Prof. Dr. Karsten Urban
Dipl.-Math. oec. Klaus Stolle
Institut für Numerische Mathematik
Universität Ulm

SoSe 2014
Blatt 4
13.06.2014

Angewandte Numerik 1

Abgabetermin: Freitag, 27.06.2014, **vor der Übung**

Dieses Übungsblatt hat eine Bearbeitungszeit von zwei Wochen.
Für dieses Übungsblatt gibt es 24 Theorie- und 25 Matlab-Punkte, sowie 6 Theorie- und 6 Matlab-Zusatzpunkte. Punkte, die mit einem * gekennzeichnet sind, sind Zusatzpunkte.
Die 50-Prozent-Grenzen liegen aktuell (inklusive Blatt 4) bei 45,5 Theoriepunkten und 45,5 Matlabpunkten.

Aufgabe 23 *(Programmieraufgabe: Bisektionsverfahren)* (3M+3M Punkte)

a) Schreiben Sie eine Matlab-Function `[a,b] = bisektion(a,b,f)`, die die Lösung einer Gleichung $f(x) = 0$ im Intervall $[a, b]$ bis auf zwei Nachkommastellen genau bestimmt. Rückgabewert soll das auf zwei Nachkommastellen genau bestimmte Intervall sein. Protokollieren Sie den Iterationsverlauf dadurch, dass Sie in jedem Schritt die Intervallgrenzen ausgeben.

b) Testen Sie ihr Programm an den folgenden Gleichungen und Startwerten:

$$\begin{aligned} x^2 - 2 = 0, &\quad a = 1,\ b = 3,\\ \sin(x) - \cos(2x) = 0, &\quad a = 0,\ b = 1,\\ 3\sin(x) + \sqrt{x} = 2^x, &\quad a = 0,\ b = \frac{\pi}{2},\\ x^3 - 7x^2 + 11 = 5, &\quad a = 2.7,\ b = 6.5. \end{aligned}$$

Interpretieren Sie die berechneten Ergebnisse.

Hinweis: Verwenden Sie anonyme Funktionen:
`<Funktionsname> = @(<Argumentliste>) Funktionsbeschreibung` legt eine Variable vom Typ `funktion-handle` an, die dann als Parameter an eine andere Funktion, beispielsweise Ihre Funktion `bisektion` übergeben werden kann. Für die erste Testgleichung $x^2 - 2 = 0$ legt `f1 = @(x) x^2-2` die Variable `f1` an, die Sie mit `[a,b] = bisektion(a,b,f1)` an die Matlab-Funktion übergeben können.

Aufgabe 24 *(Intervallschachtelung)* (4T Punkte)

Das Bisektionsverfahren beruht auf dem Prinzip der Intervallschachtelung:
Sei $I_0 = [a_0, b_0]$ ein Anfangsintervall. Eine Annäherung an einen Punkt $x^* \in I_0$ kann man mit Hilfe der Intervallschachtelung erreichen. Dazu definiert man iterativ $I_n = [a_n, b_n]$ mit

$$a_{n+1} = \begin{cases} a_n & , \text{falls } x^* \le \frac{b_n+a_n}{2} \\ \frac{b_n+a_n}{2} & , \text{falls } x^* > \frac{b_n+a_n}{2} \end{cases} \quad \text{sowie} \quad b_{n+1} = \begin{cases} \frac{b_n+a_n}{2} & , \text{falls } x^* \le \frac{b_n+a_n}{2} \\ b_n & , \text{falls } x^* > \frac{b_n+a_n}{2} \end{cases}$$

Wie viele Schritte benötigt das Bisektionsverfahren für das erste Beispiel aus der Aufgabe 23 (also $I_0 = [1, 3]$), um eine Genauigkeit von mindestens 10^{-8} zu erreichen? Bestimmen Sie $n \in \mathbb{N}$, sodass $|x^* - y| < 10^{-8}$ für alle $y \in I_n$ gilt.

Aufgabe 25 *(Programmieraufgabe: Vergleich verschiedener Verfahren)*
(1M+2M+2T+2M+1T+3M*+1T* Punkte)

a) Ändern Sie Ihre Matlab-Funktion `bisektion` aus Aufgabe 23 so ab, dass Sie die gewünschte Genauigkeit `tol` als Parameter übergeben können. Ferner soll ihre geänderte Matlab-Funktion `[xk,nit] = bisektion1(a,b,f,tol)` einen Vektor `xk` mit den Iterationswerten x_k aller durchgeführten Iterationsschritte und die Anzahl `nit` der benötigten Iterationen zurückgeben.

b) Schreiben Sie ein Matlab-Skript `testKonvergenz`, in dem Sie mit Hilfe Ihrer geänderten Funktion `bisektion1` den Wert von $\sqrt{2}$ auf 10 Nachkommastellen genau berechnen. Ausgangsintervall soll wieder $[1, 3]$ sein. Visualisieren Sie, wie sich der Fehler $|x_k - \sqrt{2}|$ während des Bisektionsverfahrens entwickelt. Plotten Sie dazu zu jedem Iterationsschritt (x-Achse) den jeweiligen Fehler (y-Achse). Wählen Sie zur besseren Darstellung für den Fehler eine logarithmische Skala (Matlab-Befehl `semilogy`).

c) Interpretieren Sie das Ergebnis. Liegt Konvergenz im Sinne der *Definition 5.1.1* vor? Begründen Sie Ihre Aussage.

d) Schreiben Sie eine Matlab-Funktion `[xk,nit] = sekanten(a,b,f,tol)`, die die Sekantenmethode implementiert. Ihre Funktion soll die gleichen Parameter und Rückgabewerte wie Ihre Funktion `bisektion1` haben. Ergänzen Sie Ihre Grafik aus Aufgabenteil a) um die Darstellung der jeweiligen Fehler der Sekantenmethode.

e) Interpretieren Sie das Ergebnis. Liegt Konvergenz vor?

f) Erweitern Sie Ihr Testprogramm und Ihre Grafik auch um die Regula Falsi (`[xk,nit] = regulaFalsi(a,b,f,tol)`, Parameter und Rückgabewerte analog zur Sekantenmethode) und das Newton-Verfahren (`[xk,nit] = newton(x0,f,Df,tol,maxit)`). Sie können Ihre Matlab-Funktion `[xk,nit] = newton(x0,f,Df,tol,maxit)` aus Aufgabe 28 verwenden.

g) Interpretieren Sie wiederum die Ergebnisse.

Aufgabe 26 *(Berechnen der Quadratwurzel mit dem Newton-Verfahren, Quadratische Konvergenz)*
(2T+3T Punkte)

Zur Berechnung von $\sqrt{a}$ mit einer rellen Zahl $a > 0$ soll das Newton-Verfahren auf die skalare Gleichung

$$p(x) := x^2 - a = 0$$

angewendet werden, d. h.

$$x_0 \text{ gegeben}, \quad x_{k+1} = x_k - \frac{p(x_k)}{p'(x_k)}, \quad k \geq 0.$$

a) Geben Sie die Iterationsvorschrift für die Bestimmung der Quadratwurzel an.

b) Sei $1/100 \leq a < 1$. Zeigen Sie, dass für den absoluten Fehler $\delta_k = x_k - \sqrt{a}$ gilt:

$$\delta_{k+1} = \frac{\delta_k^2}{2x_k} = \frac{\delta_k^2}{2(\delta_k + \sqrt{a})}$$

Leiten Sie daraus die quadratische Konvergenz ab: Es gibt ein $C > 0$, so dass für kleine $|\delta_k|$ gilt:

$$\delta_{k+1} \leq C\delta_k^2.$$

Geben Sie C explizit an.

Aufgabe 27 *(Newton-Verfahren für skalare Gleichungen und für Systeme)* (2T*+3T* Punkte)

a) Lösen Sie das nichtlineare Gleichungssystem $x^2 = 5$ näherungsweise, indem Sie drei Iterationen des Newton-Verfahrens durchführen. Verwenden Sie als Startwert $x_0 = 1$.

b) Anders als beim Bisektionsverfahren, der Regula Falsi und dem Sekantenverfahren können mit dem Newton-Verfahren auch Systeme nichtlinearer Gleichungen gelöst werden.

Betrachen Sie die Funktion $f : \mathbb{R}^2 \to \mathbb{R}^2$, definiert durch

$$f\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} := \begin{pmatrix} 2x_1^3 - x_2^2 - 1 \\ x_1 x_2^3 - x_2 - 4 \end{pmatrix}.$$

Lösen Sie das System von nichtlinearen Gleichungen $f(x, y) = 0$ näherungsweise durch Anwendung des Newton-Verfahrens für Systeme auf diese Funktion. Auf 10 Stellen genau lautet die exakte Lösung $x^* = (1.234274484, 1.661526467)^T$. Führen Sie zwei Newton-Iterationen durch, beginnen Sie mit dem Startwert $x^{(0)} = (1.2, 1.6)^T$. Nach zwei Newton-Iterationen erreichen Sie 4-stellige Genauigkeit.

Aufgabe 28 *(Programmieraufgabe: Newton-Verfahren und Halley-Verfahren)*
(3M+3M*+2M+3M+3T Punkte)

Das Newton-Verfahren und das Halley-Verfahren[1] berechnen jeweils für genügend glatte Funktionen eine zugehörige Nullstelle. Der Unterschied der beiden Verfahren besteht darin, dass beim Halley-Verfahren die Information der zweiten Ableitung genutzt wird.

a) Schreiben Sie eine Funktion `[x,nit] = newton(x0,f,df,tol,maxit)`, welche mit den Parametern

- `x0`: der Startwert
- `f`: die Funktion f als *function handle*
- `df`: die Jacobimatrix J_f bzw. die Ableitungsfunktion f' als *function handle*
- `tol`: die Toleranz für den Abbruch der Iteration
- `maxit`: eine obere Grenze für die Anzahl der durchgeführten Iterationen

eine Nullstelle einer gegebenen Funktion f mit Hilfe des Newton-Verfahrens berechnet. Ihre Funktion soll wieder einen Vektor `xk` mit den Iterationswerten x_k aller durchgeführten Iterationsschritte und die Anzahl `nit` der benötigten Iterationen zurückgeben. Beachten Sie, dass Ihre Funktion auch für Systeme, d. h. $f : \mathbb{R}^n \to \mathbb{R}^n$ funktionieren sollte.

b) Schreiben Sie analog eine Funktion `[x,nit] = halley(x0,f,df,d2f,tol,maxit)`, welche eine Nullstelle einer gegebenen Funktion $f : \mathbb{R} \to \mathbb{R}$ mit Hilfe des Halley-Verfahrens berechnet. Dabei ist der zusätzliche Parameter `d2f` die Funktion f'' (zweite Ableitung) als *function handle*. Die restlichen Parameter und die Rückgabewerte sind gleich wie bei der Matlab-Funktion für das Newton-Verfahren.

c) Schreiben Sie ein Testskript `testRoots.m`, welches beide Verfahren für folgende Funktionen

(i) $f(x) = \cos x - x$, $x_0 = 0.1$
(ii) $f(x) = e^x - 1$, $x_0 = 0.5$

testet und diese inklusive der berechneten Nullstellen dann in einer Grafik darstellt. Verwenden Sie `tol = 1e-10.`

d) Schreiben Sie ein Testskript `compareNewtonHalley.m`, welches für $f(x) = \arctan x$ und 200 Startwerte $x_0 \in [-10, 10]$ die Anzahl der Iterationen beider Verfahren bestimmt und diese Beobachtung mittels einer Grafik visualisiert (x-Achse: Startwert, y-Achse: Anzahl der Iterationen). Verwenden Sie `tol = 1e-10.`

[1] `beispielsweise http://de.wikipedia.org/wiki/Halley-Verfahren`

e) Was fällt auf? Wie interpretieren Sie die Anzahl der Iterationen für betragsmäßig große Startwerte. Erklären sie beispielhaft an einigen Startwerten den jeweiligen Verlauf der Iteration.

Aufgabe 29 *(Programmieraufgabe: Newton-Verfahren für Systeme)* (2M Punkte)

Haben Sie bei Aufgabe 28 berücksichtigt, dass das Newton-Verfahren auch für Systeme funktionieren soll? Wenden Sie Ihre Matlab-Funktion `newton` auf das nichtlineare Gleichungssystem aus Aufgabe 27 b) an und verifizieren Sie die Lösung.

Aufgabe 30 *(Programmieraufgabe: gedämpftes Newton-Verfahren)* (3M+1M+1T Punkte)

a) Schreiben Sie jetzt eine Funktion `[x,nit] = newtonGedaempft(x0,f,df,tol,maxit)`, die eine Nullstelle einer gegebenen Funktion $f : \mathbb{R} \to \mathbb{R}$ mit Hilfe des gedämpften Newton-Verfahrens berechnet. Die Parameter und die Rückgabewerte sind gleich wie bei der Matlab-Funktion für das Newton-Verfahren.

b) Ergänzen Sie Ihr Testskript `compareNewtonHalley.m` aus Aufgabe 28 um das gedämpfte Newton-Verfahren.

c) Interpretieren Sie die neue Grafik und den Iterationsverlauf des gedämpften Newton-Verfahrens.

Aufgabe 31 *(Fixpunktiteration)* (2T+3T+2T+1T Punkte)

Gesucht ist die positive Nullstelle der Funktion $f : \mathbb{R} \to \mathbb{R}$ mit

$$f(x) = x^5 - x - 1.$$

Es ist bekannt, dass f im Intervall $[0, 2]$ eine eindeutige positive Nullstelle hat.

a) Formulieren Sie dieses Problem als äquivalentes Fixpunktproblem.

b) Zeigen Sie, dass die von Ihnen gewählte Fixpunktiteration im Intervall $[0, 2]$ einen eindeutigen Fixpunkt hat. Sie dürfen dabei die Folgerung 5.6.3 benutzen.

c) Wählen Sie als Startpunkt für die Iteration den Punkt $x_0 = 2$. Wie viele Iterationen müssen Sie durchführen, um die Nullstelle mit einer Genauigkeit von 10^{-10} zu berechnen?

d) Gibt es auch eine Iterationsvorschrift, die nicht konvergieren würde? Begründen Sie Ihre Aussage.

Hinweise:
Die Programmieraufgaben sind in Matlab zu erstellen. Senden Sie alle Files in einer E-mail mit dem Betreff **Loesung-Blatt04** 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.