Probeklausur Numerisches Programmieren

Sommersemester 16/17, MUSTERLÖSUNG

Hendrik Möller

31.07.2017

Der Autor übernimmt keine Garantie über die Korrektheit der Aufgaben und 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 21 Punkte benötigt.
- Hinweis: Nach Aufassung des Autors ist diese Klausur schwerer als die ”richtige” Klausur sein wird.
- Die einzelnen Teilaufgaben sind bis auf wenige Ausnahmen 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 [4.5 + 2.5 Pkte]

In dieser Aufgabe gehen wir von einer Darstellung durch Gleitkommazahlen 8-Bit-$G_{B,t}$ aus. $G_{B,t}$ entspricht der IEEE Form mit folgenden Außnahmen:

- Der Exponent E hat nur 3 Bits,
- der Exponent besitzt nicht wie bei IEEE einen Shift, sodass es negative und positive Exponenten gibt sondern entspricht einer normalen binären Zahl (also sind hier Exponenten von 0 bis 7 möglich).
- Für die Mantisse M werden nur 4 Bits verwendet,
- das Vorzeichen bit existiert weiterhin!

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

1. 64_{10}
2. $1,375_{10}$
3. $-0xE_{16}$

Hinweis: Die IEEE Form bringt noch mehr Regeln mit außer dem Vorzeichen Bit!

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

1. $64_{10} = 1000000_2 = 1,0 \cdot 2^6 \Longrightarrow 0|110|0000$
2. $1,375_{10} = 1,011_2 = 1,011 \cdot 2^0 \Longrightarrow 0|000|0110$
3. $-0xE_{16} = -14_{10} = 1110_2 = 1,110 \cdot 2^3 \Longrightarrow 1|011|1100$

(b) Erkläre das Problem der "Absorption" und gib ein Beispiel in der 8-Bit-$G_{B,t}$ Darstellung an, bei dem das Problem auftreten würde.

"Absorption" tritt dann auf, wenn man zwei unterschiedlich große Gleitkommazahlen miteinander addiert und dadurch Informationen verloren gehen (also gerundet werden muss). Hier gibt es mit obiger Darstellung viele richtige Beispiele.
z.B:

$$0|111|1000 + 0|000|0001$$

2 . Aufgabe: Differentialgleichungen [3 + 2.5 + 2.5 Pkte]

$$y'(x) = -\frac{1}{2} \cdot x \cdot y(x)^2$$

Weiterhin gilt

$$y(-1) = 1; x \geq -1$$

(a) Stelle ein Koordinatensystem auf, mit dessen Hilfe du das obige AWP graphisch lösen würdest und zeichne eine Vermutung ein, wie die Funktion sein könnte.

Stichpunkt Richtungsfelder. Es ergibt sich in etwa eine umgekehrte Parabel die nach rechts immer flacher wird. Wichtig ist, dass nichts links von -1 ist, da mit $x \geq -1$ das AWP dort nicht mehr gilt! Außerdem muss der Wert $y(-1) = 1$ korrekt eingezeichnet sein!

(b) Berechne nun die exakte Funktion mithilfe von Separation der Variablen.

$$\frac{dy}{dx} = -\frac{1}{2}x \cdot y^2 \iff \frac{dy}{y^2} = -\frac{1}{2}x \cdot dx$$
$$\int_1^y \frac{1}{\rho^2} d\rho = \int_{-1}^x \frac{1}{2}\tau d\tau$$
$$[-\rho^{-1}]_1^y = [\frac{1}{4}\tau^2]_{-1}^x$$
$$-\frac{1}{y} + 1 = \frac{1}{4}x^2 - \frac{1}{4}$$
$$\implies y(x) = \frac{1}{\frac{1}{4}x^2 + \frac{3}{4}}$$

(c) Berechne zwei Schritte des impliziten Euler Verfahren, um das Ergebnis des AWPs an der Stelle $x = 1$ anzunähern.

$$y_{k+1} = y_k + \delta t \cdot f(t_{k+1}, y_{k+1})$$

Mit $x_0 = -1$, $x_1 = 0$ und $y_0 = 1$

$$y_1 = y_0 + 1 \cdot (-\frac{1}{2} \cdot 0 \cdot y_1^2)$$
$$y_1 = 1$$
$$y_2 = y_1 + 1 \cdot (-\frac{1}{2} \cdot x_2 \cdot y_2^2)$$
$$y_2 = 1 + 1 \cdot (-\frac{1}{2} \cdot 1 \cdot y_2^2)$$
$$y_2 = -1 + \sqrt{3}$$

3 . Aufgabe: Interpolation [2.5 + 3 + 3.5 Pkte]

Es seien folgende Stützpunkte gegeben:

$$p_0 = (-1|3); p_1 = (0|2); p_2 = (1|1)$$

(a) Geben sie das Ergebnis der Interpolationsfunktion an der Stelle $x = 3$ aus, indem sie die regulären Polynom Basisfunktionen verwenden.

$$\left[\begin{array}{ccc|c} 1 & -1 & 1 & 3 \\ 1 & 0 & 0 & 2 \\ 1 & 1 & 1 & 1 \end{array}\right] \Longrightarrow \left[\begin{array}{ccc|c} 1 & -1 & 1 & 3 \\ 0 & 1 & -1 & -1 \\ 0 & 0 & 2 & 0 \end{array}\right]$$

$$c = (2, -1, 0)^T \Longrightarrow p(x) = 2 - x$$

$$p(3) = -1$$

(b) Wenn man nun an der Interpolationsfunktion interessiert ist und einfach neue Stützpunkte hinzufügen können möchte, welches Verfahren würde man nutzen?
Rechne mit diesem Verfahren und den obigen Stützpunkten die Interpolationsfunktion aus!

Newton Verfahren!
Hier ist das Tabellenschema als Matrix aufgeschrieben (Stellt es euch als Tabelle vor)

$$\left[\begin{array}{c|ccc} -1 & 3 & -1 & 0 \\ 0 & 2 & -1 & \\ 1 & 1 & & \end{array}\right]$$

Aus der ersten Zeile erfolgen die Koeffizienten und nach Formel aus dem Skript folgt:

$$p(x) = 3 - (x + 1) + 0 = 2 - x$$

(c) Nun soll ein Verfahren gewählt werden, welches effektiv mit unterschiedlichen Stützwerten, aber gleichbleibenden Stützstellen umgehen kann.
Stelle für die folgenden Stützstellen das Verfahren auf und rechne das Interpolationspolynom so gut wie möglich aus:

$$p_0 = (-1|y_0); p_1 = (0|y_1); p_2 = (1|y_2)$$

Lagrange Basispolynome eignen sich, da man die komplett nur mit den Stützstellen ausrechnen kann!
Also im Grunde alle 3 Lagrange Basisfunktionen mit den gegebenen Werten ausrechnen und dann mit den nicht gegebenen y Werte multiplizieren ergibt das Interpolationspolynom, bei dem man nun nicht mehr weiter rechnen kann.

4 . Aufgabe: Quadratur [1 + 3 + 3.5 + 1.5 Pkte]

(a) Berechne mithilfe der Trapezregel das Integral über eine Funktion f_0, die durch folgende Punkte geht:

$$P_0(0|3); P_1(2|1)$$

$$Q_T(f) = \frac{1}{2}H \cdot (f(a) + f(b))$$

$$Q_T(f_0) = \frac{1}{2} \cdot 2 \cdot (3 + 1) = 4$$

(b) Berechne das Integral über die beiden voherigen Punkten und den folgenden zusätzlichen mit klasisschen Quadraturregeln bestmöglich:

$$P_2(1|2); P_3(3|1)$$

Hier ist die Trapezsumme gefragt (das ist zumindest im Moment bestmöglich):

$$Q_{TS}(f; 1) = 1 \cdot (\frac{3}{2} + 2 + 1 + \frac{1}{2}) = 5$$

(c) Ein weiterer Punkt über die Funktion f_0 ist bekannt, nämlich $P_4(5|8)$. Berechne mit schon obig verwendeten Methoden wieder das Integral über den gesamten Bereich bestmöglich.

Hier kann man weder Trapezsumme noch Simpsonsumme über den ganzen Bereich anwenden, weil die Schrittweite zwischen den Punkten nicht konstant wäre. Man kann sich damit behelfen, dass man die ersten vier Punkte über die Trapezsumme berechnet und den letzten Abschnitt mit einer Trapezregel:

$$5 + (\frac{1}{2} \cdot 2 \cdot (1 + 8)) = 5 + 9 = 14$$

(d) Gib die Lösung folgender Rechnung mit Quadratur nach Romberg an.
(Rechenweg ist diesmal nicht relevant!)

$$\int_0^{2\pi} sin(x)^3 dx$$

Rechnet hier nichts aus! Der Rechenweg ist nicht verlangt, sondern nur die Lösung. Bei genauem Hinschauen erkennt man, dass das Integral 0 sein muss! (Jeweils die Fläche von 0 bis π und π bis 2π neutralisieren sich!)

5 . Aufgabe: Iterative Verfahren [3 + 3 Pkte]

Wurde gelöscht weil meeeh.

6 . Aufgabe: Programmieraufgabe [4.5 + 1.5 Pkte]

Wir möchten eine iteratives Verfahren zur Nullstellenbestimmung mithilfe von Bisektion umsetzen.
Es sei schon folgende Methode definiert:

```
float f(float x) {
   // returns f(x) as float value
}
```

Vervollständige die folgende Methode, welche einen Bisektionsschritt ausrechnen und das Ergebnis auf eine sinnvolle Art und Weise zurückgeben soll.

1.5 der 6 Punkte werden für das Behandeln von Sonderfällen gegeben. Generell soll gelten, dass wenn der Bisektionsschritt nicht (mehr) ausgerechnet werden kann, die alten Variablen unverändert zurückgegeben werden.

```
// xLeft and xRight are the current reference points left/right from the zero
    point

float[] bisecStep(float xLeft, float xRight) {

   // Neuen Funktionswerte holen:
   float xLeftFunc = f(xLeft);
   float xRightFunc = f(xRight);

   // Fehler bzw. Fertig Erkennung:
   if ((xLeftFunc > 0 && xRightFunc > 0) ||
      (xLeftFunc < 0 && xRightFunc < 0) ||
      (xLeftFunc == 0 || xRightFunc == 0)) {
      return {xLeft, xRight};
   } else {

      // Neuen Mittelpunkt errechnen:
      float xMiddle = (xLeft + xRight) / 2;
      float xMiddleFunc = f(xMiddle);

      if (xMiddleFunc >= 0 && xLeftFunc > 0) {
         //xLeft = xMiddle;
         return {xMiddle, xRight};
      } else {
         //xRight = xMiddle;
         return {xLeft, xMiddle};
      }
   }
}
```