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

WS 2013/2014
Lösungsblatt 5
29.11.2013

Angewandte Numerik 2

Aufgabe 11 *(Newton-Interpolation)* (8 Punkte)

Gegeben seien äquidistante Stützstellen $t_k = t_0 + kh$. Bestimmen Sie zu $(n+1)$ gegeben Stützpunkten $(t_i, f_i), i = 0, \ldots, n$ das Interpolationspolynom $I_f \in \mathbb{P}_n$ in Newton-Form

$$I_f(s) = \nabla^0 f_0 + \frac{1}{h^1}\nabla^1 f_1(s - t_0) + \cdots + \frac{1}{n!h^n}\nabla^n f_n(s - t_0)\cdots(s - t_{n-1}).$$

a) Schreiben Sie hierzu eine Matlab-Routine `d=divdiff(t0, h, f)`, die mit dividierten Differenzen die Koeffizienten $\nabla^0 f_0, \ldots \nabla^n f_n$ des Newton-Polynoms zu den Daten $(t_i, f_i), i = 0, \ldots, n$ liefert, d. h. die Rückwärtsdifferenzen.

b) Schreiben Sie eine Routine `f=newtonpolynom(t0,h,d,s)`, die das Newton-Polynom zu den Stützstellen t_j und den Koeffizienten $\nabla^0 f_0, \ldots \nabla^n f_n$ mithilfe eines Horner-artigen Schemas im Punkt s auswertet.

Lösung:

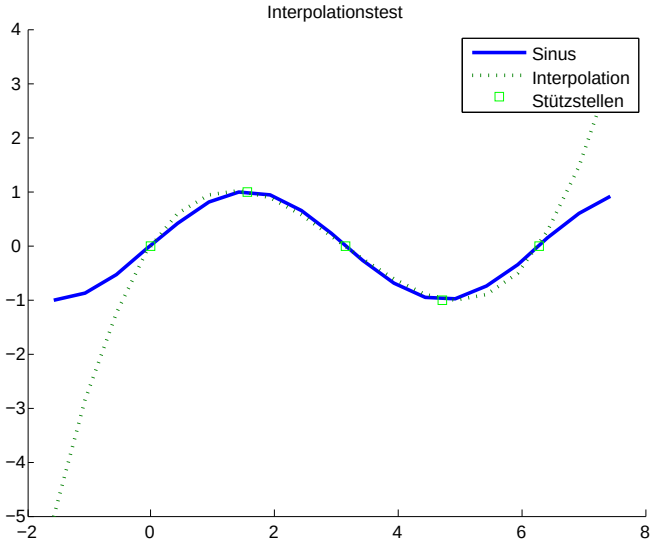
a)

```
1  function d = divdiff(f)
2  % Berechnung der Koeffizienten mit dividierten Differenzen
3
4  % Input:  f  Vektor mit Funktionswerten
5  % Output: d  Vektor mit dividierten Differenzen
6
7  n = length(f);
8
9  d = f;
10 for k = 2 : n
11     for j = n : -1 : k
12         d(:, j) = d(:,j) - d(:,j-1);
13     end
14 end
```

b)

```
function f = newtonpolynom(t0, h, d, s)
% Die Funktion wertet das Newton-Polynom an der Stelle d aus

% Input: t0 Startzeitpunkt
%        h  Schrittweite
%        f  Polynomkoeffizinten (Dividierte Differenzen)
%        s  Auswerzungspunkt
%
% Output f Newton-Polynom zu den Daten (t_i, f_i) an der Stelle s
%          (Spaltenvektor)

n = length(d);
t = t0:h:n*h;
f = d(n);

for i = n:-1:1
    f = d(i)+ (s-t(i)).*f/(i*h);
end
```

```
% Test der Funktionen anhand der Interpolation des Sinus
clear all, close all

t0 = 0;
h  = pi/2;

% zu Inderpolierende Testfunktion
T  = t0 : h : 2*pi;
F  = sin(T)
length(F)

% Berechnung der Koeffizienten mit dividierten Differenzen
d  = divdiff(F);

Tplot = -pi/2:.5:5/2*pi;
FSINplot = sin(Tplot);
size(Tplot)
FINTplot = zeros(1,size(Tplot, 2));
for i = 1:size(Tplot, 2)
    FINTplot(1,i) = newtonpolynom(t0, h, d, Tplot(i))
end

%Plot exakte Loesung, Interpolation und Knoten
figure;
hold all;
plot(Tplot, FSINplot, 'linewidth', 2);
plot(Tplot, FINTplot, ':', 'linewidth', 2);
plot(T, F, 'gs');
legend('Sinus', 'Interpolation', 'St¸tzstellen');
title('Interpolationstest');
```



Aufgabe 12 *(Mehrschrittverfahren in Rückwärtsdifferenzen Darstellung)* (10 Punkte)

Das Ziel ist es das Mehrschrittverfahren

$$y_{j+k} = y_{j+r-\ell} + h \sum_{i=0}^{r} \gamma_i^{(r,\ell)} \nabla^i f_{j+r} \tag{1}$$

zu implementieren.

a) Schreiben Sie hierzu eine Maple-Routine, welche die Koeffizienten

$$\gamma_i^{(r,\ell)} = (-1)^i \int_{-\ell}^{k-r} \binom{-s}{i} \, ds$$

mit

$$\binom{-s}{i} = \frac{-s(-s-1)(-s-2)\cdot(-s-(i-1))}{i!}$$

berechnet.

b) Schreiben Sie Matlab-Funktionen, in denen Sie die Verfahren aus Aufgabe 10 c), d.h. das Adams-Bahforth- und das Adams-Moulton-Verfahren in der Rückwärtsdifferenzen Darstellung (1) implementieren.

Hinweis: Sie können die Koeffizienten $\gamma_i^{(r,\ell)}$ auch ohne a) berechnen. Man kann einerseits die Formel

$$\sum_{j=0}^{i} \frac{\gamma_{i-j}^{(k-1,0)}}{j+1} = 1$$

zeigen, aus der sich die Koeffizienten rekursiv berechnen lassen und andererseits den Zusammenhang

$$\gamma_j^{(k,1)} = \gamma_j^{(k-1,0)} - \gamma_{j-1}^{(k-1,0)}.$$

Wenn Sie a) nicht bearbeitet haben, verwenden Sie diesen Hinweis.

c) Testen Sie diese Funktionen sowie die aus Aufgabenteil b), in dem Sie die Anfangswertaufgabe aus Aufgabe 10 d)

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

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.

d) Vergleichen Sie ihre Mehrschrittverfahren von diesem Blatt mit denen vom Blatt 4 hinsichtlich der Vor- und Nachteile der jeweiligen Darstellungsformen.

e*) Schreiben Sie eine Matlab-Funktion die das Nyström-Verfahren ($r = k-1 = 3, \ell = 1$) in der Darstellung (1) implementiert und eine weitere Funktion die das Milne-Simpson-Verfahren ($r = k = 3, \ell = 2$) ebenfalls in der Rückwärtsdifferenzen Darstellung (1) berechnet.

Lösung:

b)

```
1  function d = divdiff_mod(f)
2  % Berechnung der Koeffizienten mit dividierten Differenzen
3
4  % Input:  f  Vektor mit Funktionswerten
5  % Output: d  Vektor mit di = nabla^i*f_3
6
7  n = length(f);
8  d(1) = f(end);
9
10 for k = 2 : n
11     for j = n : -1 : k
12         if j == n
13             d(k) = f(j) - f(j-1);
14         end
15         f(j) = f(j) - f(j-1);
16     end
17 end
```

```
function [y, t] = adams_bashforthBD(y_0, h, fun, N)
% Die Funktion berechnet die numerische Loesung einer ODE
% y'=f(t,y)
% mit dem N Schrittverfahren von Adams-Bashforth
% mit R¸ckw%ortsdifferenzen
%
% INPUT:  y_0       Vektor von Startwerten
%         fun       rechte Seite der ODE
%         N         Anzahl der Schritte
% OUTPUT: y         Naeherung an y nach N Schritten
%

y = y_0;
y1 = y_0;
t = 0:h:(N+3)*h;
% Mit Maple berechnete Koeffizienten
gamma =[1 0.5 5/12 3/8];

for j=1:1:N
    f = fun(t(j:j+3),y(j:j+3));  % ODE auswerten
    d = divdiff_mod(f);
    y(j+4) = y(j+3) + h*gamma*d';
end
end
```

```
function [y,t] = adams_moultonBD(y_0, h, fun, N, maxIter, tol)
    %Moulton mit Rueckwaertsdifferenzen

y = y_0(1:3);
t = 0:h:N*h;

% Mit Maple berechnete Koeffizienten
gamma = [1 -0.5 -1/12];

for j = 1: N-2
    y_alt = 0;
    iter =1;
    while (iter <= maxIter)
        f = fun(t(j:j+2),[y(j:j+1), y_alt]); % DGL auswerten
        d = divdiff_mod(f);
        y_neu = y(j+2)+h*gamma*d';
        % Verbesserung geringer als vorgegebene Toleranz
        if ((abs(y_neu-y_alt)) <= tol)
            break
        end
        iter = iter+1;
        y_alt = y_neu;
    end
    y(j+3) = y_neu;
end
end
```

c)

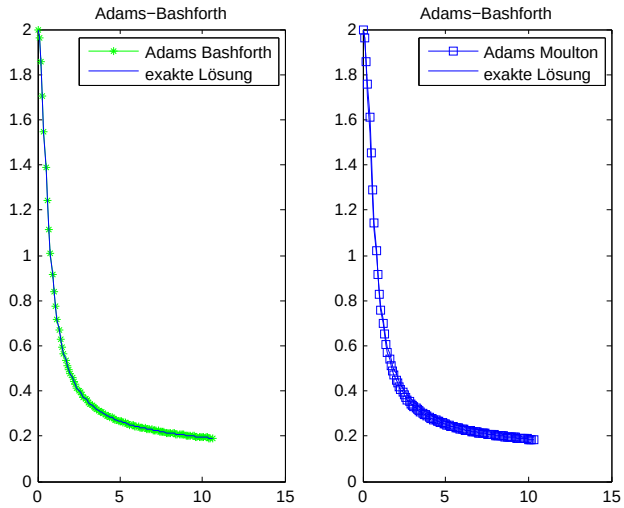
```
function dydt = f10(t,y)
    dydt = (-2*t.*y.^2)./(t.^2 + 1);
end
```

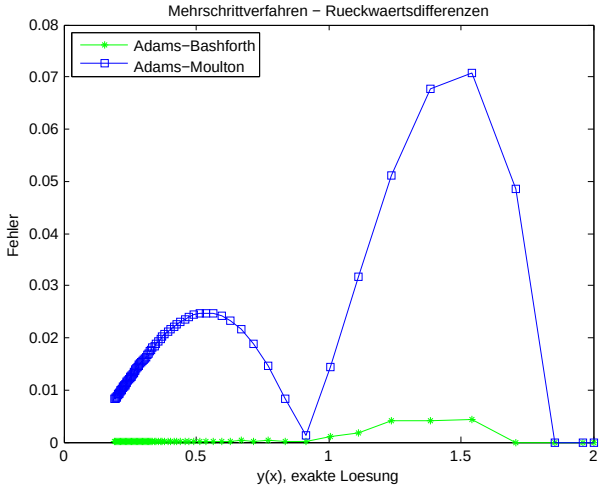
```
function y = y_exakt(t)
% Berechnet die exakte Loesung
    y = 1./(log(t.^2 + 1) + 0.5);
end
```

```
% Aufggabe 12 c)
clc; clear all; close all;
h       = 0.1; % Schrittweite
N       = 103; % Anzahl an Schritten(inklusuve der Startrechnung)
maxIter = 10000; % maximale Anzahl an Fixpunktiterationen
tol     = 0.00001; % Toleranz fuer Verbesserung in Fixpunktiteration

% Berechne exakte Werte als Startwerte
t0 = 0:h:3*h;      % Anfangszeitwerte
y0 = y_exakt(t0) % Anfangswerte

% Berechnung der numerischen Loesungen
[y_ab,y1, t_ab] = adams_bashforthBD(y0, h, @f10, N);
[y_am, t_am] = adams_moultonBD(y0, h, @f10, N, maxIter, tol);

% Plot der numerischen Loesungen und der exakten
figure(1)
subplot(1,2,1)
plot(t_ab,y_ab,'-g*',t_ab,y_exakt(t_ab));hold on;
legend('Adams Bashforth','exakte L^sung')
title('Adams-Bashforth')
subplot(1,2,2)
plot(t_am,y_am,'-bs',t_am,y_exakt(t_am));
legend('Adams Moulton','exakte L^sung')
title('Adams-Bashforth')

% Fehlerplots
figure(2)
plot(t_ab,abs(y_exakt(t_ab)-y_ab), '-g*', t_ab, abs(y_exakt(t_ab)-y1),'-r*')
hold on
plot(t_am,abs(y_exakt(t_am)-y_am),'-bs')
xlabel('t')
ylabel('Fehler')
title('Mehrschrittverfahren - Rueckwaertsdifferenzen')
legend('Adams-Bashforth','Adams-Moulton','Location','NorthWest')
```

d)

Vorteile der Lagrange-Darstellungsform (vom letzten Übungsblatt): Änderung der Schrittweite ist einfacher, weniger Rechenoperationen. Vorteile der Newton-Darstellungsform: numerisch stabiler, Erhöhung der Ordnung durch anhängen zusätzliche Stützpunke möglich, einfache Schätzung des Diskretisierungsfehlers (Restgliedabschätzung des Newton-Interpolationspolynoms).


Adams−Bashforth
Adams Bashforth
exakte Lösung
Adams−Bashforth
Adams Moulton
exakte Lösung


Mehrschrittverfahren − Rueckwaertsdifferenzen
Adams−Bashforth
Adams−Moulton
Fehler
y(x), exakte Loesung