HUMBOLDT-UNIVERSITÄT ZU BERLIN
INSTITUT FÜR MATHEMATIK
DR. L. LEHMANN; DR. ANTOINE LAURAIN, DIPL.MATH. TORSTEN BOSSE

Übungsaufgaben zur Vorlesung Mathematikorientierte Computernutzung (SS 11) Serie 5

Abgabe bis 27. 6. 2011

Aufgabe 5.1: Ableitungen B12

Sei $f : \mathbb{R} \to \mathbb{R}$ eine einmal stetig differenzierbare Funktion auf dem kompakten Intervall $[a, b] \subset \mathbb{R}$. Die Ableitung der Funktion f kann durch den Grenzwert des zentralen finiten Differenzenquotienten

$$f'(x_0) = \lim_{h \to 0} \frac{f(x_0 + h) - f(x_0 - h)}{2h}$$

an der Stelle x_0 approximiert werden.

a) Schreiben Sie ein Programm, welches für ein gegebenes $h > 0$ und einen Punkt x_0 den obigen Differenzenquotienten berechnet. Hierbei sollte die Funktionsauswertung von f in einer extra Methode geschehen (vergleiche Serie 2, Aufgabe 2).

b) Implementieren Sie desweiteren, sofern noch nicht geschehen, den ersten Teil der Aufgabe 2 von Serie 2, mit der folgenden Änderung

$$F_n(a, b) = \sum_{i=0}^{n-1} \frac{|f(x_i)|}{n}\,.$$

Somit lässt sich nun auch der absolute Flächeninhalt von Funktionen mit negativen Funktionswerten bestimmen.

c) Nutzen Ihre Programme, um sich die Ableitung und den Funktionswert an den äquidistanten Punkten $a = x_0 < x_1 < \ldots x_{N-2} < x_{N-1} = b$ von der Funktion $f(x) = \sin(x)$ in dem Intervall $[0, 2\pi]$ auszugeben. Berechnen Sie außerdem die Fläche des Integrals der Funktion über den Intervallen $[0, x_i] \subseteq [0, 2\pi]$. Lassen Sie sich ihre Resultate in einer Tabelle ausgeben:

```
#    x_i        f(x_i)       f'(x_i)        Integral[0,x_i]
0.000000      0.000000      1.000000              0.000000
    ...           ...           ...                   ...
```

Wählen Sie $N = 101$, $h = 10e - 8$ und nutzen Sie bei der Integration eine Anzahl von 1000 Stützstellen.

Hinweis: Sie können Ihre Ergebnisse visuell überprüfen. Nutzen Sie hierfür die Ausgabeumleitung nach dem Schema `java prog > output.txt`, um sich Ihre Ausgabe (Tabelle) in eine Datei zu speichern und wenden Sie das von uns gegebene gnuplot Skript[1] an.

[1] siehe COMA-Homepage

Aufgabe 5.2: Polynom-Division B8

Seien $p(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x^1 + a_0$ und $q(x) = b_m x^m + b_{m-1}x^{m-1} + \cdots + b_1 x^1 + b_0$ zwei gegebene Polynome über $\mathbb{R}$ mit rationalen Koeffizienten und $m < n \in \mathbb{N}$. Schreiben Sie ein Programm, welches die Polynomdivision

$$p(x)/q(x) = d(x) + r(x)/q(x), \qquad \deg r(x) < \deg q(x)$$

berechnet. Bestimmen Sie damit den Quotienten $d(x)$ für die Polynome

$$p(x) = 1.0x^6 - 3.031x^5 - 7.697x^4 + 9.191x^3 + 15.245x^2 - 6.641x^1 - 8.208$$
$$q(x) = 1.0x^3 - 6.537x^2 + 11.253x^1 - 5.730$$

Hinweis: Nutzen Sie vier double-Arrays der Länge n, um die Koeffizienten der benötigten Polynome zu speichern.

Aufgabe 5.3: Polynom-Arithmetik B12

Verwenden Sie die Polynomarithmetik aus der Klasse DX.java.

a) Die Tschebyscheff-Polynome

$$T_n(x) = \cos(n \arccos(x)),\ (-1 \le x \le 1)$$

erfüllen die Rekursionsformel

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x).$$

Berechnen Sie diese Polynome mittels der Rekursionsformel und geben Sie die ersten 12 Polynome mittels der Ausgabefunktionen der DX-Klasse aus.

b) Die Bernsteinpolynome n-ter Stufe sind durch die Formeln

$$B_i^n(x) = \binom{n}{i} x^i (1-x)^{n-i},\ i = 0, \ldots, n$$

gegeben und erfüllen die Rekursionsformeln

$$B_i^n(x) = (1-x)B_i^{n-1}(x) + xB_{i-1}^{n-1}(x)$$

mit $B_0^0(x) = 1$. Berechnen Sie diese Polynome mittels der Rekursionsformel. Überprüfen Sie die Relation $\sum_{i=0}^{n} B_i^n(x) = 1$ für $n = 20$.

Aufgabe 5.4: Gamma–Funktion B8

Die Γ-Funktion wird wie folgt definiert:

$$\Gamma(x) = \lim_{n\to\infty} \frac{n!n^x}{x(x+1)\cdots(x+n)},\ x \in \mathbb{R}.$$

Überprüfen Sie, für welche Werte von n der obige Bruch eine vernünftige Näherung für $x \in [1,6]$ darstellt. Zur Probe: Wenn $x = m$ eine natürliche Zahl ist, gilt $\Gamma(m) = (m-1)!$.

Aufgabe 5.5: Matrix–Faktorisierung B16

Schreiben Sie eine Klasse *Cholesky*, welche für eine gegebene symmetrische positiv definite Matrix $A \in \mathbb{R}^{n\times n}$ eine Choleskyfaktorisierung[2] $LL^\top$ berechnet. Dazu sollte die Klasse *Cholesky* eine int Variable n, ein 'public' Array x der Größe n und ein Matrix–Array L der Größe $n \times n$ beinhalten. Desweiteren sollte die Klasse über folgende Methoden verfügen:

a) public void *CholeskyFactorize*(double $A[][]$, int Adim)

b) public void *BackwardSolve*(double $b[]$, int ndim)

Hierbei soll in der Methode *CholeskyFactorize* das Matrix–Array A mit dazugehöriger Dimension mittels des Cholesky Algorithmus faktorisiert werden. Dazu kopieren Sie zuerst die Einträge von A nach L, um bei der Faktorisierung (von L) die ursprüngliche Matrix A nicht zu ändern, und speichern Sie die zugehörige Dimension *Adim* in n.

Benutzen Sie nach erfolgreicher Faktorisierung die Methode *BackwardSolve* um das Gleichungssystem

$$Ax = LL^\top x = b$$

zu lösen. Die Eingabe der rechten Seite b und der dazugehörigen Dimension geschieht hierbei mittels Übergabe bei Funktionsaufruf. Die Lösung des Gleichungssystems erhalten Sie durch zweimaliges Rückwärtseinsetzen (beachten Sie, dass L bzw. $L^\top$ Dreicksform besitzt). Speichern Sie die Lösung in dem Array x.

Testen Sie die Cholesky–Faktorisierung an der Matrix

$$A = \begin{bmatrix} 16 & -4 & -8 & 20 \\ -4 & 5 & -6 & -15 \\ -8 & -6 & 21 & 7 \\ 20 & -15 & 7 & 68 \end{bmatrix} \quad \text{und} \quad b = \begin{bmatrix} -12 \\ -11 \\ 39 \\ -4 \end{bmatrix}.$$

Der Cholesky–Faktor L zu dieser Matrix sowie der Lösungsvektor x von $Ax = b$ beinhalten nur ganzzahlige einstellige Einträge.

Hinweis: Versuchen Sie, Ihre Implementierung möglichst robust zu schreiben, d. h., benutzen Sie die Variable n, um in BackwardSolve zu verifizieren, ob die Dimensionen übereinstimmen und ob die vorherige Cholesky Zerlegung erfolgreich war – z.B. durch setzen von $n = -1$, falls der Cholesky–Algorithmus abbricht. Wann kann das passieren?

[2] siehe z.B. `http://de.wikipedia.org/wiki/Cholesky-Zerlegung`

Aufgabe 5.6: Gesetz der großen Zahlen B20

Gegeben seien 5 normierte sechsseitige Würfel, auf deren Seiten die Zahlen 1 bis 6 stehen. Berechnen Sie zunächst theoretisch, wie hoch die Wahrscheinlichkeiten p_i^*, $(i = 1, \ldots, 5)$ sind, bei einem gleichzeitigen Wurf aller 5 Würfel, einen

1. Dreierpasch (mindestens 3 Würfel zeigen die selbe Zahl)
2. Viererpasch (mindestens 4 Würfel zeigen die selbe Zahl)
3. Kniffel (alle 5 Würfel zeigen die selbe Zahl)
4. eine kleine Straße (vier Würfel zeigen die Zahlen {1,2,3,4},{2,3,4,5} oder {3,4,5,6})
5. eine große Straße (die Würfel zeigen die Zahlen {1,2,3,4,5} oder {2,3,4,5,6})

zu werfen. Verifizieren Sie die theoretischen Ergebnisse durch ein numerisches Experiment. Implementieren Sie dazu ein Programm, welches eine natürliche Zahl $n \in \mathbb{N}$ erhält. Innerhalb des Programms sollen nun n Würfe (mit allen Würfeln) simuliert werden. Nutzen Sie hierfür die *Math.random()* Funktion um sich in jedem Wurf für jeden Würfel eine Zufallszahlen zwischen 1 und 6 zu generieren. Lassen Sie das Programm nun zählen, wie oft ein Dreierpasch, Viererpasch, Kniffel, eine kleine Straße und eine große Straße geworfen wurde und geben Sie am Ende die Verhältnisse

1. $p_1(n) = \dfrac{\sharp \text{ geworfene Dreierpasch}}{n}$
2. $p_2(n) = \dfrac{\sharp \text{ geworfene Viererpasch}}{n}$
3. $p_3(n) = \dfrac{\sharp \text{ geworfene Kniffel}}{n}$
4. $p_4(n) = \dfrac{\sharp \text{ geworfene kleine Straße}}{n}$
5. $p_5(n) = \dfrac{\sharp \text{ geworfene große Straße}}{n}$

aus. Erzeugen Sie als Ausgabe eine Gegenüberstellung der theoretischen und numerischen Werte. Für hinreichend großes n entsprechen die theoretischen den numerischen Ergebnissen, d. h.,

$$\lim_{n \to \infty} p_i(n) = p_i^*, \ (i = 1, \ldots, 5).$$

Hinweis: Benutzen Sie ein int-Array, um das Ergebnis eines Wurfs zu speichern. Dieses Array kann mittels *java.util.Arrays.sort* geordnet werden und erleichtert somit die darauffolgende Auswertung.

Aufgabe 5.7: Regression B12

Gegeben sei eine *Datenwolke* (x, y) von $n \in \mathbb{N}$ Messpunkten $(x_i)_{i=1}^n \in \mathbb{R}^n$ und den dazugehörigen Messwerten $(y_i)_{i=1}^n \in \mathbb{R}^n$. Desweiteren sei $\vec{f}(x, \beta) = (f(x_i, \beta)) \in \mathbb{R}^n$ der Vektor von Funktionswerten einer Modellfunktion $f : \mathbb{R} \times \mathbb{R}^m \to \mathbb{R}$, welche von dem entsprechenden Messpunkt x_i und den m Parametern $\beta = (\beta_j)_{j=1}^m \in \mathbb{R}^m$ abhängt. Zu bestimmen sind nun diejenigen Parameter β, so dass die Funktion $\vec{f}$ die gegebenen Daten optimal approximiert, d. h., finden Sie die Lösung des Problems

$$\min_{\beta \in \mathbb{R}^m} \Psi(\beta) := \sum_{i=1}^{n} \|y_i - f(x_i, \beta)\|^2$$

Sofern die Modellfunktion $f : \mathbb{R} \times \mathbb{R}^m \to \mathbb{R}$ linear in β ist, in anderen Worten

$$f(x_i, \beta) = \sum_{j=1}^{m} \beta_j \varphi_j(x_i) \;\text{ mit } \varphi_j : \mathbb{R} \to \mathbb{R},\; j = 1, \ldots, m\,,$$

kann die Lösung des Minimierungsproblem durch das Lösen des linearen Gleichungssystems

$$C^\top C\beta - C^\top y = 0$$

bezüglich β bestimmt werden. Hierbei ist $C \in \mathbb{R}^{n \times m}$ die Matrix mit den Einträgen

$$C_{ij} = \frac{\partial f(x_i, \beta)}{\partial \beta_j} = \varphi_j(x_i).$$

a) Schreiben Sie ein Programm, welches Ihnen für eine gegebene Datenwolke und eine (lineare) Modellfunktion die beste Approximation bestimmt.

b) Wählen Sie als Modellfunktion die Polynome

$$f(x_i, \beta) = \sum_{j=1}^{m} \beta_j x^{j-1} \implies \varphi_j(x_i) = x_i^{j-1}$$

vom Grade $m - 1$ und testen Sie das Programm für $m = 1, 2, \ldots, 6$. Die Datenwolke ist gegeben durch

x	1	2	3	4	5	6	7
y	2.0	2.5	2.5	3.4	3.7	6.6	3

c) Benutzen Sie die DM Klasse aus dem Paket HUMath bzw. eine Cholesky–Faktorisierung zum Lösen des Gleichungssystems.

Aufgabe 5.8: Differentialgleichungen B12

Gegeben sei die folgende Differentialgleichung mit Anfangsbedingung

$$y'(x) = f(x, y(x)) = y(x), \quad x \in [0,1], \qquad y(0) = 1, \tag{1}$$

deren exakte Lösung $y(x) = e^x$ ist. Im Folgenden sollen Approximationsverfahren für die Lösung von solchen Anfangswertaufgaben untersucht werden. Dabei wird eine Folge von Näherungswerten y_k für die exakte Lösung $y(x_k)$ in einer Folge von Argumenten x_k erzeugt. Betrachten Sie hierfür eine äquidistante Diskretisierung $0 = x_0 < x_1 < \cdots < x_m = 1$ des Intervalls $[0,1]$. Die Schrittweite der Diskretisierung ist die Länge $h = h(m) := 1/m$ der Intervalle $[x_i, x_{i+1}]$.

a) Schreiben Sie ein Programm, welches (1) mit Hilfe des expliziten Eulerverfahrens löst. Berechnen Sie dazu die Approximation y^E der exakten Lösungstrajektorie durch folgende Rekursion:

$$y^E_{k+1} = y^E_k + hf(x_k, y^E_k) \text{ mit } x_k = x_0 + kh, \quad k = 0, 1, \ldots, m-1 \,,$$

ausgehend von dem Startpunkt $y^E_0 = y(x_0) = y(0) = 1$.

b) Schreiben Sie außerdem ein Programm, welches das Verfahren von Heun zur Lösung von (1) benutzt. Die Approximation y^H wird bei diesem Verfahren durch die zweistufige Rekursion

$$\begin{aligned} \tilde{y}^H_{k+1} &= y^H_k + hf(x_k, y^H_k), \\ y^H_{k+1} &= y^H_k + \frac{h}{2}\left(f(x_k, y^H_k) + f(x_{k+1}, \tilde{y}^H_{k+1})\right), \end{aligned}$$

$k = 0, \ldots, m-1$, definiert. Startpunkt ist wieder $y^H_0 = 1$.

c) Vergleichen Sie die Fehler zwischen den beiden Verfahren, d.h., betrachten Sie in beiden Fällen den maximalen Fehler $\max_{i=0,\ldots,m} |y_i - y(x_i)|$ zwischen der approximierten und der exakten Lösung an den Diskretiserungspunkten. Untersuchen Sie die Fehler für verschieden feine Diskretisierungen $m = 10, 100, 1000$.