Martin-Luther-Universität Halle-Wittenberg
Institut für Numerische Mathematik
Prof. Dr. M. Arnold, B. Burgermeister
Wintersemester 2005/2006

Numerische Mathematik II

14. Übungsblatt, Besprechung am 01.02.2006

`http://sim.mathematik.uni-halle.de/~arnold/courses/SoS05.num/`

Aufgabe 1. *Nullstellen von Polynomen*

Bestimmen Sie alle Nullstellen des MATLAB-Polynoms `p=poly([0.1:0.1:1])`

a) mit Hilfe des Newton-Verfahrens: Beginnen Sie mit einem ausreichend großen Startwert $x_0 = 10$ und bestimmen Sie mit Hilfe des Newton-Verfahrens die größte Nullstelle des Polynoms. Verwenden Sie als Fehlerschranke 10^{-12}. Spalten Sie die so gefundene Nullstelle mittels Polynomdivision ab und wiederholen Sie das Verfahren mit dem übriggebliebenen Polynom bis Sie alle Nullstellen gefunden haben. (*Hinweis:* benutzen Sie die MATLAB-Funktionen zum Auswerten, Ableiten und Dividieren von Polynomen.)

b) als Eigenwerte der Frobenius-Matrix (vergleiche Übungsblatt 1) mit dem MATLAB-Befehl `eig`.

Vergleichen Sie das Ergebnis jeweils mit der exakten Lösung $(0.1, 0.2, \ldots, 1)$.

Aufgabe 2. *Nichtlineares Ausgleichsproblem*

Gegeben seien die folgenden Messwerte (t_i, b_i) einer biochemischen Reaktion: (vergleiche Deuflhard/Hohmann, Numerische Mathematik I, Beispiel 4.18)

t	b	t	b	t	b	t	b	t	b
6	24.19	42	57.39	78	52.99	114	49.64	150	46.72
12	35.34	48	49.56	84	53.83	120	57.81	156	40.68
18	43.43	54	55.60	90	59.37	126	54.79	162	35.14
24	42.63	60	51.91	96	62.35	132	50.38	168	45.47
30	49.92	66	58.27	102	61.84	138	43.85	174	42.40
36	51.53	72	62.99	108	61.62	144	45.16	180	55.21

(Diese Werte stehen auch auf der Website zur Vorlesung bereit.)

Bestimmen Sie mittels Gauß-Newton-Verfahren optimale Parameter x_1, x_2, x_3 für die Funktion

$$\varphi(x,t) := x_1 \exp(-(x_2^2 + x_3^2)t)\frac{\sinh(x_3^2 t)}{x_3^2}$$

so dass der mittlere quadratische Fehler $\sum_i \|\varphi(x,t_i) - b_i\|_2^2$ für die oben angegebenen Messdaten minimal wird. Benutzen Sie als Startwert jeweils $x_1 = 4, x_2 = 0.055, x_3 = 0.21$.

Bitte wenden!

Aufgabe 3. *Newton-Verfahren*

Zur Bestimmung der Nullstellen einer Funktion $F : D \subset \mathbb{R}^n \to \mathbb{R}^n$ werden in der Praxis oft das Newton-Verfahren und seine Varianten eingesetzt. Für die gesamte Aufgabe gelte $RTOL = ATOL = 10^{-6}$ und die Funktion $F(x)$ sei ausreichend oft differenzierbar. Die Funktion F sei jeweils in einer geeigneten `MATLAB`-Funktion `function f=fname(x)` enthalten und liefere in `f` den Funktionswert zurück. Zur Auswertung von Funktionen deren Name als Parameter übergeben wurde, kann der `MATLAB`-Befehl `feval` benutzt werden.

a) Beim Newton-Verfahren wird die Jacobi-Matrix F_x benötigt. Diese soll hier in einer `MATLAB`-Funktion `jacobi.m` durch numerische Differentiation über mehrfache Auswertung der Funktion `fname` bestimmt werden. Mit

$$\frac{\partial F(x)}{\partial x_i} \approx \frac{F(x + he_i) - F(x)}{h}$$

setzt sich die Jacobi-Matrix spaltenweise zusammen. Wählen Sie $h = 10\sqrt{\text{eps}}$ (vgl. Übungsblatt 4, Aufgabe 2). Verwenden Sie für diese Funktion den Programmkopf

```
function jac=jacobi(f,x,f0)
```

Dabei bezeiche `f0` den Funktionswert $F(x)$ um ihn nicht erneut berechnen zu müssen.

b) Schreiben Sie eine `MATLAB`-Funktion `newton.m`, die ein nichtlineares Gleichungssystem $F(x) = 0$ mit Hilfe des Newton-Verfahrens löst. Die benötigte Jacobi-Matrix wird von der Funktion `jacobi.m` aus a) geliefert. Um eine Abschätzung über den gesamten Rechenaufwand zu bekommen, soll eine Variable `zaehler` die Anzahl der Funktionsaufrufe von $F(x)$ enthalten, da diese (neben der Lösung des Systems $F_x(x^{(k)}) \cdot p^{(k)} = -F(x^{(k)})$) den Hauptaufwand des Verfahrens ausmachen. Damit auch in der Funktion `jacobi.m` mitgezählt werden kann, definiere man die Variable als `global` (siehe `MATLAB`-Hilfe). Der Programmkopf der Newton-Funktion ist wie folgt zu gestalten:

```
function [loes,zaehler]=newton(f,x0,RTOL,ATOL)
```

Als Abbruchkriterium der Iteration verwende man $\left\|p^{(k)}\right\|_2 \leq RTOL \left\|x^{(k)}\right\|_2 + ATOL$

Brechen Sie das Programm mit einer Fehlermeldung ab, wenn nach mehr als 100 Iterationen keine ausreichend genaue Lösung gefunden wurde.

c) Berechnen Sie die Nullstelle von $\arctan(x)$ mit den Startwerten $x^{(0)} = 1.3$ und $x^{(0)} = 1.4$. Für arctan braucht kein separates `m`-File geschrieben werden, da es schon fest in `MATLAB` enthalten ist. Der Funktionsaufruf von `newton.m` sieht dann beispielsweise so aus:

```
>> [loes,zaehler]=newton(@atan,1.3,1e-6,1e-6)
```

Wie ist das Ergebnis zu interpretieren?

d) Zur Vergrößerung der Konvergenzgebiets kann das sogenannte *gedämpfte Newton-Verfahren* benutzt werden. Berechnen Sie dazu genau wie beim gewöhnlichen Newton-Verfahren $p^{(k)} = -(F_x(x^{(k)}))^{-1}F(x^{(k)})$ und setzen Sie dann $x^{(k+1)} = x^{(k)} + \lambda p^{(k)}$ mit $\lambda \in (0, 1]$ so, dass die Folge $(\left\|F(x^{(k)})\right\|)$ streng monoton fallend ist. Beginnen Sie dazu mit $\lambda = 1$ und testen Sie in jedem Schritt, ob $\left\|F(x^{(k)} + \lambda p^{(k)}\right\| < \left\|F(x^{(k)})\right\|$ gilt. Falls diese Bedingung nicht erfüllt ist, verkleinern Sie λ (z.B. durch Halbieren) solange, bis die Bedingung erfüllt ist. Brechen Sie die Berechnung ab, wenn λ zu klein wird ($\lambda < 10^{-4}$).

Implementieren Sie das gedämpfte Newton-Verfahren als Funktion

```
function [loes,zaehler]=newtond(f,x0,RTOL,ATOL)
```

Wie verhält sich das gedämpfte Newton-Verfahren mit der Funktion $\arctan(x)$ und $x^{(0)} = 1.4$ oder $x^{(0)} = 100$? Protokollieren Sie dazu auch die Schrittweiten λ.