Prof. Dr. Stefan Funken
M.Sc. Andreas Bantle
Dipl.-Math. oec. Klaus Stolle

Universität Ulm
Institut für Numerische Mathematik
Sommersemester 2015

Numerische Analysis - Matlab-Blatt 4
Lösung

(Besprechung in den MATLAB-Tutorien in KW 23/24)

Hinweise:
Siehe MATLAB-Blatt 1/2.

Aufgabe 6 *(Dividierte Differenzen)* (5+5+5 Punkte)
In dieser Aufgabe wollen wir die $(n+1)$-te Ableitung mittels dividierter Differenzen an mehreren Stellen x_k mit

$$x_k = a + k\,\varepsilon, \quad k = 0, ..., N.$$

approximieren ($N = \frac{b-a}{\varepsilon}$). Dazu verwenden wir die Identität

$$(n+1)!\,[x_0, ..., x_{n+1}]f = f^{(n+1)}(\xi) \quad \text{mit} \quad \xi \in \left[\ \min\{x_0, ..., x_{n+1}\}, \max\{x_0, ..., x_{n+1}\}\ \right].$$

Bearbeiten Sie die folgenden Aufgaben.

(i) Schreiben Sie eine MATLAB-Funktion

`val = approxDeriv(x,fx,n),`

die die Stützstellen `x` $= (x_0, ..., x_N)$, die Funktionswerte `fx` und die Ordnung der Ableitung `n` als Eingabeparameter erhält. Die Funktion soll die dividierte Differenzen

$$n!\,[x_k, ..., x_{k+n}]f, \quad k = 0, ..., N-n$$

als Approximationen der Ableitung $f^{(n)}\left(\frac{x_k+x_{k+n}}{2}\right)$ berechnen und in dem Vektor `val` zurückgeben.

```
function val = approxDeriv(x,fx,n)

for k = 2:n+1
  for j = length(fx):-1:k
    fx(j) = (fx(j) - fx(j-1))/(x(j)-x(j-k+1));
  end
end
val = fx(n+1:end)*prod(1:n);
```

(ii) Schreiben Sie ein MATLAB-Skript `main.m`, in welchem Sie für eine gegebene Schrittweite `eps` $= 10^{-2}$ die $(n+1)$-te Ableitung der Funktion f auf dem Intervall $[a, b]$ mit der Funktion aus Aufgabe (i) auswerten und zusammen mit der exakten Ableitung in ein Schaubild plotten. Implementieren Sie die folgenden Beipsiele

- $f(x) = \cos(x)$, $n = 3$, $[a, b] = [0, 2\pi]$
- $f(x) = (\exp(1 - x^2) - 1)^{1/3}$, $n = 0$, $[a, b] = [-0.5, 0.5]$

```
clc, clear all, close all
format long e
%*** definiere Toleranz
eps = logspace(-4,-1,100);
% eps=1e-1;

```

```
%*** Beispiel 1
n = 3;
f = @(x) cos(x);
fd = @(x) cos(x);
a=0;b=2*pi;



%*** Beispiel 2
n = 0;
f = @(x) (exp(1-x.^2)-1).^(1/3);
fd = @(x) -2/3*exp(1-x.^2).*x./(exp(1-x.^2)-1).^(2/3);
a=-0.5;b=0.5;

for j=1:length(eps)
  x   = a:eps(j):b;
  fd2 = approxDeriv(x,f(x),n+1);
  xn  = x(1:end-(n+1)) + (n+1)*eps(j)/2;
  fd1 = fd(xn);
  err(j) = max(abs(fd2-fd1));
  figure(1)
  plot(xn,fd(xn),'*-r',xn,fd2,'--b');

end

figure(2)
loglog(eps,err,'-r*')
```

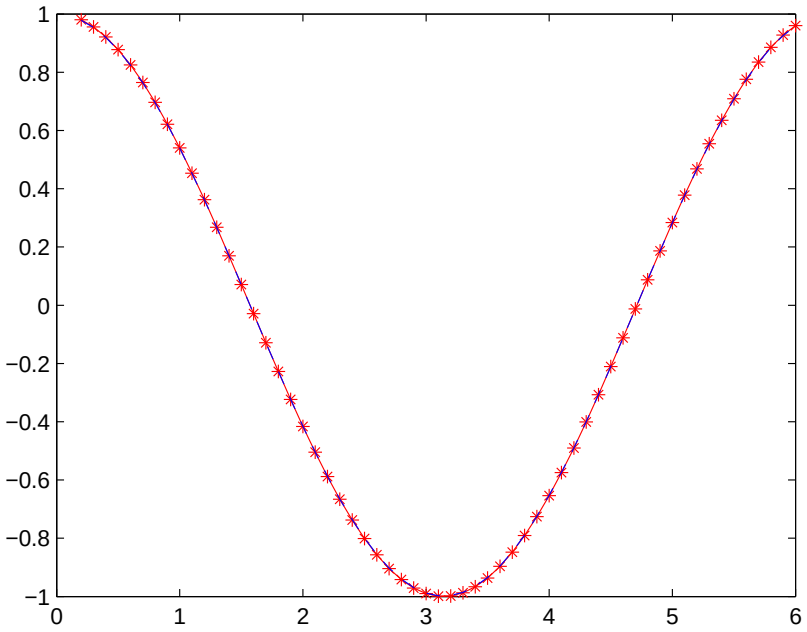

Abbildung 1: $f(x) = \cos(x)$

(iii) Erweitern Sie das Skript aus Aufgabenteil (ii), sodass die $(n+1)$-te Ableitung für die Schrittweiten

```
eps=logspace(-4,-1,100)
```

sowie der maximale Fehler auf dem Intervall $[a, b]$ berechnet werden. Plotten Sie den maximalen Fehler über die Schrittweite `eps` in doppelt-logarithmischer Skala. Was stellen Sie fest?

Man erhält für kleiner werdendes ε immer bessere Näherungen für die Ableitung. Wenn die Toleranz ε aber zu klein wird, d.h. die Stützstellen zu nah beieinander liegen, treten bei den dividierten Differenzen Instabilitäten auf und der Approximationsfehler steigt wieder. Das untenstehende Bild zeigt dies am Beispiel $f(x) = \cos(x)$.

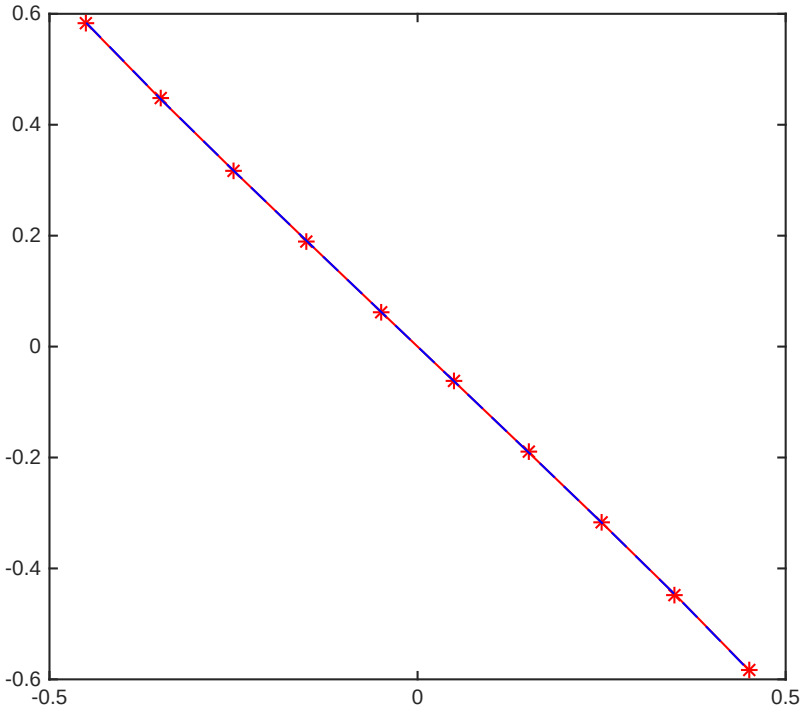

Abbildung 2: $f(x) = (\exp(1-x^2)-1)^{1/3}$

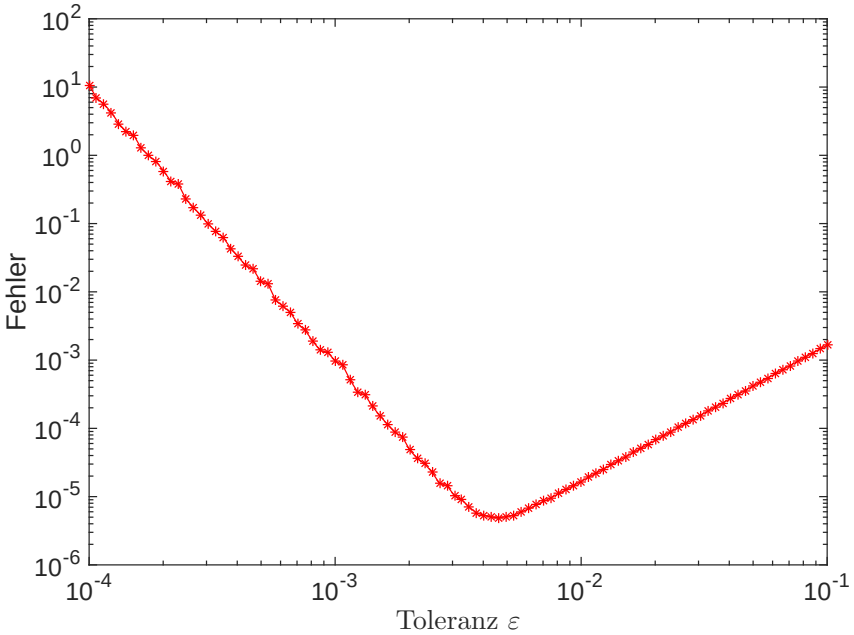

Abbildung 3: Fehler über die Toleranz ε.

Aufgabe 7 *(Interpolationsfehler)* (4+3+3+5 Punkte)

Der Interpolationsfehler hängt stark von der Wahl der Stützstellen ab, denn es gilt

$$|f(x) - P(f|x_0, ..., x_n)| \leq \frac{|(x-x_0)\cdot ... \cdot (x-x_n)|}{(n+1)!} |f^{(n+1)}(\xi)|, \quad \xi \in [\min(x_0, ..., x_n), \max(x_0, ..., x_n)].$$

Im Folgenden wollen wir den Term

$$\omega_{n+1}(x) := (x - x_0) \cdot ... \cdot (x - x_n)$$

für verschiedene Arten von Stützstellen numerisch untersuchen.

(i) Schreiben Sie ein MATLAB-Skript, welches $\omega_{n+1}(x)$ auf dem Intervall $[-1, 1]$ für

- ein äquidistantes Gitter $x_k = -1 + (k-1)\frac{2}{n-1}$ $(k = 1, ..., n)$
- die Tschebyscheff-Nullstellen $x_k = \cos\left(\frac{2k-1}{2n}\pi\right)$ $(k = 1, ..., n)$
- eine graduiertes Gitter $x_k = -1 + \left(\frac{k}{\frac{n}{2}+1}\right)^{\beta}$, $x_{k+\frac{n}{2}} = 1 - \left(\frac{k}{\frac{n}{2}+1}\right)^{\beta}$, $\beta = 2$, n gerade, $(k = 1, ..., \frac{n}{2})$

und $n = 30$ plottet. Sie dürfen zur Auswertung von $\omega_{n+1}(x)$ die Funktion `hornerNewton.m` aus Aufgabe 4 verwenden. Zeichnen Sie eine Legende ein. Welche Stützstellen sind am Besten? Vergleichen Sie das Ergebnis mit dem Ergebnis aus Aufgabe 5.

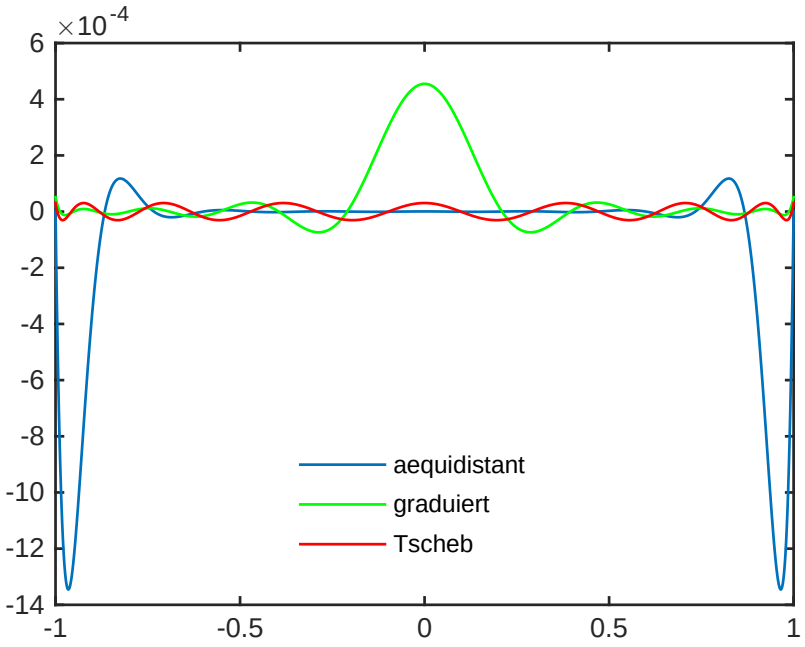

Abbildung 4: Für $n = 2^4$

Die Nullstellen der Tschebyscheff-Polynome sind die beste Wahl der Stützstellen (optimal da alle Extrema den gleichen Wert annehmen). Die äquidistanten Stützstellen approximieren die Funktion wie schon in Aufgabe 5 am Rand recht schlecht, das graduierte Gitter ist in der Intervallmitte nicht ausreichend.

(ii) Erweitern Sie das Skript, sodass für $n \in \{2^1, ..., 2^{10}\}$ jeweils der maximale Wert

$$M(n) := \max_{x \in [-1,1]} |\omega_{n+1}(x)|$$

bestimmt wird (Verwenden Sie dazu `x=linspace(-1,1,10000)`). Plotten Sie anschließend den maximalen Wert $M(n)$ über n in einer semi-logarithmischen Skala (`semilogy`). Was fällt Ihnen auf?

Man sieht, dass das $(n+1)$-te Newton Polynom bzgl. der Tschefyscheff-Knoten am schnellsten konvergiert. Knovergenzrate für die graduierten Knoten ist schlechte als für die Tschebyscheff-Knoten aber besser als die äquidistanten Knoten.

(iii) Bestimmen Sie numerisch die Konvergenzrate für alle drei Arten von Stützstellen, d.h. bestimmen Sie die konstante $0 < C < 1$ mit

$$M(n) \approx C^{n+1}.$$

Vergleichen Sie die Ergebnisse mit den theoretischen Aussagen, die Sie kennen.

`Für die Tschebyscheff-Knoten ergibt sich` $C = 1/2$, `der optimale Wert, der auch in der Vorlesung bewiesen wurde. Für äquidistante Knoten ergibt sich numerisch` $C \approx 0.732$.

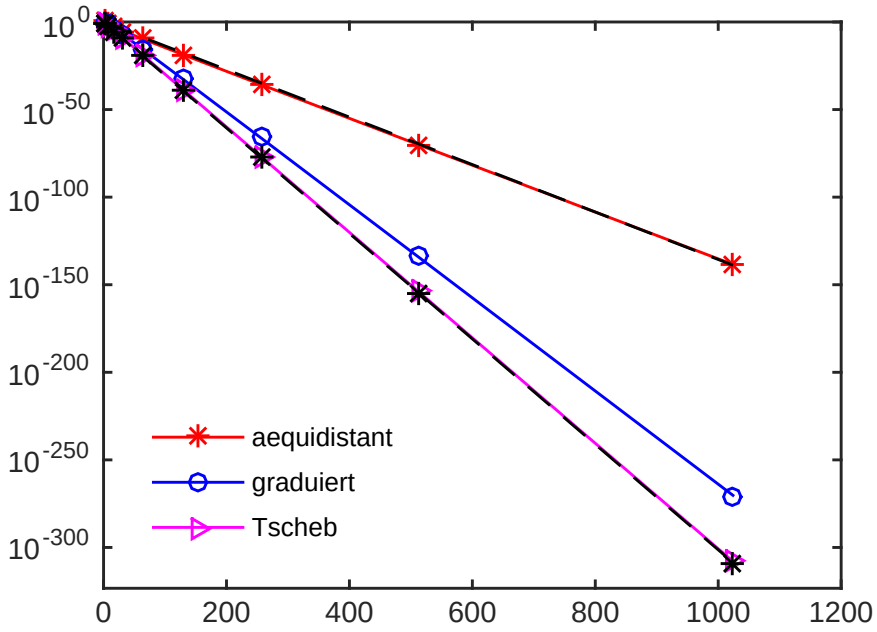

Abbildung 5: $M(n)$ über n

```
clc, clear all, close all
format long e
%*** definiere Toleranz
eps = logspace(-4,-1,100);
% eps=1e-1;

%*** Beispiel 1
n = 3;
f = @(x) cos(x);
fd = @(x) cos(x);
a=0;b=2*pi;



%*** Beispiel 2
n = 0;
f = @(x) (exp(1-x.^2)-1).^(1/3);
fd = @(x) -2/3*exp(1-x.^2).*x./(exp(1-x.^2)-1).^(2/3);
a=-0.5;b=0.5;

for j=1:length(eps)
  x   = a:eps(j):b;
  fd2 = approxDeriv(x,f(x),n+1);
  xn  = x(1:end-(n+1)) + (n+1)*eps(j)/2;
  fd1 = fd(xn);
  err(j) = max(abs(fd2-fd1));
  figure(1)
  plot(xn,fd(xn),'*-r',xn,fd2,'--b');

end

figure(2)
loglog(eps,err,'-r*')
```

(iv) Schreiben Sie ein MATLAB-Skript, welches den optimalen Parameter $\beta \in [1, 2.5]$ für das graduierte Gitter bestimmt. Legen Sie sich dazu äquidistante Werte für β an, berechnen Sie für jedes dieser β den maximalen Fehler $M(512)$ und plotten Sie $M(512)$ über β. Lassen Sie sich zusätzlich den Wert von β ausgeben, der $M(512)$ minimiert. Setzen Sie den optimalen Parameter von β in das Skript aus Aufgabenteil (ii) ein.

```
%**** graduiertes Gitter untersuchen
clc,clear all, close all
beta=linspace(1,2.5,100);
x=linspace(-1,1,10000);

N = 2^9;
a=[zeros(N,1);1];
for k=1:length(beta)
  if mod(N,2)==1
    x0=[-1+((1:N/2)/(N/2+1)).^beta(k),0,1-((1:N/2)/(N/2+1)).^beta(k)];
  else
    x0=[-1+((1:N/2)/(N/2+1)).^beta(k),1-((1:N/2)/(N/2+1)).^beta(k)];
  end
  x0 = leja(x0);
  w2=evalNewton(a,x,x0);
  m2(k)=max(abs(w2));
end

[m,idx] = min(m2);
betaOpt = beta(idx)

figure(3)
semilogy(beta,m2,'-r*')
xlabel('$\beta$','interpreter','latex','fontsize',20)
ylabel('$\max |\omega_{n+1}|$','interpreter','latex','fontsize',20)
set(gca,'fontsize',20)
```

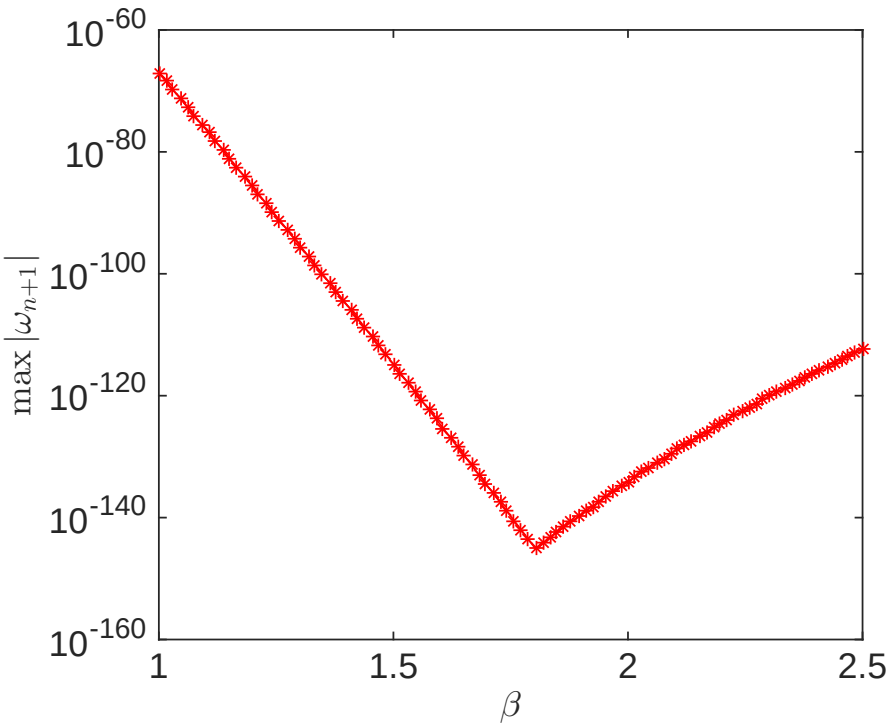

Abbildung 6: $\beta_{opt} \approx 1.80303$