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 1
Lösung

(Besprechung in den MATLAB-Tutorien in KW 17/18)

Hinweise

(i) Bitte melden Sie sich im SLC unter
`https://slc.mathematik.uni-ulm.de/portal/catalog/details/term/WS2014/lecture/949`
für die Vorlesung an.

(ii) Abgabe der Übungsblätter nur **zu zweit**! (Bis auf max. eine Ausnahme ;))

(iii) Zulassungskriterium für die Klausur: 50% der Übungspunkte der MATLAB- sowie der Theorie-Blätter.

(iv) Auf jedem Theorie-Übungsblatt wird es eine auf Englisch gestellte Aufgabe sowie eine Aufgabe, die in LaTeX abgegeben werden muss, geben. Die auf Englisch gestellte Aufgabe kann auf Deutsch beantwortet werden, handschriftliche Lösungen der LaTeX- Aufgabe werden mit 0 Punkten bewertet!

(v) Außerdem müssen die '*.tex'-Dateien der LaTeX-Aufgabe per Email an **numerik2ss15@gmail.com** mit dem Betreff

Blatt_Blattnumer, Name1 Vorname1, Name2 Vorname2

gesendet werden (Nachnamen alphabetisch sortiert!), also z.B für das erste Theorieblatt von Max Maier und Steffen Schneider:

Blatt_01, Maier Max, Schneider Steffen

Aufgabe 1 *(Nicht-Lineare Gleichungen)* (20 Punkte)

(i) Schreiben Sie eine Funktion `function x = myBisection(f,x0,x1,tol)` welche das Bisektions-Verfahren für die Funktion `f` und das Startintervall `[x0,x1]` durchführt. Das Verfahren soll abgebrochen werden, wenn die Intervallänge im k-ten Schritt kleiner als die Toleranz `tol` ist. Der Eingabeparameter `f` ist ein function-handle, der Rückgabewert `x` ist ein Vektor, der alle Intervallmittelpunkte der einzelnen Schritte enthält.

```
function x = myBisection(f,x0,x1,tol)

fx0 = f(x0);
fx1 = f(x1);

k = 1;
while abs(x0-x1)>tol
  x(k) = (x0+x1)/2;
  tmp = f(x(k));
  if abs(tmp)<eps
    break;
  elseif tmp*fx0<0
    x1 = x(k);
    fx1 = tmp;
  else
    x0 = x(k);
    fx0=tmp;
  end
```

```
19    k = k+1;
20  end
```

(ii) Schreiben Sie eine Funktion `function x = myNewton(f,f1,x0,tol)` welche das Newton Verfahren für die Funktion `f` mit Ableitung `f1` und Startwert `x0` durchführt. Das Verfahren soll abgebrochen werden, wenn $|f(x_k)| <$`tol` ist. Die Eingabeparameter `f` und `f1` sind function-handles, der Rückgabewert `x` ist ein Vektor, der alle Iterierten des Verfahrens enthält.

```
1  function x = myNewton(f,f1,x0,tol)
2
3  x(1) = x0;
4  k = 1;
5  while abs(f(x(k))>tol)
6    x(k+1) = x(k)-f(x(k))/f1(x(k));
7    k = k+1;
8  end
```

(iii) Schreiben Sie eine Funktion `function x = mySekanten(f,x0,x1,tol)` welche das Sekanten Verfahren für die Funktion `f` und Startwerten `x0` und `x1` durchführt. Das Verfahren soll abgebrochen werden, wenn $|f(x_k)| <$`tol` ist. Die Eingabeparameter `f` ist ein function-handle, der Rückgabewert `x` ist ein Vektor, der alle Iterierten des Verfahrens enthält.

```
1  function x = mySekanten(f,x0,x1,tol)
2
3  x(1) = x0;
4  x(2) = x1;
5  k = 2;
6  fxk1 = f(x(1));
7  fxk = f(x(2));
8  while abs(f(x(k)))>tol
9    x(k+1) = x(k)-(x(k)-x(k-1))/(fxk-fxk1)*fxk;
10   k=k+1;
11   fxk1 = fxk;
12   fxk = f(x(k));
13 end
```

(iv) Laden Sie das Skript main.m von der Homepage herunter und vervollständigen sie dieses. Das Skript soll folgende Aufgaben erfüllen:

- Definition der Funktionen $f(x) = (x+1)(x-1)$, $g(x) = (x-1)^2$ und $h(x) = x^{10} - 0.1$ sowie deren Ableitung als function-handles. Definition der exakten gesuchten Nullstelle `x_exact_f`, `x_exact_g` und `x_exact_h` (Zeilen 2-8).
- Definition der Anfangeswerte und der Toleranz für das Abbruchkriterium (Zeile 10-13).
- Durchführung der Verfahren sowie Berechnung der absoluten Fehler für f, g und h (Zeile 15-27).
- Grafische Darstellung der Fehler in semilogarithmischer Skala: Zeichnen Sie die Fehler aller Verfahren für jede Funktion in ein Bild und fügen Sie Achsenbeschriftungen und eine Legende ein (ab Zeile 29).

```
1  close all
2  % define functions and derivatives
3  f  = @(x) (x+1)*(x-1);
4  f1 = @(x) 2*x;
5  x_exact_f = 1;
6
7  h  = @(x) x^10-0.1;
8  h1 = @(x) 10*x^9;
9  x_exact_h = 0.1^(1/10);
10
11 g  = @(x) (x-1)^2;
12 g1 = @(x) 2*x-2;
13 x_exact_g = 1;
14
15 % initial values for Newton and secant method
16 x0 = 2; x1 = 1.9;
17 % tolerance for stopping criterion
```

```
tol = 1e-6;

% iterates for Newton and secant method for f
x_newton_f = myNewton(f,f1,2,tol);
x_secant_f = mySekanten(f,1.9,2,tol);
x_bisection_f = myBisection(f,0.75,2,tol);
% absolute errors for newton and secant method for f
error_newton_f = abs(x_newton_f-x_exact_f);
error_secant_f = abs(x_secant_f-x_exact_f);
error_bisection_f = abs(x_bisection_f-x_exact_f);

% iterates for Newton and secant method for g
x_newton_g = myNewton(g,g1,2,tol);
x_secant_g = mySekanten(g,2,1.9,tol);
x_bisection_g = myBisection(g,0.75,2,tol);
% absolute errors for newton and secant method for g
error_newton_g = abs(x_newton_g-x_exact_g);
error_secant_g = abs(x_secant_g-x_exact_g);
error_bisection_g = abs(x_bisection_g-x_exact_g);

% iterates for Newton and secant method for h
x_newton_h = myNewton(h,h1,2,tol);
x_secant_h = mySekanten(h,0,2,tol);
x_bisection_h = myBisection(h,0,2,tol);
% absolute errors for newton and secant method for h
error_newton_h = abs(x_newton_h-x_exact_h);
error_secant_h = abs(x_secant_h-x_exact_h);
error_bisection_h = abs(x_bisection_h-x_exact_h);

% semilogarithmic plots of all errors
figure(1)
semilogy(error_newton_f,'-bo');
hold on
semilogy(error_secant_f,'-rd');
semilogy(error_bisection_f,'-c>');
legend('Newton f', 'Secant f', 'Bisection f')
xlabel('number of iterations n')
ylabel('absolute error')
figure(2)
semilogy(error_newton_g,'-bo');hold on
semilogy(error_secant_g,'-rd');
semilogy(error_bisection_g,'-c>');
legend('Newton g', 'Secant g', 'Bisection g')
xlabel('number of iterations n')
ylabel('absolute error')
figure(3)
semilogy(error_newton_h,'-bo');hold on
semilogy(error_secant_h,'-rd');
semilogy(error_bisection_h,'-c>');
legend('Newton h', 'Secant h', 'Bisection h')
xlabel('number of iterations n')
ylabel('absolute error')
hold off
```

(v) Interpretieren Sie die Ergebnisse im Hinblick auf folgende Fragestellungen:

- Welches der Verfahren ist das Bessere?
  Das Newton-Verfahren konvergiert für einfache Nullstellen lokal quadratisch und ist damit das schnellste der drei Verfahren.
- Für welche der Funktionen funktionieren die Verfahren besser? Warum?
  Bei der Funktion f konvergieren das Newton- und das Sekantenverfahren von Anfang an quadratisch gegen die einfache Nullstelle.
  Bei der Funktion g haben wir wegen der doppelten Nullstelle nur lineare Konvergenz des Newton- und Sekantenverfahrens. Das Bisektionsverfahren konvergiert bei dieser Funktion gar nicht, da die Voraussetzung des Vorzeichenwechsels nicht erfüllt ist.
  Bei der Funktion h sieht man, dass die quadratische Konvergenz des Newtonverfahrens erst nach einigen

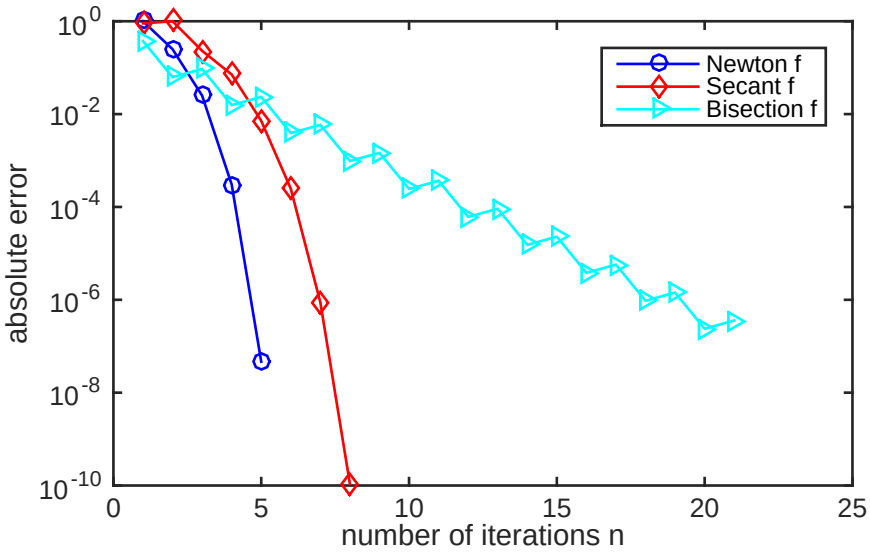

Abbildung 1: Fehlerplot der Funktion f

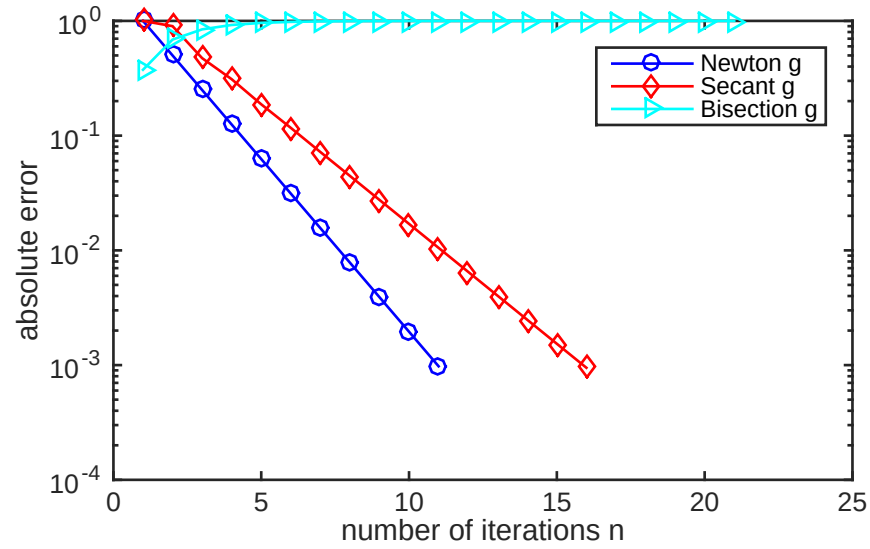

Abbildung 2: Fehlerplot der Funktion g

Iterationsschritten eintritt. Das Sekantenverfahren konvergiert nicht da der Startwert zu weit von der exakten Lösung entfernt ist.

- Ist das Abbruch-Kriterium sinnvoll?
  Je nach Aufgabenstellung ist das Kriterium sinnvoll. Ist man daran interessiert, dass der Funktionswert kleiner als die vorgegebene Toleranz ist, so ist das Abbruchkriterium sinnvoll. Ist man jedoch an einer möglichst genauer Approximation der Nullstelle interessiert, so ist das Kriterium nicht passend. Obwohl der Funktionswert kleiner als die Toleranz ist, kann die approximierte Nullstelle noch weit weg on der exakten liegen.

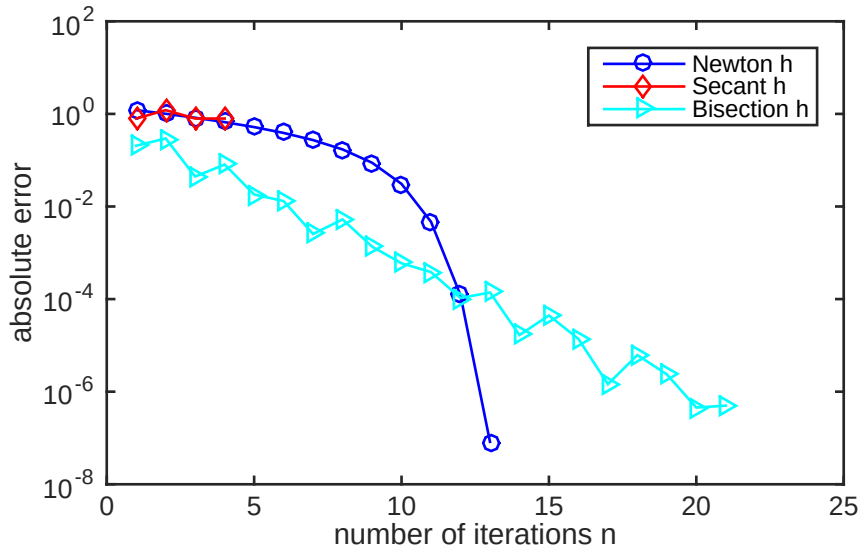

Abbildung 3: Fehlerplot der Funktion h