Probeklausur Numerisches Programmieren

Wintersemester 17/18, MUSTERLÖSUNG

Hendrik Möller

28.02.2018

Der Autor übernimmt keine Garantie über die Korrektheit der Aufgaben oder Lösungen.

Des Weiteren ist diese Klausur nur als ein "Vorschlag" von möglichen Aufgaben(-typen) anzusehen, wie sie in der eigentlichen Klausur drankommen könnten. Ein Bezug oder Ähnlichkeit zu vorhandenen Aufgaben sind vollkommen willkürlich.

- Schreiben Sie nicht mit Bleistift oder in roter/grüner Farbe!
- Als Hilfsmittel ist einzig und allein ein handschriftlich, beidseitig beschriebenes Blatt DIN A4 mit eigenen Notizen erlaubt (keine Ausdrucke, keine Kopien).
- Die vorgesehene Arbeitszeit beträgt 90 Minuten
- In dieser Klausur können Sie insgesamt 45 Punkte erreichen.
  Zum Bestehen werden 20 Punkte benötigt.
- Hinweis: Nach Auffassung des Autors ist diese Klausur schwerer als die "richtige" Klausur sein wird.
- Die einzelnen Teilaufgaben sind unabhängig voneinander lösbar. Sollten Sie zu einer Teilaufgabe keine Lösung finden, so können Sie diese Teilaufgabe also einfach überspringen und mit der nächsten Teilaufgabe fortfahren.

1 . Aufgabe: Gleitkommazahlen [4 + 2 + 2.5 Pkte]

In dieser Aufgabe gehen wir von einer (fiktiven) Darstellung durch Gleitkommazahlen 8-Bit-$D_{B,t}$ aus. $D_{B,t}$ entspricht der IEEE Form mit Ausnahme von folgender Bit-Verteilung und Regeln:

- Der Exponent E hat nur 4 Bits (also sind hier Exponenten von -7 bis 8 möglich)
- Für die normalisierte Mantisse M stehen die letzten 4 Bits zur Verfügung
- Die erste Eins der Mantisse muss abgespeichert werden

(a) Wandle folgende Zahlen in die 8-Bit-$D_{B,t}$ Darstellung um:

1. 6_{10}
2. $9/8$
3. 100111_2

Hier ist das "|" nur als sichtbare Trennung zwischen Bereichen geschrieben (normal existieren die natürlich nicht!)

1. $6_{10} = 0110_2 = 2^2 \cdot 1.10_2 = 1001|1100$
2. $9/8 = 1.125_{10} = 1.001_2 = 2^0 \cdot 1001_2 = 0111|1001$
3. $100111_2 = 2^5 \cdot 1.00111_2 \approx 2^5 \cdot 1.010_2 = 1100|1010$

(b) Was für Probleme neben Sinnlosigkeit und der abgespeicherten Eins hat Darstellungsform aus Teilaufgabe (a)?

Kein Vorzeichen Bit: Positive und Negative Zahlen werden genau gleich dargestellt.

(c) Die Zahlen $x = 13$ und $y = 13,375$ werden in eine Operation geworfen. Es wird die Rechnung $z = x - y$ ausgeführt. Der Computer spuckt $z = 0$ als Ausgabe aus. Eingaben werden zunächst in die 8-Bit-$D_{B,t}$ umgewandelt, bevor mit ihnen gerechnet wird.

Was ist passiert? Zeige deine Theorie, indem du einen Rechenweg angibst.

$13 = 1101_2 = 1010|1101$ $13.375 = 1101.011_2 \approx 1010|1101$ In der Gleitkommadarstellung rundet sich 13.375 zu 13. Durch die Subtraktion entsteht dann logischerweise die null.

2 . Aufgabe: Quadratur [2.5 + 4 + 2 Pkte]

(a) Wir nehmen an, dass vier Stützstellen an den festgelegten Koordinaten $x_0 = 0$, $x_1 = 2$, $x_2 = 3$ und $x_3 = 4$ einer unbekannten Funktion gegeben sind. Argumentiere, ob sich die Gauß Quadratur und das Romberg-Verfahren hier geeignet wären.

Keines eignet sich perfekt. Gauß-Quadratur nicht, weil die x-Koordinaten festgelegt sind, und Gauß-Quadratur rechnet einen ja die Koordinaten aus, die man verwenden sollte. Romberg eignet sich in Theorie schon, praktisch können wir nur Trapezsummen mit zwei unterschiedlichen Schrittweiten berechnen.

(b) Untersuchen wir eine uns unbekannte Funktion $f(x)$. Wir haben folgende Grafik

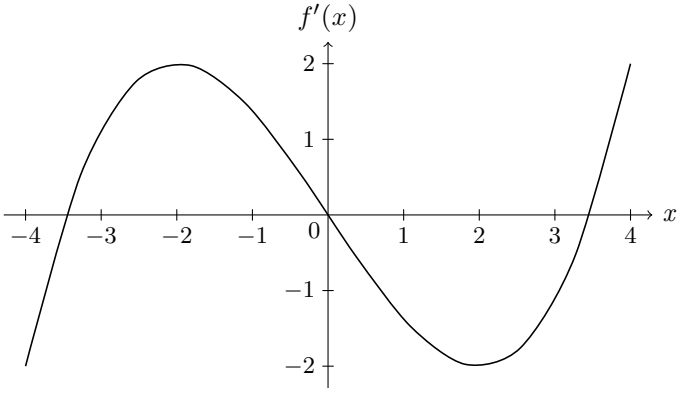

Abbildung 1: Gegebene Informationen über $f'(x)$

gegeben und möchten die Funktion auf ihre Kondition überprüfen. Erläutere für die x-Positionen $x_0 = -2$, $x_1 = 0$, $x_2 = 2$ und $x_3 = 4$ jeweils, ob sie (verglichen mit den anderen Positionen) einen gut oder schlecht konditionierten Bereich der Funktion $f(x)$ entspricht.

schlecht, gut, schlecht und schlecht konditioniert, wenn man die Punkte von links nach rechts durchgeht. Der Graph zeigt die Ableitungsfunktion. Dementsprechend stehen betragsmäßig hohe Funktionswerte für schlechtere Konditionen.

(c) Mit den x-Koordinaten aus Teilaufgabe (b) als Stützstellen, berechne die Fläche unter $f'(x)$ im Intervall $[0, 4]$ mit klassischen Quadraturregeln bestmöglich. Gib einen Rechenweg an.

$$Q_{SS}(f') = 4 \cdot \frac{0 - 8 + 2}{6} = -4$$

3 . Aufgabe: Interpolation [3 + 1 + 1.5 + 1.5 Pkte]

Es seien folgende Stützpunkte gegeben:

$$p_0 = (-1|2); p_1 = (0|1); p_2 = (1|2)$$

(a) Berechnen sie das Interpolationspolynom und geben sie das Ergebnis an der Stelle $x = 3$ an. Nutzen sie hierfür eine beliebige sinnvolle Kombination aus polynomiellen Basisfunktionen.

$$\left(\begin{array}{ccc|c} 1 & -1 & 1 & 2 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 2 \end{array}\right) = \left(\begin{array}{ccc|c} 1 & 0 & 0 & 1 \\ 0 & -1 & 1 & 1 \\ 0 & 0 & 2 & 2 \end{array}\right) \longrightarrow p(x) = x^2 + 1 \longrightarrow p(3) = 10$$

(b) Ein Kollege Ü. behauptet, er löst Teilaufgabe (a) lieber, indem er das Newton-Verfahren anwendet. Sein Grund; ein genaueres Ergebnis hat bei ihm höchste Priorität. Erläutere, ob das sinnvoll ist oder nicht.

Quatsch, Interpolationspolynome sind eindeutig. Ergebnisse von Newton Verfahren besitzen diesselbe Genauigkeit wie Polynom-Basisfunktionen. So würde er nur länger rechnen.

(c) Derselbe Kollege Ü. zeigt einem seine Rechnung, die er mit Aitken-Neville angefertigt hat. Dabei seht ihr, dass in seiner Tabelle seine Stützstellen nicht in numerisch aufsteigender Reihenfolge angeordnet sind. Was sagt ihr dazu? Markiert in der Berechnungsfunktion, um eure These zu unterlegen.
Berechnungsformel von Aitken-Neville:

$$p[i,k] := p[i,k-1] + \frac{x - x[i]}{x[i+k] - x[i]} \cdot (p[i+1,k-1] - p[i,k-1])$$

Reihenfolge in der Tabelle ist völlig egal, da in der Formel die Relation der x-Werte einberechnet wird (Der Teil unter dem Bruchstrich ist dafür verantwortlich)

(d) Ein Tag später kommt der Kollege Ü. wieder. Er hat seine ultimative Antwort: Spline-Interpolation. Für die einzelnen Splines hat er definiert:

$$p_0(t_0(x)), x \in [-1, 1]$$

$p_0(t_0(x))$ sei korrekt ausgerechnet worden und es sollen die kubischen Basispolynome, die im Intervall $[0, 1]$ berechnet worden sind, dafür genutzt werden.

Definiere $t_0(x)$ sinnvoll und erläutere kurz, warum Spline-Interpolation nicht vernünftig genutzt wurde.

$$t_0(x) = \frac{1}{2}x + 0.5$$

Es ist total unsinnvoll, weil er das ganze nur mit einem Spline macht. Von stückweiser Interpolation kann man hier noch nicht reden. Sinnvoll wäre es, wenn er mindestens zwei Splines erstellt hätte.

4 . Aufgabe: Iterative Verfahren [3.5 + 2 + 1.5 Pkte]

Es sei eine Matrix A gegeben, welche die Eigenwerte $\lambda_0 = 1$ und $\lambda_1 = 4$ besitzt.

(a) Es seien die Eigenvektoren

$$v_0 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, v_1 = \begin{pmatrix} 3 \\ 1 \end{pmatrix}$$

gegeben. Dabei sei v_0 der Eigenvektor zum Eigenwert λ_0 und v_1 zu λ_1. Berechne die Matrix A.

Beide Eigenwerte mit ihren Vektoren und der Formel $Av = \lambda v$ aufstellen, Gleichungssystem lösen.
Ergebnis:

$$\begin{pmatrix} 1 & 9 \\ 0 & 4 \end{pmatrix}$$

(b) Berechne die Konvergenzrate q_0 mit einem Shift von $\mu_0 = 0$ und q_1 mit einem Shift von $\mu_1 = 2$. Gegen die Eigenvektoren welcher Eigenwerte konvergiert die Power Iteration bei beiden Shifts jeweils? Vergleiche die Konvergenzraten. Was kann man daraus folgern (bezüglich Shift und Konvergenzgeschwindigkeit)?

$$q_0 = \frac{1}{4}; q_1 = \frac{1}{2}$$

Mit beiden Shifts konvergiert das Verfahren gegen v_1. Folgern kann man, dass sich ein Shift auch dann auf die Konvergenzgeschwindigkeit auswirkt, auch wenn es immer noch gegen denselben Wert konvergiert. Spezifisch könnte man noch sagen, dass sich der Shift von 2 nicht lohnt, da er die Konvergenzrate nur in die Höhe getrieben hat.

(c) Der beförderte Kollege Ü. kommt vorbei und macht einen Vorschlag: Um auf den Eigenvektor v_0 möglichst schnell iterativ zu kommen, schlägt er einen Shift von $\mu = 4$ vor. Argumentiere, ob das sinnvoll ist. Wenn nicht, mache einen besseren Vorschlag.

Es ist ein extrem guter Vorschlag. Nicht nur konvergiert das Verfahren gegen v_0, sondern auch noch sehr schnell. Die Konvergenzrate mit diesem Shift beträgt:

$$q_3 = \frac{0}{3} = 0$$

5 . Aufgabe: Differentialgleichungen [4 + 3 Pkte]

Gegeben sei

$$y'(t) = t^2 \cdot y(t)$$

Weiterhin gilt

$$y_0 = y(0) = 1; t \geq -1$$

(a) Stelle ein Koordinatensystem auf, mit dessen Hilfe du das obige AWP grafisch lösen würdest. Zeichne in die Grafik einen Schritt mit Heun und mit explizitem Euler Verfahren ein. Die Anfangswerte seien jeweils $y_0 = 1$ und $t_0 = 0$, die Schrittweite $\delta t = 2$.

Stichwort Richtungsfeld. Euler Schritt ist bei $(2, 1)$, während der Heun Schritt bei $(2, 5)$ endet.

(b) Durch einen instabilen Algorithmus des Kollegen Ü. sei immer eine fehlerhafte Eingabe garantiert:

$$y_\epsilon(0) = y_0 + \epsilon$$

Bestimmen sie mithilfe von Separation der Variablen die fehlerhafte Funktion $y_\epsilon(t)$

Schauen sie die zwei Fälle an:

$$\lim_{t \to \infty} |y(t) - y_\epsilon(t)|$$

und

$$\lim_{t \to 0} |y(t) - y_\epsilon(t)|$$

Aus den zwei Fällen jeweils schließend, gut oder schlecht konditioniert?

$$y_\epsilon(t) = (y_0 + \epsilon) \cdot e^{\frac{t^3}{3}}$$

Gegen null gut konditioniert, gegen unendlich schlecht konditioniert.

6 . Aufgabe: Programmieraufgabe [5.5 + 1.5 Pkte]

Wir möchten das Jacobi-Verfahren zum Lösen von linearen Gleichungssystemen implementieren.
Ein Schritt des Jacobi Verfahrens:

$$x^{(k+1)} = x^{(k)} + (diag(A))^{-1}(b - Ax^{(k)})$$

Folgende Methoden seien gegeben:

```
float[] matVec(float[][] mat, float[] vec) {
   //returns result of matrix-vector multiplication of mat and vec
}

float[] vecAdd(float[] a, float[] b) {
   //returns result of vector addition
}

float[] vecSub(float[] a, float[] b) {
   //returns a - b via vector subtraction
}

float[] scalVec(float scalar, float[] vec) {
   // returns scalar * vec as correct operation
}
```

(a) Vervollständige die folgende Methode, welche einen Jacobi Schritt ausrechnet und das Ergebnis als ein float Array zurückgeben soll.

Sonderfälle oder die Korrektheit der Dimensionen der Variablen müssen nicht überprüft werden!

```
// A is the A matrix, first index stands for the column index
// b is the result vector from the formula Ax = b
// oldX is the x vector of the previous step

float[] jacobiStep(float[][] A, float[] b, float[] oldX) {

   //Calculate current residuum:
   float[] r = vecSub(b, matVec(A, oldX));

   float[] y = new float[oldX.length()];

   //Loop over dimension:
   for (int i = 0; i < oldX.length() - 1; i++) {

        y[i] = (1 / A[i][i]) * r[i];

   }

   float[] newX = vecAdd(y, oldX);

   return newX;

}
```

(b) Dein beförderter Kollege Dr. Ü. verlangt, dass das ”in-place” Prinzip angewendet wird. Erläutere kurz, wie du den Programmcode abändern müsstest, um dies umzusetzen.

Die r Berechnung auch noch in die For-Schleife packen und zeilenweise berechnen. Außerdem noch in der Schleife die Einträge des x-Vektors überschreiben anstelle danach komplett zu berechnen. Unnötige Zeilen wie newX und die y Defintion sind nicht mehr notwendig und können entfernt werden.