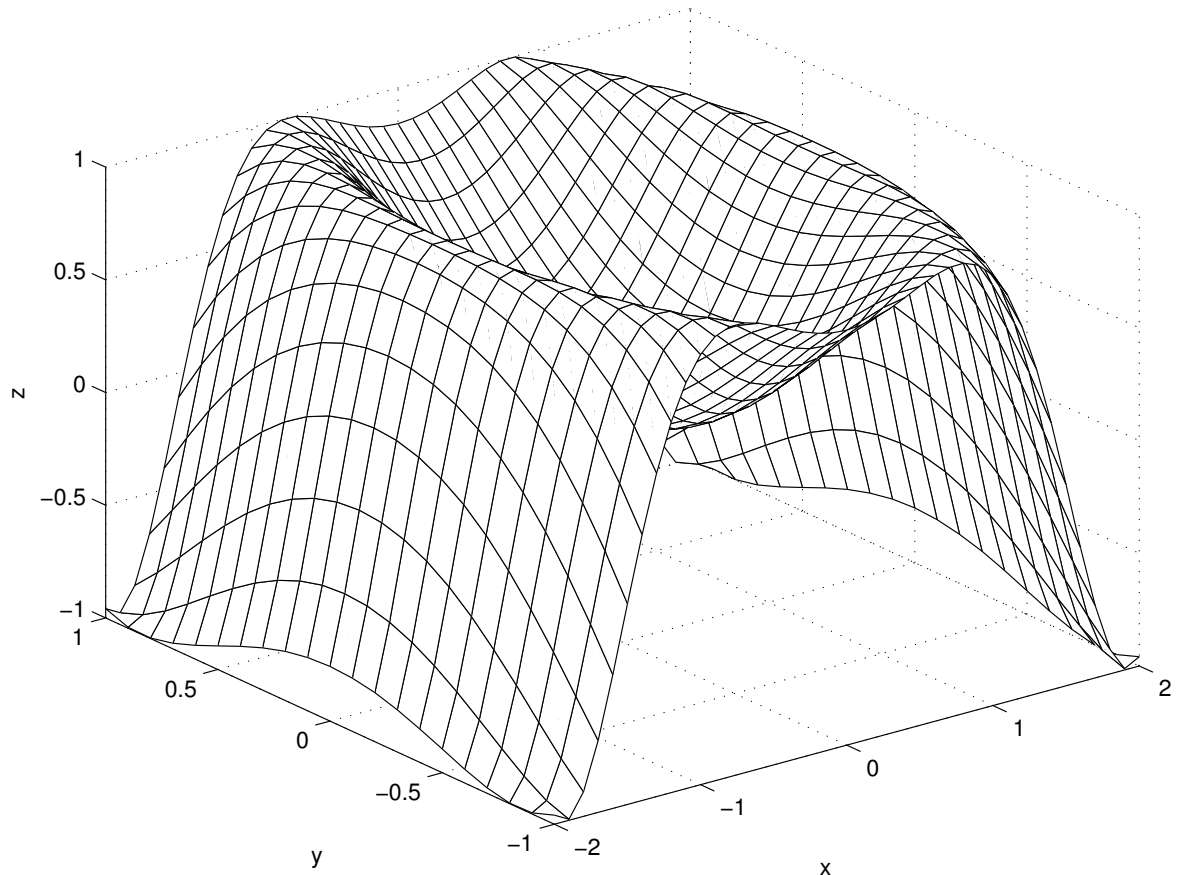


plot(x,y,'o')



1.6 Grafische Darstellungen 3-D

```
>x=-2:0.1:2;
>y=-1:0.1:1;
>[X,Y]=meshgrid(x,y);      % Gitter mit den Punkten (x,y)
>Z=sin(X.*X+Y.*Y);        % Werte von sin(x^2+y^2) auf dem Gitter
>mesh(X,Y,Z)              % zeichnet die Punkte (x,y,z) und verbindet sie
>xlabel('x')
>ylabel('y')
>zlabel('z')
```



1.7 Nullstellen eines Polynoms

Bestimmung der Nullstellen des Polynoms

$$p(x) = x^4 - 2x^2 - 3x - 2.$$

```
>roots([1 0 -2 -3 -2])    % nimmt den Koeffizientenvektor von p als Eingabe
```

ans =

```
    2.0000
   -1.0000
  -0.5000 + 0.8660i
  -0.5000 - 0.8660i
```

2 Programmieren in Matlab

2.1 M-Files

Ein M-File ist eine Textdatei `dateiname.m`, die mehrere Zeilen mit Matlab-Befehlen enthält. Diese Befehle kann man in Matlab «am Stück» ausführen mit

```
>dateiname
```

In M-Files kann man auch Kommentare zu gewissen Zeilen schreiben. Ein Kommentar beginnt mit `%` und endet mit dem Zeilenende.

2.2 for-Schleifen

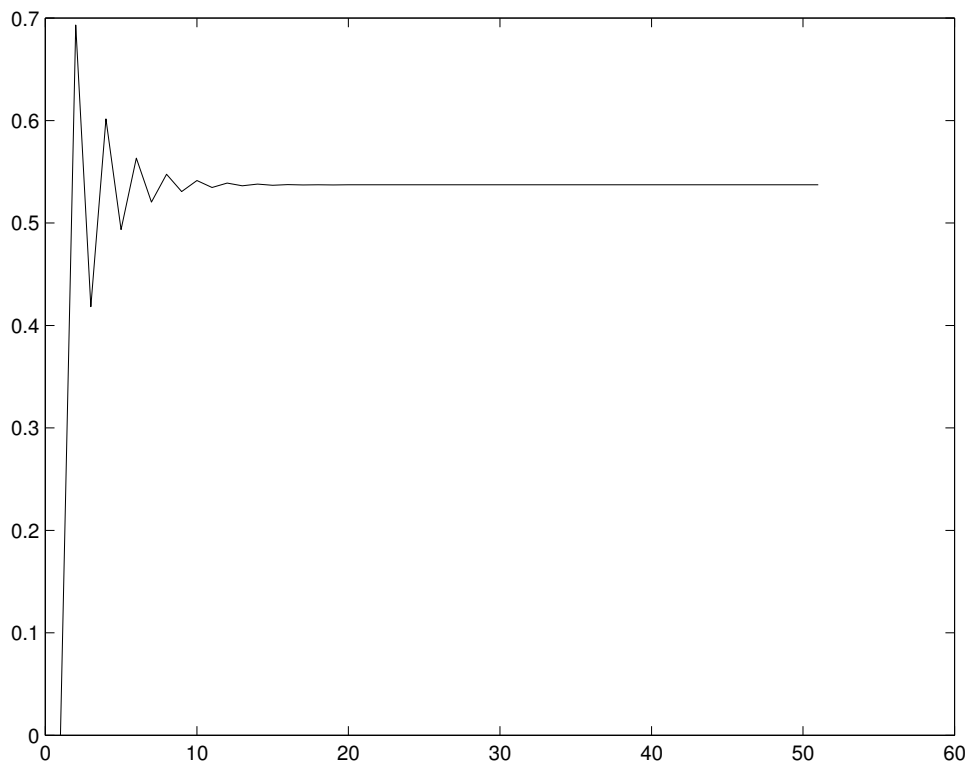
Mit dem Matlab-Befehl `for` realisiert man eine for-Schleife. Als Beispiel soll hier die Fixpunktiteration

$$\begin{aligned}x_1 &= 0 \\x_{k+1} &= \ln(2 - x_k^2), \quad k \geq 1\end{aligned}$$

betrachtet werden. In ein M-File `Fix.m` schreiben wir

```
n=50;  
x(1)=0;  
for k=1:n  
    x(k+1) = log(2-x(k)^2);  
end
```

und starten die Fixpunktiteration mit dem Befehl `Fix`. Wir zeichnen die Werte x_k :



Der Fixpunkt liegt etwa bei 0.5373. Er ist eine Näherungslösung der Gleichung $x^2 + e^x = 2$.

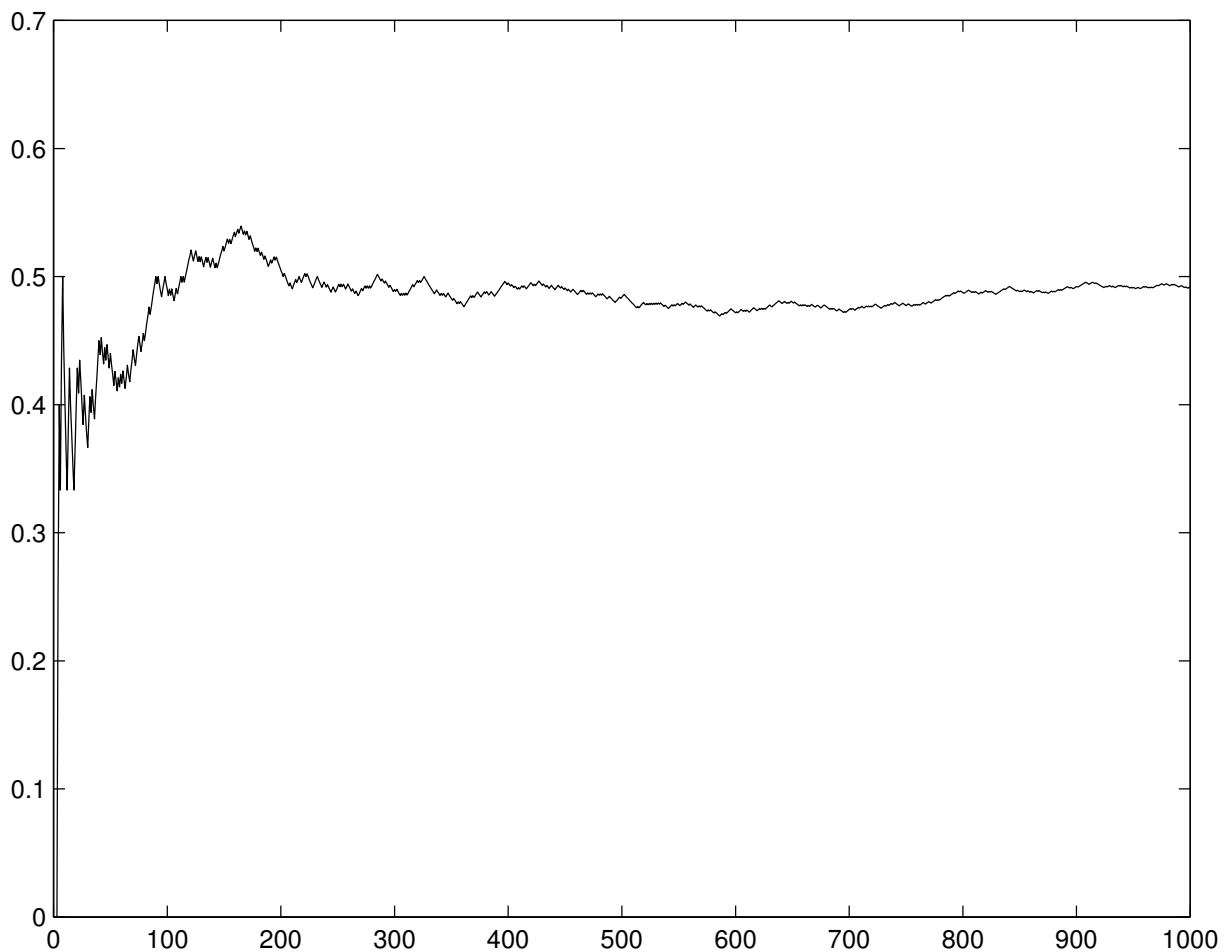
2.3 if-Bedingungen

Mit dem Matlab-Befehl `if` kann man eine if-Bedingung stellen. Als Beispiel soll hier die Simulation von Münzwürfen betrachtet werden. Ein Münzwurf entspricht hier einer Zufallszahl in $[0, 1]$. Ist die Zahl grösser als $1/2$, zählt der Wurf als «Kopf», sonst als «Zahl». Das folgende M-File `Muenzwurf.m` berechnet nach jedem Münzwurf den laufenden Mittelwert

$$\frac{\text{Anzahl «Kopf»}}{\text{Anzahl Würfe}}.$$

```
n=1000;           % Anzahl Muenzwuerfe
x=rand(n,1);     % n Zufallszahlen in [0,1] ("faire" Muenze)
AnzahlKopf=0;    % Zaehler fuer Anzahl Kopf
for k=1:n
    if x(k)>0.5    % Wurf zaehlt als Kopf, wenn die Zufallszahl > 0.5 ist
        AnzahlKopf=AnzahlKopf + 1;
    end
    Mittel(k)=AnzahlKopf/k;
end
```

Wir zeichnen eine Realisierung der Folge `Mittel(k)`:



Der Wert nach 1000 Münzwürfen liegt hier bei 0.4920. Für echte Zufallszahlen ist der Erwartungswert 0.5.

3 Funktionen in Matlab

3.1 Eigene Funktionen definieren

Mit dem Befehl `function` kann man in Matlab eigene Funktionen definieren. Als Beispiel betrachten wir die Funktion $f(t, y) = t + e^y$. Schreibe die folgenden Zeilen in ein M-File:

```
function z = f(t,y)
z = t + exp(y);
```

Speichere die Datei unter dem Namen `f.m`. Es ist wichtig, dass die Datei denselben Namen hat wie die Funktion! Jetzt kannst du den Funktionswert von f an einer beliebigen Stelle (t, y) berechnen mit dem Befehl `f(t,y)`. Die Funktion `f` kann jetzt also wie jede andere Matlab-Funktion (z. B. `sin`) benutzt werden:

```
>f(1,2.5)    % Wert von f an der Stelle (1,2.5)
```

```
ans =
    13.1825
```

```
>t=1;
>s=2.5;
>f(t,s)      % Wert von f an der Stelle (t,s)
```

```
ans =
    13.1825
```

```
>f(s,t)      % Wert von f an der Stelle (s,t)
```

```
ans =
    5.2183
```

Die Reihenfolge der Eingabeargumente ist wesentlich!
Variablen innerhalb der Funktionsdefinition sind nur lokal sichtbar. Die Namen der Variablen in der Funktionsdefinition und beim Aufruf müssen *nicht* übereinstimmen.

3.2 Vordefinierte Matlab-Funktionen benutzen

In Matlab gibt es viele vordefinierte Funktionen für die verschiedensten Aufgaben. Als Beispiel betrachten wir hier die Funktion `ode45`:

Wir betrachten eine Anfangswertaufgabe der Form

$$y' = f(t, y), \quad 0 < t \leq T \quad (1)$$

$$y(0) = y_0. \quad (2)$$

Die Gleichung (1) ist eine gewöhnliche Differenzialgleichung für die Funktion $y(t)$ und (2) ist die Anfangsbedingung bei $t = 0$.

Um eine solche Anfangswertaufgabe mit Matlab numerisch zu lösen, benutzen wir den Befehl `ode45`. Dieser Befehl verwendet ein so genanntes Runge-Kutta-Verfahren mit automatischer Schrittweitensteuerung der Ordnung 4(5). Die Syntax ist

```
[t,y] = ode45('f',[0 T],y0)
```

Dabei muss `f` eine Matlab-Funktion sein, `T` ist der Zeitpunkt, bis zu dem man die Lösung berechnen will, und `y0` ist der Startwert. `ode45` gibt `t` und `y` zurück. Dabei ist `t` der Vektor der Zeitpunkte t_i , $0 = t_1 < t_2 < \dots < t_{N-1} < t_N = T$, und `y` ist der Vektor mit den Werten der numerischen Lösung \bar{y} mit $\bar{y}_i \simeq y(t_i)$. Die numerische Lösung kann man dann mit `plot(t,y)` zeichnen.

Problem

Löse die Anfangswertaufgabe

$$\begin{aligned} y' &= e^y \sin(t), & 0 < t \leq 10 \\ y(0) &= -\ln(3) \end{aligned}$$

numerisch mit `ode45`. Zeichne die numerische Lösung \bar{y} und die exakte Lösung

$$y(t) = -\ln(\cos(t) + 2)$$

in dieselbe Grafik. Zeichne auch den Fehler $y - \bar{y}$.

Lösung

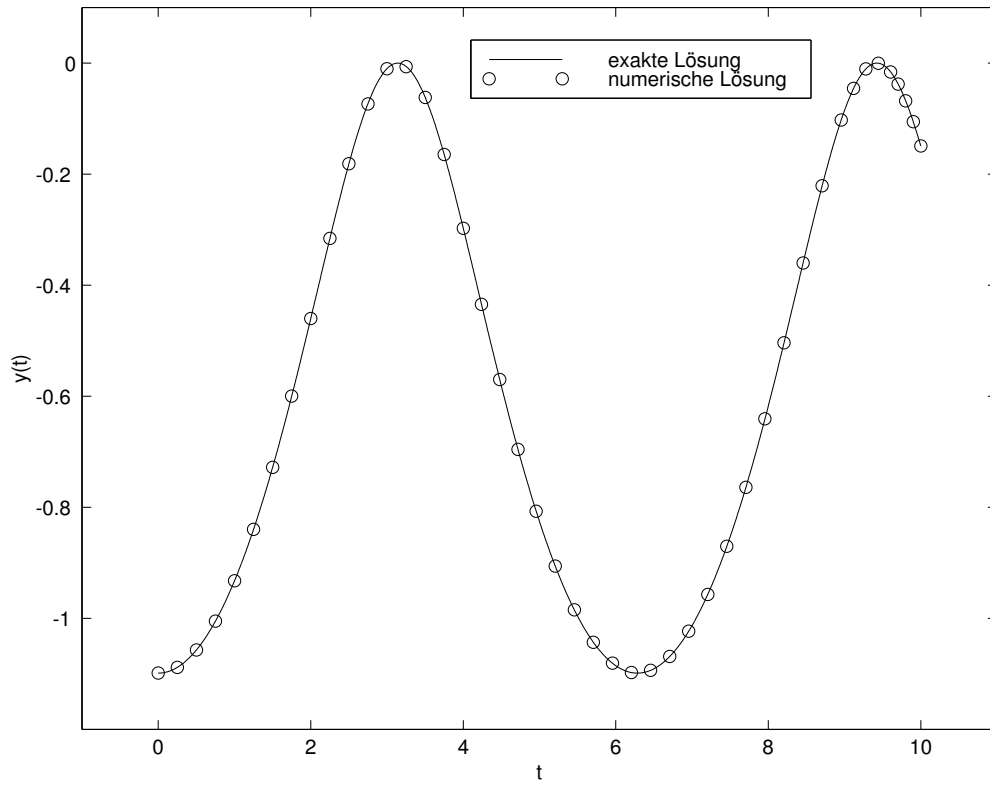
Zunächst definieren wir die Funktion f in einem M-File `f.m`:

```
function z = f(t,y)
z=exp(y)*sin(t);
```

Das M-File `Loesung.m` löst die Anfangswertaufgabe und zeichnet die Lösungen sowie den Fehler gegenüber der exakten Lösung:

```
[t,y]=ode45('f',[0 10],-log(3));
tex=0:0.01:10;           % Zeitpunkte fuer die exakte Loesung
yex=-log(cos(tex)+2);   % Werte der exakten Loesung
plot(tex,yex,'-',t,y,'o')
axis([-1 11 -1.2 0.1])
xlabel('t')
ylabel('y(t)')
legend('exakte Loesung','numerische Loesung')
figure                   % oeffnet ein neues Grafikfenster
plot(t,-log(cos(t)+2)-y,'o-')
axis([-1 11 -1e-3 1e-3])
xlabel('t')
ylabel('Fehler')
```

Die Lösungen y und \bar{y}



Der Fehler $y - \bar{y}$

