Richard Huber
richard.huber@uni-graz.at

Numerische Mathematik für LAK
12.Übungsblatt – 21ten Januar, 2020

12. Aufgabenblatt

In früheren Aufgaben beschäftigten wir uns mit dem Lösen von linearen Gleichungssystemen. Allerdings gibt es natürlich auch nichtlineare Gleichungen oder System die von Interesse sind, und die in der Regel schwerer zu lösen sind. Dabei wird nach einer Lösung x^* der Gleichung $f(x) = 0$ für eine (nichtlineare) Funktion $f\colon \mathbb{R}^N \to \mathbb{R}^N$ gesucht.

Aufgabe 1 (Bisektionsverfahren)**.** *Ein sehr "primitives", aber dennoch durchaus nützliches Verfahren zum approximativen Lösen eines nichtlinearen Gleichungssystems ist das Bisektionsverfahren. Dabei startet man mit $a_0 < b_0 \in \mathbb{R}$ sodass $f(a_0) < 0$ und $f(b_0) > 0$ für eine stetige Funktion $f\colon [a_0, b_0] \to \mathbb{R}$. Dann wird mit $x_{n+1} = \frac{a_n+b_n}{2}$, die Iteration*

$$a_{n+1} = \begin{cases} a_n & \text{falls } f(x_{n+1}) > 0 \\ x_{n+1} & \text{falls } f(x_{n+1}) \leq 0 \end{cases} \qquad b_{n+1} = \begin{cases} x_{n+1} & \text{falls } f(x_{n+1}) \geq 0 \\ b_n & \text{falls } f(x_{n+1}) < 0 \end{cases} \tag{1}$$

betrachtet. Falls $a_n = x_n = b_n$ d.h. $f(x_n) = 0$, so ist x_n Lösung der Gleichung (und die Iteration kann abgebrochen werden), andernfalls konvergiert $(x_n)_n$ für n gegen ∞ gegen eine Lösung von $f(x) = 0$.

Es soll das Bisektionsverfahren mit $a_0 = 1$ und $b_0 = 2$ durchgeführt werden um eine Nullstelle der Funktion $f(x) = x^2 - 2$ zu finden.

a) *Berechnen Sie die Grenzen a_n und b_n des Bisektionsverfahrens für $n = 1, \dots, 5$. Skizzieren Sie außerdem die Intervalle $[a_n, b_n]$ für $n = 1, \dots 5$ um die Funktionsweise des Algorithmus zu veranschaulichen. Sie können für die Berechnungen auch Matlab verwenden.*

b) *Wir bezeichnen mit $\delta_n = |x_n - x^*|$ wobei $x^* = \sqrt{2}$ die exakte Lösung ist. Zeigen Sie, dass für jedes $n \geq 1$*

$$\delta_n \leq \left(\frac{1}{2}\right)^n |b_0 - a_0|. \tag{2}$$

c) *Schätzen Sie via (2) den Wert δ_n für $n = 5$ ab ohne x^* explizit zu nutzen. Berechnen Sie darüber hinaus, wie viele Iterationen n höchstens nötig sind um $\delta_n < 10^{-10}$ zu erreichen.*

Hinweis. *Nutzen Sie den Zwischenwertsatz um zu zeigen, dass x^* stets in $[a_n, b_n]$ liegt, und schätzen Sie die Größe diese Intervalls ab.*

Bemerkung. *Das Bisektionsverfahren beruht auf dem Zwischenwertsatz für stetige Funktionen der Ihnen aus der Analysis bekannt sein sollte (wenn $f(a) < 0$ und $f(b) > 0$, muss es in $[a, b]$ eine Nullstelle von f geben), sowie der Vollständigkeit der reellen Zahlen. Dieses Verfahren lässt sich auf Dimensionen $N > 1$ verallgemeinern, was aber nicht gänzlich trivial ist da mehr Funktionen gleichzeitig berücksichtigt werden müssen.*

Matlab-Aufgabe 2 (Newton-Iteration)**.** *Mittels der Newton-Iteration lässt sich eine approximative Lösung eines nichtlinearen Gleichungssystems finden. Für eine stetig differenzierbare Funktion $f\colon \mathbb{R}^N \to \mathbb{R}^N$, für einen Startpunkt z_0 und die Iteration*

$$z_{n+1} = z_n - \big(D[f](z_n)\big)^{-1} f(z_n), \tag{3}$$

Richard Huber
richard.huber@uni-graz.at

Numerische Mathematik für LAK
12.Übungsblatt – 21ten Januar, 2020

gilt $x_n \to x^*$ sodass $f(x^*) = 0$ falls $\|x_0 - x^*\|$ hinreichend klein und die Funktion f gewisse (technische) Voraussetzungen erfüllt (auf welche wir nicht genauer eingehen). Hierbei bezeichnet

$$D[f](x) = \begin{pmatrix} \frac{\partial f_1(x)}{\partial x_1} & \cdots & \frac{\partial f_1(x)}{\partial x_N} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_N(x)}{\partial x_1} & \cdots & \frac{\partial f_N(x)}{\partial x_N} \end{pmatrix}$$

die Jakobimatrix der Funktion f und $(D[f](x))^{-1}$ die Inverse Matrix.

a) *Schreiben Sie [z,fnew]=my_newton_solver(z_0,Maxiter,f,Df), eine Matlab-Funktion welche anhand eines Startwerts z_0 eine Newton-Iteration für die Funktion f mit Jakobimatrix Df durchführt, welche nach "Maxiter" Iterationen abbricht, Die Funktion gibt die erreichte Approximation z (finale Iterierte) und den dazugehörigen Wert fnew=$f(z)$ zurück. In (3) sollte nicht tatsächlich die Inverse berechnet werden, sondern mit Backslash Operation gearbeitet werden. Der Input f und Df soll Funktionshandles entsprechen (siehe Hinweis). Testen Sie Ihre Implementierung für die Funktionen my_function.m, my_jacobi.m mit dem Startwert $z_0 = [1; 3; 2]^T$;.*

b) *Erweitern Sie die Funktion zu [z,fnew,Iter,flag]=my_newton_solver(z_0,my_eps,Maxiter,f,Df) sodass die Iteration abgebrochen wird falls $\|f(z_k)\| <$my_eps. Dies wird als erfolgreiche Newton-Iteration gesehen, während die Fälle in denen bis "Maxiter" iteriert wird (keine Konvergenz) oder das Newton-Verfahren nicht anwendbar ist als unerfolgreich gelten. Um zu überprüfen ob die Iteration anwendbar ist, stellen Sie in jeder Iteration sicher, dass die Determinante von Df im Betrag größer als 10^{-5} ist und brechen Sie andernfalls die Iteration ab. Der Algorithmus erhält zwei zusätzliche Outputs: die Anzahl der benötigten Iterationen als "Iter" und "flag" welches 1 ist wenn das Verfahren erfolgreich war (also vorzeitig fertig ist) und -1 andernfalls (wenn nicht durchführbar oder nicht konvergiert).*

c) *Gegeben sei der Matlabcode der Funktion my_newton_graphics.m (siehe unten). Erklären Sie den Code, und damit verbunden was die resultierende Grafik darstellt und welchen Einfluss die Parameter N und FOV haben. Führen Sie die Funktion mit Ihrer Implementierung des my_newton_solver für $N = 200$ und FOV$= 1$ aus (dies kann ein paar Minuten dauern). Achtung, Ihre Funktion muss dafür für komplexe Zahlen funktionieren.*

Hinweis. *Ein Funktionshandle lässt sich beispielsweise mit dem Befehl f=@(x) my_function(x); oder f=@(x) x^2-1; initialisieren. Dann lässt sich mit f(x) entsprechende Funktion aufrufen, und f kann wie eine Variable verwendet werden. Der Befehl "continue" in einer For-Schleife zwingt das Programm die aktuellen Iteration der Schleife zu beenden und mit der nächsten Iteration der For-Schleife fortzufahren. Wenn eine 3-dimensionale Matrix mit einem Bild assoziiert wird, gibt der Wert in (m, n, k) die Helligkeit des Pixels mit Koordinaten m, n an, während k der RGB-Farbe entspricht, siehe link.*

Bemerkung. *Das Newton-Verfahren ist von fundamentaler Bedeutung in angewandter Mathematik, da es mit hoher Geschwindigkeit Lösungen nichtlinearer Systeme findet, und dabei relativ geringen Rechenaufwand hat. Allerdings haben nichtlineare Gleichungen oft mehr als eine Lösung, und dann stellt sich die Frage zu welcher Lösung das Verfahren konvergiert. Dieses Beispiel zeigt allerdings, dass das Konvergenzverhalten in hohem Maß von Startwerten abhängt, selbst bei einer so unkomplizierten Funktion wie einem Polynom.*

Richard Huber
richard.huber@uni-graz.at

function [Graphics] = my_newton_graphics (N, FOV)

```
f=@(x) x^3-1;          Df=@(x) 3*x^2;          #Creating Funktionhandles

my_eps=10^-6;          Maxiter=100;            #Additional Parameters

Solutions=[1,-1/2+i*sqrt(3)/2,-1/2-i*sqrt(3)/2];  #Analytic Solutions

#Initialize Matrices for saving results
Graphics=zeros(2*N+1,2*N+1,3);  noconvergence=zeros(2*N+1,2*N+1,3);

for n=1:2*N+1                                  #Iterating over each pixel (m,n)
  for m=1:2*N+1
    z_0=FOV*(n-N-1+i*(m-N-1))/N;               #Initial complex value

    #Executing my_newton_solver
    [z,fnew,iter,flag] = my_newton_solver (z_0,my_eps,Maxiter,f,Df);

    if flag==-1                                #Case no solution was found
      noconvergence(m,n,:)=1;
      continue;                                #Skipping to next iteration
    end

    for k=1:3
      if abs(z-Solutions(k))<10^-3;            #Which solution was found
        Graphics(m,n,k)=iter;                  #Filling Graphics
      end                                      #end if
    end                                        #end for k
  end                                          #end for m
end                                            #end for n

#Preparing results for visual presentation
Graphics(Graphics> prctile(Graphics(:),98))=prctile(Graphics(:),98);
Graphics=Graphics/max(Graphics(:));
Graphics=Graphics+noconvergence;

#Visualize results
figure();
imshow(Graphics);

endfunction
```

Richard Huber
richard.huber@uni-graz.at

Numerische Mathematik für LAK
12.Übungsblatt – 21ten Januar, 2020

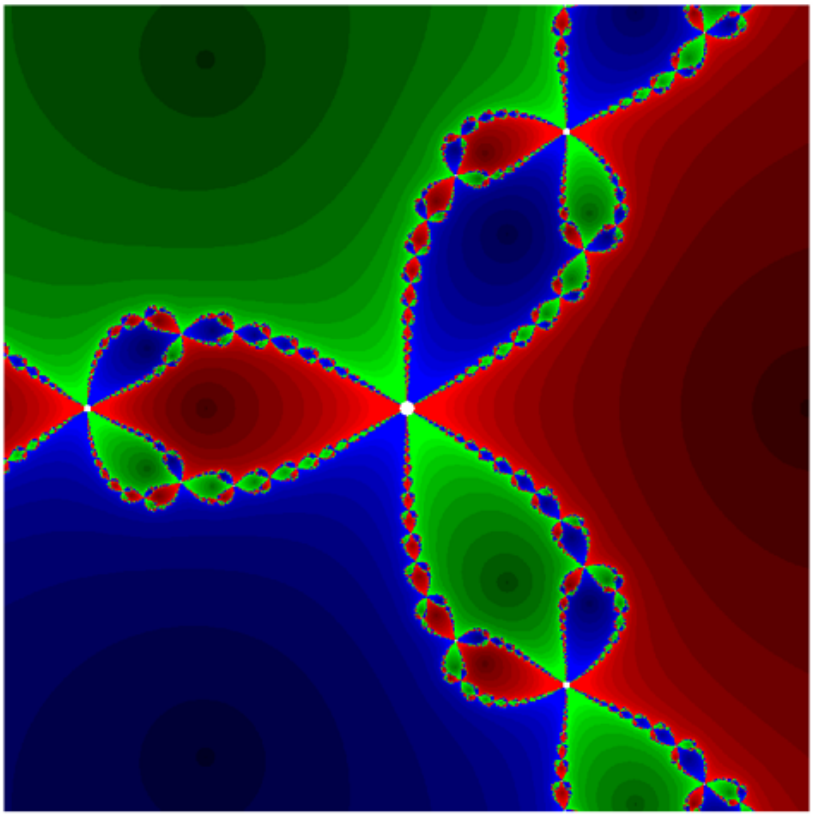

Abbildung 1: Resultat von my_newton_graphics mit $N = 500$ und FOV= 1. Da diese Berechnung einige Minuten dauert kann es sinnvoller sein mit kleinerem N zu beginnen.

University of Graz

Institut für Mathematik
und wissenschaft. Rechnen