Prof. Dr. Stefan Funken
Dipl.-Math. Katharina Becker-Steinberger
Dipl.-Math. oec. Klaus Stolle
Institut für Numerische Mathematik
Universität Ulm

WS 2013/2014
Lösungsblatt 4
22.11.2013

Angewandte Numerik 2

Aufgabe 9 *(Steife Systeme, Explizite vs. Implizite RKV)* (16 Punkte)

Gegeben sind folgende Anfangswertprobleme:

1.
$$y'(t) = \begin{pmatrix} -2 & 1 \\ 1 & -2 \end{pmatrix} y, \quad y(0) = \begin{pmatrix} 2 \\ 3 \end{pmatrix}$$

2.
$$y'(t) = \begin{pmatrix} -2 & 1 \\ 998 & -999 \end{pmatrix} y, \quad y(0) = \begin{pmatrix} 2 \\ 3 \end{pmatrix}$$

a) Lösen Sie beide Anfangswertprobleme analytisch. Berechnen Sie hierzu jeweils die Eigenwerte und Eigenvektoren der Koeffizientenmatrix und diskutieren Sie ausführlich Ihre Ergebnisse!

b) Lösen Sie beide Anfangswertprobleme für $t \in [0, 10]$ und einer vorgegebenen relativen Toleranz von 0.01 jeweils mit der MATLAB-Routine `ode45` und anschliessend mit `ode23tb`. Informieren Sie sich über beide Routinen und diskutieren Sie die Ergebnisse bzgl. Anzahl der benötigten Schritte / Schrittweiten und nehmen Sie Bezug auf ihre analytischen Ergebnisse aus Teil a)!

c) Programmieren Sie das 3-stufuge implizite Runge-Kutta-Verfahren mit dem Koeffizientenschema

$$\begin{array}{c|ccc} 0 & \frac{1}{6} & -\frac{1}{6} & 0 \\ \frac{1}{2} & \frac{1}{6} & \frac{1}{3} & 0 \\ 1 & \frac{1}{6} & \frac{5}{6} & 0 \\ \hline & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \end{array}$$

Verwenden Sie zur Lösung des impliziten Gleichungssystems das einfache Fixpunktverfahren mit $M \in \mathbb{N}$ Iterationen und Startwert 0. Versuchen Sie dabai, die spezielle Struktur des Verfahrens (letzte Spalte von B verschwindet!) möglichst gut auszunutzen. Testen Sie Ihr Programm an der Anfangswertaufgabe

$$y'(t) = -2ty(t)^2, \quad y(0) = 1.$$

Dieses hat die exakte Lösung

$$y(t) = \frac{1}{t^2 + 1}.$$

Berechnen Sie dann für $M = 1, 2, 3, 4, 5$ jeweils mit den Schrittweiten $h := \frac{1}{N}, N \in \{10, 20, 40, 80, 160, 320\}$ die Fehler $e_N = |y_N - y(1)|$ an der Stelle $t = 1$ sowie die numerische Konvergenzordnung

$$\frac{\ln\left(\frac{e_{N/2}}{e_N}\right)}{\ln(2)}, \quad N \neq 10.$$

Was Beobachten Sie?

d) Testen Sie Ihre Routine aus c) auch an den oben gebebenen Anfangswertaufgaben. Wie muss die Schrittweite h gewählt werden damit die Fixpunktiteration konvergiert?

Lösung:

a) Um eine homogene Fundamentallösung zu bestimmen

1.

$$P(\lambda) = (-2-\lambda)^2 - 1 = 0$$

Daraus folgt $\lambda_{1/2} = \pm 1 + 2$ d.h. $\lambda_1 = -3$ und $\lambda_2 = -1$. Die Eigenvektoren lauten

$$v^1 = \begin{pmatrix} 0.7071 \\ -0.7071 \end{pmatrix}, \quad v^2 = \begin{pmatrix} 0.7071 \\ 0.7071 \end{pmatrix}$$

Für die allgemeine homogene Lösung

$$y_h(t) = \sum_{k=1}^{2} e^{\lambda_k t} v^k$$

müssen noch die Konstanten $c_k, k = 1, 2$ bestimmt werden. Mit dem Startwert erhalten wir das folgende Gleichungssystem

$$c_1 e^0 \begin{pmatrix} 0.7071 \\ -0.7071 \end{pmatrix} + c_2 e^0 \begin{pmatrix} 0.7071 \\ 0.7071 \end{pmatrix} = \begin{pmatrix} 2 \\ 3 \end{pmatrix}.$$

Lösen dieses Gleichungssystems liefert $c_1 = -0.7071$, $c_2 = 3.5356$. Damit gilt

$$y_h(t) = -0.7071 e^{-3t} \begin{pmatrix} 0.7071 \\ -0.7071 \end{pmatrix} + 3.5356 e^{-1t} \begin{pmatrix} 0.7071 \\ 0.7071 \end{pmatrix}.$$

2.

$$\begin{aligned} & P(\lambda) = (-2-\lambda)(-999-\lambda) - 998 = \lambda^2 + 1001\lambda + 1000 = 0 \\ \iff & \left(\lambda + \frac{1001}{2}\right)^2 + 1000 - \frac{(1001^2)}{4} = 0 \\ \iff & \left(\lambda + \frac{1001}{2}\right)^2 = \frac{998001}{4} \\ \implies & \left(\lambda + \frac{1001}{2}\right) = \pm \frac{999}{2} \end{aligned}$$

Also $\lambda_1 = -1$ und $\lambda_2 = --1000$ mit den Eigenvektoren

$$v^1 = \begin{pmatrix} 0.7071 \\ 0.7071 \end{pmatrix} \quad v^2 = \begin{pmatrix} -0.001 \\ 0.999 \end{pmatrix}$$

Für die allgemeine homogene Lösung

$$y_h(t) = \sum_{k=1}^{2} e^{\lambda_k t} v^k$$

müssen jetzt noch wie zuvor die Konstanten $c_k, k = 1, 2$ bestimmt werden.

$$c_1 e^0 \begin{pmatrix} 0.7071 \\ 0.7071 \end{pmatrix} + c_2 \begin{pmatrix} -0.001 \\ 0.999 \end{pmatrix} = \begin{pmatrix} 2 \\ 3 \end{pmatrix}$$

Lösen dieses Gleichungssystems liefert $c_1 = 2.829$, $c_2 = 0.9989$. Damit gilt

$$y_h(t) = 2.829 e^{-1t} \begin{pmatrix} 0.7071 \\ 0.7071 \end{pmatrix} + 0.9989 e^{-1000t} \begin{pmatrix} -0.001 \\ 0.999 \end{pmatrix}$$

Die Eigenwerte der Koeffizientenmatrix von Beispiel 1 haben dieselbe Größenordnung, während die Eigenwerte der Koeffizientenmatrix von Beispiel 2 weisen sehr unterschiedliche Größenordnung auf. Beispiel 2 ist ein steifes Problem. Der Lösungsanteil y_2 von Beispiel 2 wird sehr schnell klein (vgl. graphische Darstellung der exakten Lösung). Dies wirkt sich wie man unten sieht auf die numerische Lösung aus. Das numerische Verfahren `ODE45` richtet aus Stabilitätsgründen die Schrittweite nach dem schnellen Anteil aus und benötigt daher unverhältnismäßig viele Integrationsschritte.

b)

```
function yH = yH_exakt(V, D, c, t)
% Die Funktion yH_exakt berechnet die exakte Loesung eines
% linearen homogenen AWP 1.Ordnung mit konstanter Koeffizientenmatrix:
%
% y'(t) = A*y(t)
%
% mit y einem n dimensionalen Vektor, A nxn Matrix
% Input:  V  nxn Matrix, Spalten sind EV von A
%         D  nxn Matrix, Diadonaleintaege sind EW von A
%         c  n dimensionaler Konstanten-Vektor (wird durch AW bestimmt)
%         t  Vektor mit Zeitschritten
% Output: yH exakte Loesung

% Initialisiere yH
yH = zeros(size(t,1),1);

% Schleife ueber Dimension
for i = 1 : size(V,2)
    yH = yH + (c(i) * V(:,i) * exp(D(i,i) * t));
end
end
```

```

function dydt = odeH(t,y,A)

dydt = A * y;

end
```

```
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
% Aufgabe 9a):
%
%--- Analytische Bestimmung der homogenen Loesung ------------------------

clear all, close all

fprintf('\n*********************************************************\n');
fprintf('\nAufgabe 9a: Analytische Bestimmung der homogenen Loesung\n');
fprintf('\n*********************************************************\n');

% Koeffizientenmatrizen
A1 = [-2  1;   1    -2];
A2 = [-2  1; 998  -999];

% Zur Bestimmung der Fundamentalloesung der beiden ODE Systeme 1.Ordnung,
% berechnen wir die Eigenwerte und Eigenvektoren von A1 und A2.
```

```
19
20 fprintf('\nBsp1: Eigenvektoren und Eigenwerte\n')
21 [V1,D1] = eig(A1)
22 fprintf('\nBsp2: Eigenvektoren und Eigenwerte\n')
23 [V2,D2] = eig(A2)
24
25 % Anfangswert
26 y0 = [2; 3];
27
28 % Um die Konstanten der homogenen Loesung auf den Anfangswert anzupassen,
29 % loesen wir jeweils das LGS V*c = y0
30 fprintf('\nBsp1: Konstanten\n')
31 c1 = V1\y0
32 fprintf('\nBsp2: Konstanten\n')
33 c2 = V2\y0
34
35 % Berechne exakte Loesung der homogenen linearen ODE
36 ht = linspace(0,10, 25);
37 y1H = yH_exakt(V1, D1, c1, ht);
38 y2H = yH_exakt(V2, D2, c2, ht);
39
40 %-------------------------------------------------------------------------
41 %-------------------------------------------------------------------------
42 % Aufgabe 9b):
43 %
44 %--- Numerische Bestimmung der Loesung ---------------------------------
45
46 fprintf('\n*******************************************************\n');
47 fprintf('\nAufgabe 9b: Explizites vs. implizites RKV\n');
48 fprintf('\n*******************************************************\n');
49
50
51 % Berechne zeitabhaengige Inhomogenitaet
52 h11 = 2*sin(ht);
53 h12 = 2*(cos(ht)-sin(ht));
54 h21 = 2*sin(ht);
55 h22 = 999*(cos(ht)-sin(ht));
56
57 % Setze relative Toleranz
58 options = odeset('RelTol', 1e-2);
59
60 % Setze Zeitintervall
61 Tspan = [0 10]; % Loese von t=0 bis t=10
62
63 %--- Loese homogenen linearen ODE --------------------------------------
64
65 % mit RKV 4(5)
66 [T1_ode45H Y1_ode45H] = ode45(@(t,y) odeH(t,y,A1), Tspan, y0, options);
67 [T2_ode45H Y2_ode45H] = ode45(@(t,y) odeH(t,y,A2), Tspan, y0, options);
68
69 % mit implizitem RKV
70 [T1_ode23tbH Y1_ode23tbH] = ode23tb(@(t,y) odeH(t,y,A1), Tspan, y0, options);
71 [T2_ode23tbH Y2_ode23tbH] = ode23tb(@(t,y) odeH(t,y,A2), Tspan, y0, options);
72
```

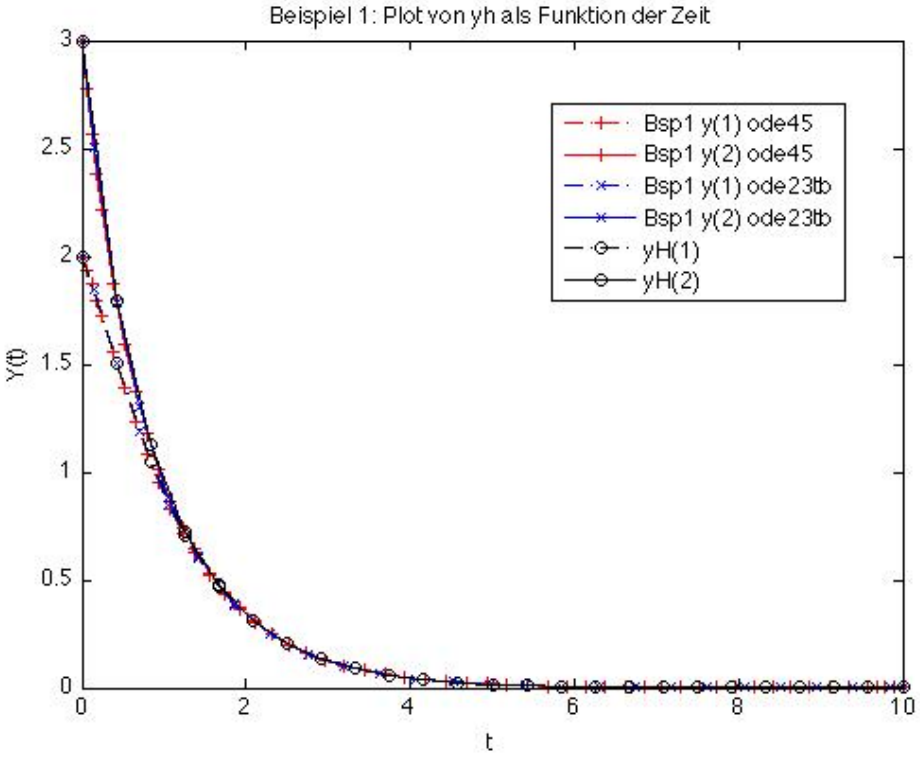
```
73  % Anzahl der benoetigten Schritte
74  iterT1_45H = length(T1_ode45H);
75  iterT2_45H = length(T2_ode45H);
76  iterT1_23H = length(T1_ode23tbH);
77  iterT2_23H = length(T2_ode23tbH);
78
79  % minimale Schrittweite
80  h1_ode45Hmin = min(T1_ode45H(2:end) - T1_ode45H(1:end-1));
81  h2_ode45Hmin = min(T2_ode45H(2:end) - T2_ode45H(1:end-1));
82  h1_ode23tbHmin = min(T1_ode23tbH(2:end) - T1_ode23tbH(1:end-1));
83  h2_ode23tbHmin = min(T2_ode23tbH(2:end) - T2_ode23tbH(1:end-1));
84
85  % maximale Schrittweite
86  h1_ode45Hmax = max(T1_ode45H(2:end) - T1_ode45H(1:end-1));
87  h2_ode45Hmax = max(T2_ode45H(2:end) - T2_ode45H(1:end-1));
88  h1_ode23tbHmax = max(T1_ode23tbH(2:end) - T1_ode23tbH(1:end-1));
89  h2_ode23tbHmax = max(T2_ode23tbH(2:end) - T2_ode23tbH(1:end-1));
90
91  % mittlere Schrittweite
92  h1_ode45H = sum(T1_ode45H(2:end) - T1_ode45H(1:end-1))/iterT1_45H;
93  h2_ode45H = sum(T2_ode45H(2:end) - T2_ode45H(1:end-1))/iterT2_45H;
94  h1_ode23tbH = sum(T1_ode23tbH(2:end) - T1_ode23tbH(1:end-1))/iterT1_23H;
95  h2_ode23tbH = sum(T2_ode23tbH(2:end) - T2_ode23tbH(1:end-1))/iterT2_23H;
96
97  % Tabellarische Ausgabe der Anzahl der benoetigten Schritte
98
99  fprintf('\nAnzahl␣der␣benoetigten␣Schritte,␣minimale,␣maximale␣und')
100 fprintf('\nmittlere␣Schrittweiten:\n')
101
102 fprintf('\nBsp1␣␣␣␣␣␣␣␣␣␣ODE45␣␣␣␣␣␣␣␣␣␣␣␣␣ODE23tb␣␣\n')
103 fprintf('␣␣␣␣␣␣␣iter␣␣␣␣␣␣␣␣␣␣␣%2.0f␣␣␣␣␣␣␣␣␣␣␣␣␣␣%2.0f␣\n', ...
104     iterT1_45H, iterT1_23H)
105 fprintf('␣␣␣␣␣␣␣hmin␣␣␣%2.6e␣␣␣␣␣%2.6e␣\n', h1_ode45Hmin, h1_ode23tbHmin)
106 fprintf('␣␣␣␣␣␣␣hmax␣␣␣%2.6e␣␣␣␣␣%2.6e␣\n', h1_ode45Hmax, h1_ode23tbHmax)
107 fprintf('␣␣␣␣␣␣␣hmean␣␣%2.6e␣␣␣␣␣%2.6e␣\n', h1_ode45H, h1_ode23tbH)
108
109 fprintf('\nBsp2␣␣␣␣␣␣␣␣␣␣ODE45␣␣␣␣␣␣␣␣␣␣␣␣␣ODE23tb␣␣\n')
110 fprintf('␣␣␣␣␣␣␣iter␣␣␣␣␣␣␣␣␣%6.0f␣␣␣␣␣␣␣␣␣␣␣␣␣␣%2.0f␣\n', ...
111     iterT2_45H, iterT2_23H)
112 fprintf('␣␣␣␣␣␣␣hmin␣␣␣%2.6e␣␣␣␣␣%2.6e␣\n', h2_ode45Hmin, h2_ode23tbHmin)
113 fprintf('␣␣␣␣␣␣␣hmax␣␣␣%2.6e␣␣␣␣␣%2.6e␣\n', h2_ode45Hmax, h2_ode23tbHmax)
114 fprintf('␣␣␣␣␣␣␣hmean␣␣%2.6e␣␣␣␣␣%2.6e␣\n', h2_ode45H, h2_ode23tbH)
115
116 %--- Grafische Ausgabe der Loesung ------------------------------------------
117
118 % Plot der numerischen und exakten homogenen Loesung von Beispiel 1
119 figure(1)
120 plot(T1_ode45H, Y1_ode45H(:, 1), '+-.r',  ...
121      T1_ode45H, Y1_ode45H(:, 2), '+-r', ...
122      T1_ode23tbH, Y1_ode23tbH (:,1), 'x-.b', ...
123      T1_ode23tbH, Y1_ode23tbH (:,2), 'x-b', ...
124      ht, y1H(1,:), 'o-.k', ht, y1H(2,:), 'o-k' );
125 legend('Bsp1␣y(1)␣ode45', 'Bsp1␣y(2)␣ode45', ...
126        'Bsp1␣y(1)␣ode23tb', 'Bsp1␣y(2)␣ode23tb', ...
```

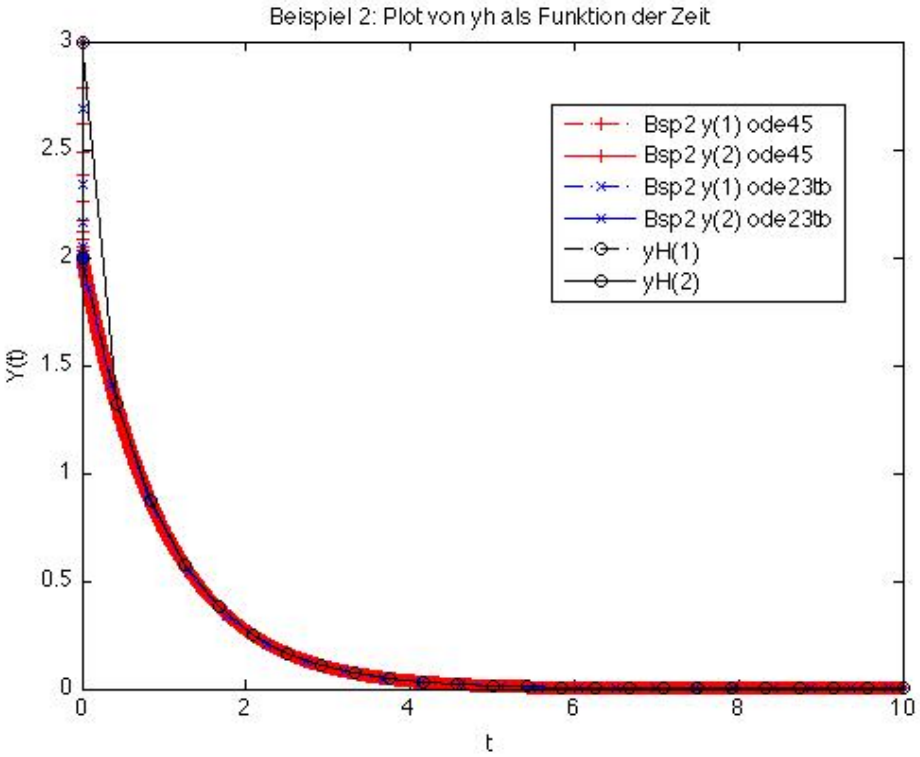
```
127         'yH(1)', 'yH(2)','Location', 'Best')
128 title('Beispiel 1: Plot von yh als Funktion der Zeit');
129 xlabel('t'); ylabel('Y(t)');
130
131 % Plot der numerischen und exakten homogenen Loesung von Beispiel 2
132 figure(2)
133 plot(T2_ode45H, Y2_ode45H(:, 1), '+-.r', ...
134      T2_ode45H, Y2_ode45H(:, 2), '+-r', ...
135      T2_ode23tbH, Y2_ode23tbH (:,1), 'x-.b', ...
136      T2_ode23tbH, Y2_ode23tbH (:,2), 'x-b', ...
137      ht, y2H(1,:), 'o-.k', ht, y2H(2,:), 'o-k' );
138 legend('Bsp2 y(1) ode45', 'Bsp2 y(2) ode45', ...
139         'Bsp2 y(1) ode23tb', 'Bsp2 y(2) ode23tb', ...
140         'yH(1)', 'yH(2)','Location', 'Best')
141 title('Beispiel 2: Plot von yh als Funktion der Zeit');
142 xlabel('t'); ylabel('Y(t)');
```

```
>> Aufgabe9

************************************************************

Aufgabe 9a: Analytische Bestimmung der homogenen Loesung

************************************************************

Bsp1: Eigenvektoren und Eigenwerte

V1 =

   0.707106781186547   0.707106781186547
  -0.707106781186547   0.707106781186547


D1 =

    -3     0
     0    -1


Bsp2: Eigenvektoren und Eigenwerte

V2 =

   0.707106781186547  -0.001002003505004
   0.707106781186547   0.999999497994362


D2 =

          -1            0
           0        -1000


Bsp1: Konstanten
```

```
c1 =

  -0.707106781186548
   3.535533905932738


Bsp2: Konstanten

c2 =

   2.829842753937755
   0.998999500502381


************************************************************

Aufgabe 9b: Explizites vs. implizites RKV

************************************************************

Anzahl der benoetigten Schritte, minimale, maximale und
mittlere Schrittweiten:

Bsp1            ODE45              ODE23tb
        iter              53                   26
        hmin    5.971608e-02       1.394838e-01
        hmax    2.500000e-01       4.432869e-01
        hmean   1.886792e-01       3.846154e-01

Bsp2            ODE45              ODE23tb
        iter           12073                   32
        hmin    2.386257e-04       3.692240e-04
        hmax    1.020852e-03       4.744472e-01
        hmean   8.282945e-04       3.125000e-01
```

Grafische Ausgabe der numerischen Lösung:


Beispiel 1: Plot von yh als Funktion der Zeit
Bsp1 y(1) ode45
Bsp1 y(2) ode45
Bsp1 y(1) ode23tb
Bsp1 y(2) ode23tb
yH(1)
yH(2)
Y(t)
t
0
0.5
1
1.5
2
2.5
3
0
2
4
6
8
10


Beispiel 2: Plot von yh als Funktion der Zeit
Bsp2 y(1) ode45
Bsp2 y(2) ode45
Bsp2 y(1) ode23tb
Bsp2 y(2) ode23tb
yH(1)
yH(2)
Y(t)
t
0
0.5
1
1.5
2
2.5
3
0
2
4
6
8
10

c)

```
function [y,t]=rungekuttaImplicit(f,ta,ya,te,N,M)
 h=(te-ta)/N;
 h6=h/6;
 t = ta:h:te; % Gitter anlegen
 y=zeros(length(ya),N+1);
 y(:,1) = ya; % Startwert schreiben

 for i=1:N % durchlaufe Gitterpunkte
    %setze Startwerte fuer Iterationsverfahren
    k1=zeros(length(ya),1);
    k2=zeros(length(ya),1);
    %starte Iterationsverfahren
    for j=1:M
      k1alt=k1;  % alte Werte merken fuer Iteration
      k2alt=k2;
      k1=f(t(i),y(:,i)+h6*(k1alt-k2alt));
      k2=f(t(i)+h/2,y(:,i)+h6*(k1alt+2*k2alt));
    end
    k3=f(t(i)+h,y(:,i)+h6*(k1+5*k2));
    y(:,i+1)=y(:,i)+h6*(k1+4*k2+k3);
 end
end
```

```
function loglogTriangle(x1,x2,y1,slope,type)
hold on
slope=-slope;
scale=y1/(x1^slope);
loglog([x1 x2],[x1 x2].^slope*scale,'k');
if type=='l'
  loglog([x1 x2],[x2 x2].^slope*scale,'k');
  loglog([x1 x1],[x1 x2].^slope*scale,'k');
  text(sqrt(x1*x2),(x1^(.1)*x2^(.91))^slope*scale,'1');
  text(x1^(.87)*x2^(.05),sqrt(x1*x2)^slope*scale,rats(-slope));

elseif type=='u'
  loglog([x1 x2],[x1 x1].^slope*scale,'k');
  loglog([x2 x2],[x1 x2].^slope*scale,'k');
  text(sqrt(x1*x2),(x1^(1)*x2^(.05))^slope*scale,'1');
  text(x1^(.09)*x2^(.865),sqrt(x1*x2)^slope*scale,rats(-slope));
end
```

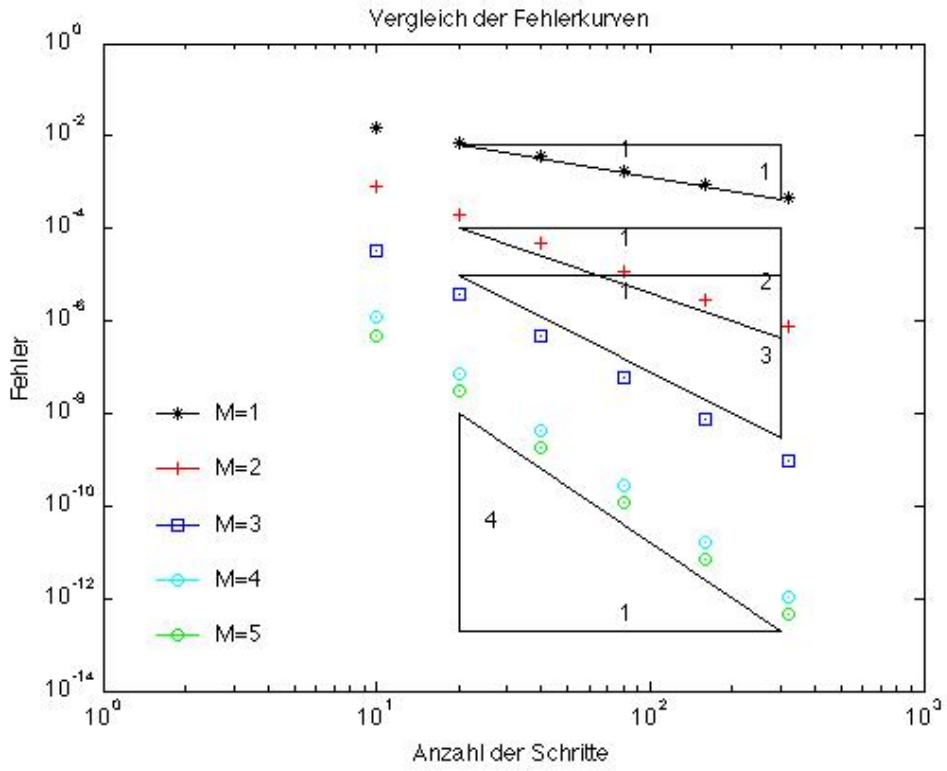
```
function bwLegend(X1,Y1,Y2,art,abc)
plot(10,10^(-1),'w.')
for i=1:size(art,1)
  y=10^(Y1+(i-1)*(Y2-Y1)/size(art,1));
  x1=10^X1;
  x2=10^(X1+.15);
  x3=10^(X1+.075);
  x4=10^(X1+.22);
  plot([x1 x2],[y y],'w')
  plot([x1 x2],[y y],[art(i,1),art(i,3:4)]);plot(x3,y,art(i,:));text(x4,y,abc(i,:
end
```

```
clear all;
close all;

%-------------------------------------------------------------------------
% Aufgabe 9c
%-------------------------------------------------------------------------

f = @(t,y)-2*t*y(1)*y(1); % Zu loesende ODE
y_exakt = @(t) 1/(t^2+1); % exakte Loesung
ta = 0;
te = 1;
ya = 1;

Nstart = 10;
step = 5; %Nstart*2^step
M = 5;

%fuer Fehlerplot
art = ['k*-';'r+-'; 'bs-';'co-';'go-'; 'mx-'; 'c>-'; 'r<-'; 'g^--';...
      'bv--'; 'co--'; 'm+--';'r*:'; 'gx:'; 'bs:'; 'cd:'; 'mo:'];

yex = y_exakt(te); % exakte Loesung an der Stelle te=1

for MM = 1:M
    NN = Nstart;
    fprintf('\n\nFuer M=%d ergibt sich:',MM);
    fprintf('\n  N      h           ||yh(1)-y(1)||   Konv.ord.');
    fprintf('\n-------------------------------------------------');
    for i = 0:step
        [yN,tN] = rungekuttaImplicit(f,ta,ya,te,NN,MM);
        eN = norm(yN(:,end)-yex);
        %Berechne Konvergenzordnung
        if(i>0)
          konvord = log(eNalt/eN)/log(2);
          fprintf('\n%3d %9.6f %18.10e %11.6f',NN,1/NN,eN,konvord)
        else %erste Zeile
          fprintf('\n%3d %9.6f %18.10e',NN,1/NN,eN);
        end
        loglog(NN,eN,art(MM,:));
        hold on;
        eNalt = eN;
        NN = 2*NN;
    end
    fprintf('\n');
    fprintf('\nExakte Loesung: %18.10e',yex);
    fprintf('\nyh(1)         : %18.10e',yN(:,end));
end
fprintf('\n');

title('Vergleich der Fehlerkurven');
bwLegend(0.2,-8,-14,art(1:5,:),['M=1'; 'M=2'; 'M=3'; 'M=4'; 'M=5']);
loglogTriangle(20,300,10^(-2.2), 1,'u');
loglogTriangle(20,300,10^(-4), 2,'u');
```

```
54 loglogTriangle(20,300,10^(-5), 3,'u');
55 loglogTriangle(20,300,10^(-8), 4,'l');
56 xlabel('Anzahl der Schritte');
57 ylabel('Fehler');
58 hold off;
```

```
Fuer M=1 ergibt sich:
  N       h          ||yh(1)-y(1)||    Konv.ord.
-------------------------------------------------
 10   0.100000    1.4816236850e-02
 20   0.050000    7.2761637875e-03     1.025929
 40   0.025000    3.6068737133e-03     1.012429
 80   0.012500    1.7958314127e-03     1.006097
160   0.006250    8.9603752903e-04     1.003021
320   0.003125    4.4755203762e-04     1.001504

Exakte Loesung:    5.0000000000e-01
yh(1)          :   4.9955244796e-01

Fuer M=2 ergibt sich:
  N       h          ||yh(1)-y(1)||    Konv.ord.
-------------------------------------------------
 10   0.100000    8.2104904699e-04
 20   0.050000    1.9639846882e-04     2.063685
 40   0.025000    4.8013566128e-05     2.032270
 80   0.012500    1.1869288954e-05     2.016209
160   0.006250    2.9506698503e-06     2.008119
320   0.003125    7.3559301228e-07     2.004063

Exakte Loesung:    5.0000000000e-01
yh(1)          :   5.0000073559e-01

Fuer M=3 ergibt sich:
  N       h          ||yh(1)-y(1)||    Konv.ord.
-------------------------------------------------
 10   0.100000    3.2649889401e-05
 20   0.050000    3.8949080370e-06     3.067417
 40   0.025000    4.7529195896e-07     3.034703
 80   0.012500    5.8691617733e-08     3.017588
160   0.006250    7.2915807836e-09     3.008851
320   0.003125    9.0864749058e-10     3.004439

Exakte Loesung:    5.0000000000e-01
yh(1)          :   4.9999999909e-01

Fuer M=4 ergibt sich:
  N       h          ||yh(1)-y(1)||    Konv.ord.
-------------------------------------------------
 10   0.100000    1.2292804553e-06
 20   0.050000    7.2000532647e-08     4.093663
 40   0.025000    4.3572737640e-09     4.046510
 80   0.012500    2.6799085173e-10     4.023170
160   0.006250    1.6615375742e-11     4.011593
```

```
320   0.003125    1.0340617251e-12     4.006125

Exakte Loesung:     5.0000000000e-01
yh(1)          :    5.0000000000e-01

Fuer M=5 ergibt sich:
  N       h           ||yh(1)-y(1)||     Konv.ord.
-------------------------------------------------------
 10   0.100000    4.8761516824e-07
 20   0.050000    2.9785292011e-08     4.033071
 40   0.025000    1.8360439835e-09     4.019928
 80   0.012500    1.1390477450e-10     4.010700
160   0.006250    7.0916605921e-12     4.005561
320   0.003125    4.4197978610e-13     4.004071

Exakte Loesung:     5.0000000000e-01
yh(1)          :    5.0000000000e-01
```



d)

Aufgabe 10 *(Mehrschrittverfahren)* (12 Punkte)

a) Schreiben Sie eine Maple-Routine, welche

$$\beta_i^{(r,\ell,k)} := \int_{-\ell}^{k-r} \prod_{\substack{p=0 \\ p\neq i}} \frac{r-p+\tilde{s}}{i-p}\, d\tilde{s}$$

analytisch berechnet.

b) Verifizieren Sie diese Routine für das Adams-Moulton-Verfahren ($\ell = 1$) mit $r = k = 2$.

c) Bestimmen Sie mit der Routine die Koeffizienten für das Adams-Moulton-Verfahren ($\ell = 1$) mit $r = k = 3$ und das Adams-Bashforth-Verfahren ($\ell = 0$) mit $r = k - 1 = 3$.

d) Testen Sie die in c) berechneten Verfahren numerisch indem Sie zwei Matlab-Funktionen schreiben, die die folgenden Anfangswertaufgabe

$$y' = -\frac{2xy^2}{x^2+1}, \quad y(0) = 2$$

lösen. Die Schrittweite sei hierbei $h = 0.1$. Die exakte Lösung ist übrigens

$$y(x) = \frac{1}{\ln(x^2+1) + 0.5}.$$

Verwenden Sie als Startwerte die exakte Lösung an den Gitterpunkten, d.h. $y_j = y(x_j)$ für $j = 0, \ldots, k-1$. Plotten Sie jeweils den Fehler gegen die exakte Lösung.

Lösung:

a)

```
restart;
calc_beta := proc(k,r,ell,i)
  int(product((r-p+s)/(i-p),p=0..i-1) * product((r-p+s)/(i-p),p=i+1..r),s = -ell
end proc:
```

```
k:=2;
r:=2;
ell:=1;

seq(calc_beta(k,r,ell,i),i=0..k);
```

liefert

$$-\frac{1}{12}, \frac{2}{3}, \frac{5}{12}$$

b) Adams-Bashforth-Verfahren mit $k - 1 = r = 3, \ell = 0$:

```
k:=4;
r:=3;
ell:=0;

seq(calc_beta(k,r,ell,i),i=0..k);
```

liefert

$$-\frac{3}{8}, \frac{37}{24}, \frac{-59}{24}, \frac{55}{24}$$

Adams-Moulton-Verfahren mit $k = r = 3, \ell = 0$

```
k:=3;
r:=3;
ell:=1;

seq(calc_beta(k,r,ell,i),i=0..k);
```

liefert

$$\frac{1}{24}, -\frac{5}{24}, \frac{19}{24}, \frac{3}{8}$$

c)

```
function dy = f10(t,y )
dy = -2*t*y*y/(t*t+1);
end
```

```
function y_exakt = f10exakt(t)
y_exakt = 1./(log (t.^2+1)+0.5);
end
```

Als Startwerte werden die exakten Lösungen genommen:

```
function [y] = adams_bashforth_exakt(t_0, y_0, t_b, f, N, exakt)
% Die Funktion loest eine gegebene gewoehnliche Differentialgleichung (ODE)
% y'=f(t,y)
% mit dem N-Schrittverfahren von Adams-Bashforth
%
% INPUT:  t_0       Anfangszeitpunkt
%         y_0       Startwert y(t_0)
%         f         rechte Seite der ODE
%         N         Anzahl der Schritte
%         exakt     exakte Loesung der ODE
% OUTPUT: y         Naeherung an y nach N Schritten
%

h =(t_b-t_0)/N;
t = t_0;
y = zeros(length(y_0), N+1);
y(:,1) = y_0(:)';
fi(:,1) = f(t_0, y_0);

% Startrechnung
for i = 1:3
    % Berechne exakte Werte
    y_start(i+1) = exakt(t+i*h);
    y(:,i+1) = y_start(i+1);
    fi(:,i+1) = f(t_0+i*h, y(:,i+1));
end

for i=4:N
    y(:, i+1) = y(:,i)+h/24*(55*fi(:,i)-59*fi(:,i-1)+37*fi(:,i-2)-9* fi(:, i-3));
    fi(:,i+1) = f(t_0+i*h, y(:, i+1));
end
end
```

```
function [ y ] = adams_moulton_exakt(t_0,y_0,t_b,f,N,exakt)
% Die Funktion loest eine gegebene gewoehnliche Differentialgleichung (ODE)
% y'=f(t,y)
% mit dem N-Schrittverfahren von Adams-Moulton
%
% INPUT:  t_0       Anfangszeitpunkt
%         y_0       Startwert y(t_0)
%         f         rechte Seite der ODE
%         N         Anzahl der Schritte
%         exakt   exakte Loesung der ODE
% OUTPUT: y         Naeherung an y nach N Schritten
```

```
12
13 h =(t_b-t_0)/N;
14 t = t_0;
15 y = zeros(length(y_0), N+1);
16 y(:,1) = y_0(:)';
17 fi(:,1) = f(t_0, y_0);
18
19 % Startrechnung
20 for i = 1:3
21     % Berechne exakte Werte
22     y(:,i+1) = exakt(t+i*h);
23     fi(:,i+1) = f(t_0+i*h, y(:,i+1));
24 end
25
26 % Praediktor-Korrektor-Schritt
27 for i=4:N
28     % Praediktor
29     yp = y(:,i)+h/24*(55*fi(:,i)-59*fi(:,i-1)+37*fi(:,i-2)-9* fi(:, i-3));
30     % Korrektor
31     y(:, i+1) = y(:,i) +h/24*(9* f(t_0+i*h, yp)+19*fi(:,i)...
32         -5*fi(:,i-1)+fi(:,i-2));
33     fi(:,i+1) = f(t_0+i*h, y(:, i+1));
34 end
35 end
```

Alternative mit variablem Einschrittverfahren zur Berechnung der Startwerte:

```
1  function y = einschritt(phi, f, N, t_0, y_0)
2
3  % Funktion zur Implementierung von Einschrittverfahren zur Loesung der ODE
4  %       y'=f(t,y)
5  % mit Anfangswert y(t_0)= y_0 mittels der Verfahrensfunktion phi und
6  % der Schrittweite h=1/N
7
8  h = 1/N;
9  t = t_0;
10
11 y = zeros(length(y_0), N+1);
12 y(:,1)= y_0(:)';
13
14 for k=1:N
15    y(:,k+1)= y(:,k)+h*feval(phi, f, t, y(:,k),h);
16    t = t + h;
17 end
18
19 end
```

```
1  function [y] = adams_bashforth(t_0, y_0, t_b, f, N, verfahren)
2  % Die Funktion loest eine gegebene gewoehnliche Differentialgleichung (ODE)
3  % y'=f(t,y)
4  % mit dem N-Schrittverfahren von Adams-Bashforth
5  %
6  % INPUT:  t_0        Anfangszeitpunkt
7  %         y_0        Startwert y(t_0)
8  %         f          rechte Seite der ODE
```

```
%           N           Anzahl der Schritte
%           verfahren   Einschrittverfahren fuer die Startrechnung
% OUTPUT:   y           Naeherung an y nach N Schritten
%

h = (t_b-t_0)/N;
t = t_0;
y = zeros(length(y_0), N+1);
y(:,1) = y_0(:)';
fi(:,1) = f(t_0, y_0);

% Startrechnung
for i = 1:3
    y_start = einschritt(verfahren, f, N, t +i*h, y(:,i));
    y(:,i+1) = y_start(i+1);
    fi(:,i+1) = f(t_0+i*h, y(:,i+1));
end

for i=4:N
    y(:, i+1) = y(:,i)+h/24*(55*fi(:,i)-59*fi(:,i-1)+37*fi(:,i-2)-9* fi(:, i-3));
    fi(:,i+1) = f(t_0+i*h, y(:, i+1));
end
end
```

```
function [ y ] = adams_moulton(t_0,y_0,t_b,f,N,verfahren)
% Die Funktion loest eine gegebene gewoehnliche Differentialgleichung (ODE)
% y'=f(t,y)
% mit dem N-Schrittverfahren von Adams-Moulton
%
% INPUT:    t_0         Anfangszeitpunkt
%           y_0         Startwert y(t_0)
%           f           rechte Seite der ODE
%           N           Anzahl der Schritte
%           verfahren   Einschrittverfahren fuer die Startrechnung
% OUTPUT:   y           Naeherung an y nach N Schritten

h =(t_b-t_0)/N;
t = t_0;
y = zeros(length(y_0), N+1);
y(:,1) = y_0(:)';
fi(:,1) = f(t_0, y_0);

% Startrechnung
for i = 1:3
y_start = einschritt(verfahren,f, N,t +i*h,y (:,i));
y(:,i+1) = y_start(i+1);
fi(:,i+1) = f(t_0+i*h, y(:,i+1));
end

% Praediktor-Korrektor-Schritt
for i=4:N
    % Praediktor
    yp = y(:,i)+h/24*(55*fi(:,i)-59*fi(:,i-1)+37*fi(:,i-2)-9* fi(:, i-3));
    % Korrektor
```

```
    y(:, i+1) = y(:,i) +h/24*(9* f(t_0+i*h, yp)+19*fi(:,i)...
        -5*fi(:,i-1)+fi(:,i-2));
    fi(:,i+1) = f(t_0+i*h, y(:, i+1));
end
end
```

```
% Aufgabe 10: Test Verfahren von Adams-Bashforth und Adams-Moulton
clear all
close all

fprintf('\n************************************************************\n');
fprintf('\nAufgabe 10: Test Verfahren von Adams-Bashforth und Adams-Moulton\n');
fprintf('\n************************************************************\n');

h = 0.1;
t_0 = 0;
t_b = 3;
N = (t_b-t_0)/h;
y_0 = 2;

eab = zeros(N+1,1);
eam = zeros(N+1,1);

% Exakte Loesung
t = linspace(t_0,t_b, N+1);
y_exakt = f10exakt(t)

%-------------------------------------------------------------------------
% Loese mit Mehrschrittverfahren
%-------------------------------------------------------------------------

% Variante 1: Startwerte mit RKV berechnet
yab_runge = adams_bashforth(t_0, y_0, t_b, @f10, N, @rungekutta);
yam_runge = adams_moulton(t_0, y_0, t_b, @f10, N, @rungekutta);

% Variante 2: Startwerte mit explizitem Euler berechnet
yab_euler = adams_bashforth(t_0, y_0, t_b, @f10, N, @euler);
yam_euler = adams_moulton(t_0, y_0, t_b, @f10, N, @euler);

% Variante 3: Startwerte entsprechen exakter Loesung
yab_exakt = adams_bashforth_exakt(t_0, y_0, t_b, @f10, N, @f10exakt);
yam_exakt = adams_moulton_exakt(t_0, y_0, t_b, @f10, N, @f10exakt);

%-------------------------------------------------------------------------
% Berechne globale Fehler
%-------------------------------------------------------------------------

% Variante 1: Startwerte mit RKV berechnet
eab_runge = abs(yab_runge - y_exakt);
eam_runge = abs(yam_runge - y_exakt);

% Variante 2: Startwerte mit explizitem Euler berechnet
eab_euler = abs(yab_euler - y_exakt);
eam_euler = abs(yam_euler - y_exakt);
```

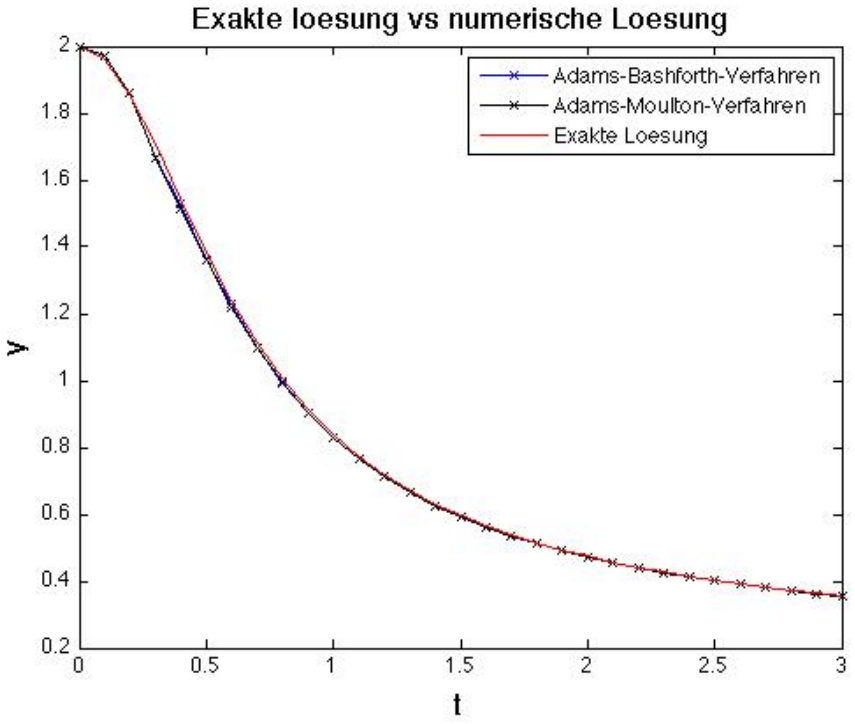
```
49
50 % Variante 3: Startwerte entsprechen exakter Loesung
51 eab_exakt = abs(yab_exakt - y_exakt);
52 eam_exakt = abs(yam_exakt - y_exakt);
53
54 %-------------------------------------------------------------------------
55 % Tabellarische Ausgabe
56 %-------------------------------------------------------------------------
57
58 % Ausgabe der Iterierten und der exakten Loesung (fuer Variante mit exakten
59 % Startwerten)
60 fprintf('\nt     Adams-Bashforth   Adams-Moulton   Exakte Loesung \n')
61 for i=1:N+1
62 fprintf('%2.2f   %2.6e     %2.6e    %2.6e   \n', ...
63     t_0+(i-1)*h, yab_exakt(i), yam_exakt(i) , y_exakt(i))
64 end
65
66 % Ausgabe der globalen Fehler (fuer Variante mit exakten Startwerten)
67 fprintf('\nt   globaler Fehler   Adams-Bashforth   Adams-Moulton \n')
68 for i=1:N+1
69 fprintf('%2.2f                %2.6e     %2.6e \n',  t_0+(i-1)*h, ...
70     eab_exakt(i), eam_exakt(i))
71 end
72
73 %-------------------------------------------------------------------------
74 % Graphische Ausgabe
75 %-------------------------------------------------------------------------
76
77 % Exakte Loesung und numerische Loesung
78 figure(1)
79 t = linspace(t_0,t_b, N+1);
80 plot(t, yab_runge,'x-', t, yam_runge, 'k-x')
81 title('Exakte loesung vs numerische Loesung ','FontSize', 14)
82 xlabel('t', 'FontSize', 14)
83 ylabel('y', 'FontSize', 14)
84 hold on;
85 plot(t,1./(log ( t.^2+1)+0.5),'r')
86 legend ('Adams-Bashforth-Verfahren', 'Adams-Moulton-Verfahren',...
87     'Exakte Loesung', 'Location', 'Northeast')
88
89
90 % Vergleich der globalen Fehler
91 figure(2)
92 plot(t, eab_exakt, 'r-*', t, eam_exakt, 'b-*')
93 title('Globale Fehler Adams-Bashforth- vs. Adams-Moulton-Verfahren',...
94     'FontSize', 14)
95 legend ('Adams-Bashforth-Verfahren', 'Adams-Moulton-Verfahren',...
96     'Location', 'Northeast')
97
98 figure(3)
99 plot(t, eab_runge, 'r-+', t, eab_euler, 'r-o', t, eab_exakt, 'r-*', ...
100     t, eab_runge, 'b-+', t, eam_euler, 'b-o', t, eam_exakt, 'b-*' )
101 title('Globale Fehler AB- vs. AM-Verfahren, unterschiedliche Startwerte', ...
102     'FontSize', 14)
```

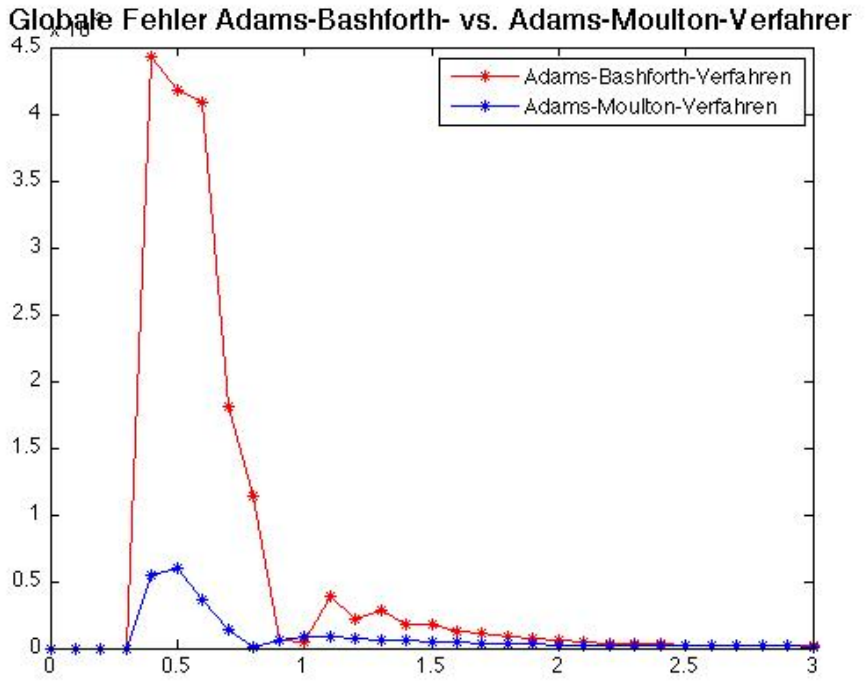
```
103 legend ('ABV runge', 'ABV euler', 'ABV exakt', 'AMV runge', ...
104     'AMV euler', 'AMV exakt','Location', 'Northeast')
105
106
107
108 % g2 = figure
109 % plot(t, eab, 'r-*', t, eam, 'b-*')
110 % title('Exakte loesung vs numerische Loesung ','FontSize', 14)
111 % xlabel('t', 'FontSize', 14)
112 % ylabel('y', 'FontSize', 14)
```

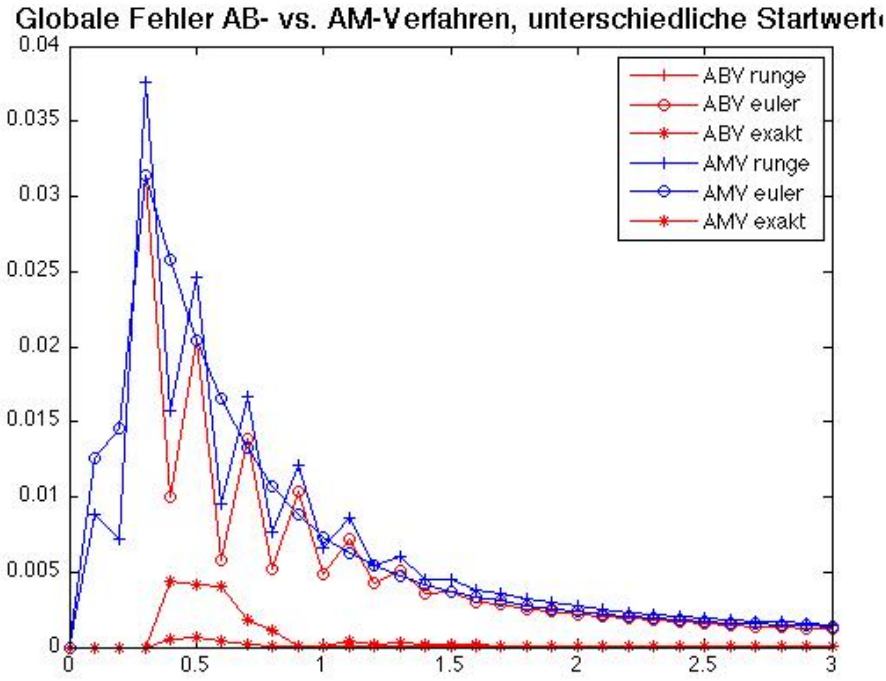
t	Adams–Bashforth	Adams–Moulton	Exakte Loesung
0.00	2.000000e+00	2.000000e+00	2.000000e+00
0.10	1.960975e+00	1.960975e+00	1.960975e+00
0.20	1.854528e+00	1.854528e+00	1.854528e+00
0.30	1.705967e+00	1.705967e+00	1.705967e+00
0.40	1.546634e+00	1.541663e+00	1.542210e+00
0.50	1.387028e+00	1.382259e+00	1.382851e+00
0.60	1.242504e+00	1.238049e+00	1.238414e+00
0.70	1.114437e+00	1.112484e+00	1.112624e+00
0.80	1.006472e+00	1.005331e+00	1.005332e+00
0.90	9.145782e−01	9.147030e−01	9.146396e−01
1.00	8.380790e−01	8.382032e−01	8.381196e−01
1.10	7.730108e−01	7.734827e−01	7.733997e−01
1.20	7.181801e−01	7.184664e−01	7.183918e−01
1.30	6.710602e−01	6.714122e−01	6.713477e−01
1.40	6.306665e−01	6.308947e−01	6.308395e−01
1.50	5.955385e−01	5.957622e−01	5.957150e−01
1.60	5.649284e−01	5.650889e−01	5.650482e−01
1.70	5.379874e−01	5.381301e−01	5.380946e−01
1.80	5.141743e−01	5.142856e−01	5.142543e−01
1.90	4.929734e−01	4.930691e−01	4.930413e−01
2.00	4.740060e−01	4.740851e−01	4.740599e−01
2.10	4.569410e−01	4.570092e−01	4.569864e−01
2.20	4.415160e−01	4.415745e−01	4.415536e−01
2.30	4.275083e−01	4.275596e−01	4.275402e−01
2.40	4.147344e−01	4.147797e−01	4.147617e−01
2.50	4.030393e−01	4.030798e−01	4.030630e−01
2.60	3.922926e−01	3.923292e−01	3.923134e−01
2.70	3.823834e−01	3.824167e−01	3.824019e−01
2.80	3.732172e−01	3.732477e−01	3.732336e−01
2.90	3.647126e−01	3.647408e−01	3.647275e−01
3.00	3.567999e−01	3.568261e−01	3.568134e−01

t	globaler Fehler	Adams–Bashforth	Adams–Moulton
0.00		0.000000e+00	0.000000e+00
0.10		0.000000e+00	0.000000e+00
0.20		0.000000e+00	0.000000e+00
0.30		0.000000e+00	0.000000e+00
0.40		4.423476e−03	5.475570e−04
0.50		4.176570e−03	5.924236e−04
0.60		4.090109e−03	3.644603e−04
0.70		1.813017e−03	1.397864e−04
0.80		1.140021e−03	8.461821e−07

0.90	6.137485e−05	6.345890e−05
1.00	4.055364e−05	8.363542e−05
1.10	3.888918e−04	8.301583e−05
1.20	2.117134e−04	7.462498e−05
1.30	2.874234e−04	6.454140e−05
1.40	1.729975e−04	5.516807e−05
1.50	1.764749e−04	4.722047e−05
1.60	1.197508e−04	4.073137e−05
1.70	1.072693e−04	3.550416e−05
1.80	8.001229e−05	3.129842e−05
1.90	6.786817e−05	2.789630e−05
2.00	5.396804e−05	2.511986e−05
2.10	4.534307e−05	2.283020e−05
2.20	3.760859e−05	2.092101e−05
2.30	3.199359e−05	1.931150e−05
2.40	2.735455e−05	1.794023e−05
2.50	2.374087e−05	1.676029e−05
2.60	2.080036e−05	1.573558e−05
2.70	1.842919e−05	1.483812e−05
2.80	1.648331e−05	1.404600e−05
2.90	1.487898e−05	1.334191e−05
3.00	1.354225e−05	1.271203e−05







Hinweise:

Die Programmieraufgaben sind in Matlab zu erstellen. Senden Sie alle Files in einer email mit dem Betreff **Loesung-Blatt4** an **angewandte.numerik@uni-ulm.de** (Abgabetermin jeweils wie beim Theorieteil). Drucken Sie zusätzlich allen Programmcode sowie die Ergebnisse aus und geben Sie diese vor der Übung ab. Der Source Code sollte strukturiert und, wenn nötig, dokumentiert sein.