Lehrstuhl für Wissenschaftliches Rechnen
Fakultät für Informatik
Technische Universität München

Esolution
Sticker mit SRID hier einkleben

Hinweise zur Personalisierung:

- Ihre Prüfung wird bei der Anwesenheitskontrolle durch Aufkleben eines Codes personalisiert.
- Dieser enthält lediglich eine fortlaufende Nummer, welche auch auf der Anwesenheitsliste neben dem Unterschriftenfeld vermerkt ist.
- Diese wird als Pseudonym verwendet, um eine eindeutige Zuordnung Ihrer Prüfung zu ermöglichen.

Numerisches Programmieren

Klausur: IN0019 / Testexam
Prüfer: Prof. Dr. Hans-Joachim Bungartz
Datum: Dienstag, 1. Februar 2022
Uhrzeit: 08:00 – 23:45

Bearbeitungshinweise

- Diese Klausur umfasst **14 Seiten** mit insgesamt **6 Aufgaben**.
  Bitte kontrollieren Sie jetzt, dass Sie eine vollständige Angabe erhalten haben.
- Die Gesamtpunktzahl in dieser Klausur beträgt 50 Punkte.
- Das Heraustrennen von Seiten aus der Prüfung ist untersagt.
- Als Hilfsmittel sind zugelassen:
  - alle Hilfsmittel, insbesondere Bücher, persönliche Notizen, Internetsuchmaschinen, selbst erstellte Skripte und Programmcodes.
  - nicht erlaubt sind Hilfestellungen von Dritten oder Kommunikation mit Dritten.
  - nicht erlaubt sind Plagiate jeder Art.
- Mit * gekennzeichnete Teilaufgaben sind ohne Kenntnis der Ergebnisse vorheriger Teilaufgaben lösbar.
- **Es werden nur solche Ergebnisse gewertet, bei denen der Lösungsweg erkennbar ist.** Auch Textaufgaben sind **grundsätzlich zu begründen**, sofern es in der jeweiligen Teilaufgabe nicht ausdrücklich anders vermerkt ist.
- Schreiben Sie weder mit roter / grüner Farbe noch mit Bleistift.
- Beachten Sie die TUMExam Empfehlungen zur Abgabe von PDFs.

Aufgabe 1 Gleitkommazahlen und Kondition (10 Punkte)

0 1 2 3

a) Wandeln Sie die Zahl $x = -\frac{23}{7} = 11.\overline{010}_2$ in das binäre IEEE Float Format um. Geben Sie das Ergebnis als 32-Bit Bitfolge an und markieren Sie Vorzeichen, Exponent und Mantisse. Der Rechenweg muss erkenntlich sein!
Hinweis: Das IEEE Float Format verwendet 1 Bit für das Vorzeichen, 8 Bit für den Exponenten und 23 Bit für die Mantisse. Runden Sie wie in der Übung besprochen. Markieren Sie eine mögliche Periodizität in der Berechnung mittels Klammern (z.B.: $11.\overline{010}_2 = 11.(010)_2$). Schreiben sie bei den Bitfeldern die ganze Zahl aus und kürzen Sie diese nicht ab!

$x = -\frac{23}{7} = 11.\overline{010} = 1.1\overline{010} \cdot 2^1$
Bitfeld:
1 10000000 10100100100100100100101 (aufgerundet)
VZ EXP MANTISSE

0 1 2

b) **Kondition einer Matrix**
Bestimmen Sie die Konditionszahl, das lineare Gleichungssystem

$$Ax = b$$

zu lösen, für die Matrix A:

$$A = \begin{pmatrix} 100 & 0 \\ 0 & 0.1 \end{pmatrix}.$$

cond(A) = 100/0.1 = 1000

0 1

c) Angenommen Sie möchten ein Problem mit bekannter relativer Konditionszahl $K = 42$ lösen. Sie wissen, dass ihr Messgerät für den Inputwert eine relative Messungenauigkeit von 10^{-2} hat. Mit welchem maximalen Fehler E müssen Sie bei der Ausgabe rechnen, wenn Sie annehmen, dass Sie fehlerfrei rechnen?

Ausgabefehler $= 42 * 10^{-2} = 0.42$

d) Gegeben ist die Funktion $f(x) = 1 - x^2$.

0
1
2
3
4

i) Berechnen Sie die Konditionszahl von $f(x)$ in Abhängigkeit von x.

$cond(f, x) = \left|\frac{x2x}{1-x^2}\right|$

ii) Für welche Werte ist die Funktion schlecht konditioniert?

$x = \pm 1$

iii) Welcher Effekt ist für die schlechte Kondition verantwortlich?

Auslöschung (oder auch möglich: Nullstelle der Funktion; da relative Kondition)

Aufgabe 2 **Fixpunktiteration (4 Punkte)**

Gegeben sind die Funktionen $f(x) = x^2 - 4$ und $g(x) = x^2 - 4x + 4$.

0 1 2

a) Berechnen Sie die Nullstellen von $f(x)$ und $g(x)$.

$f(x) : \pm 2$
$g(x) : 2$ (doppelt)

0 1

b) Formulieren Sie das Newtonverfahren für $f(x)$ indem Sie die entsprechende Iterationsvorschrift $\Phi_1(x)$ angeben.

$\Phi_1(x) = x - \frac{x^2-4}{2x}$

0 1

c) Formulieren Sie das Newtonverfahren für $g(x)$ indem Sie die entsprechende Iterationsvorschrift $\Phi_2(x)$ angeben.

$\Phi_2(x) = x - \frac{x^2-4x+4}{2x-4}$

Aufgabe 3 Interpolation (7 Punkte)

In dieser Aufgabe soll eine Polynominterpolation mithilfe des Aitken-Neville Verfahrens durchgeführt werden.

a) Wir verwenden das aus der Übung bekannte Dreiecksscheme zur Veranschaulichung des Aitken-Neville Algorithmus:

0 1 2 3

x_i	$i \setminus k$	0		1		2		$\cdots$
x_0	0	$p[0,0]$	$\rightarrow$	$p[0,1]$	$\rightarrow$	$p[0,2]$	$\rightarrow$	$\cdots$
			$\nearrow$		$\nearrow$			
x_1	1	$p[1,0]$	$\rightarrow$	$p[1,1]$	$\rightarrow$	$\vdots$		
			$\nearrow$					
x_2	2	$p[2,0]$	$\rightarrow$	$\vdots$				
$\vdots$	$\vdots$	$\vdots$						

Gegeben sind die Punkte $P_0 = (0,0)$, $P_1 = (1,1)$ und $P_2 = (2,4)$ und der Auswertungspunkt $x = -1$. Berechnen Sie die Einträge des Dreiecksschemas für diesen Fall. Geben Sie die vollständigen Rechenweg an und spezifizieren Sie den berechneten Eintrag (Beispiel: p[0][2] = <Berechnung>). Geben Sie außerdem das Ergebnis der Auswertung an der Stelle -1 an (also $p(-1)$ für das Interpolationspolynom p).

P[0][0] = 0
P[1][0] = 1
P[2][0] = 4
P[0][1] = 0 + (-1 - 0)/(1-0) * (1-0) = -1
P[1][1] = 1 + (-1 - 1)/(2-1) * (4-1) = -5
P[0][2] = -1 + (-1 - 0)/(2-0) * (-5 +1) = 1
p(-1) = 1

b) Wie können die Werte in der vorherigen Tabelle für $k \geq 1$ grafisch ermittelt werden? Beschreiben Sie die grafische Berechnung.

0 1 2

k = 1: Auswerten der Geraden zwischen P_1 und P_2 an -1-> p[0][1]
Auswerten der Gerade zwischen P_2 und P_3 an -1 -> p[1][1]
k = 2: Auswerten der Parabel zwischen allen Punkten an -1 -> p[0][2]

0 1 2

c) Einer ihrer Mitstudenten verwendet den Aitken-Neville Algorithmus, um ein Interpolationsproblem zu lösen. Da er mit der Genauigkeit unzufrieden ist, erhöht er die Anzahl der Stützpunkte immer weiter. Hierbei beobachtet er aber eine Verschlechterung der Ergebnisse bei mehr als 10 Stützpunkten. Verzweifelt sucht er nach einem Bug in seinem Code und zieht Sie zu rate. Was ist das Problem und wie sieht eine mögliche Lösung hierfür aus?

Problem: Runge-Effekt (Oszillationen entstehen durch hohen Grad) Lösung: Chebyshev Punkte, Stückweise Interpolation, Splines, ...

Aufgabe 4 Quadratur (8 Punkte)

a) In dieser Teilaufgabe betrachten wir die Funktion

0 1 2

$$f(x) = x^5 + 4 \cdot x^2 + 1, \tag{4.1}$$

welche wir im Intervall $[a, b] = [-1, 1]$ numerisch integrieren wollen, d.h., wir suchen eine Approximation des Integrals $I(f) = \int_{-1}^{1} f(x)dx$.
Berechnen Sie die Trapezsumme $Q_{TS}(f, h)$ für $N = 2$ (d.h. mit 2 Trapezen).

$$h = (b - a)/N = 1$$

$$Q_{TS}(f, 1) = 1 \cdot (\frac{f(-1)}{2} + f(0) + \frac{f(1)}{2})$$

$$= 2 + 1 + 3 = 6$$

b) Sie testen für eine andere Funktion $g(x)$ mehrere Quadraturverfahren und bemerken, dass Ihre Approximation noch nicht gut genug ist. Deshalb wollen Sie die Anzahl der Stützstellen für das jeweilige Quadaturverfahren "verdoppeln"(Anzahl der Stützstellen $n + 1$ wird auf $2n + 1$ erhöht; n ist gerade). Welchen Nachteil hat hier die Gauß-Quadratur gegenüber der Trapezsumme? Nehmen Sie an, dass die Berechnung der Stützstellen (x_i) und der Gewichte (w_i) keinen Aufwand erfordert. Stattdessen werden die Berechnungskosten durch das Auswerten der Funktion $g(x)$ dominiert.

0 1 2

Bei der Gaußquadratur unterscheiden sich alle Stützpunkte von den alten. Deshalb brauchen wir doppelt soviele Funktionsauswertungen im Vergleich zur Trapezsumme, da wir dort die alten Punkte wiederverwenden können.

c) Welchen Polynomgrad kann man mit einer Gaußquadratur mit N Punkten richtig integrieren (keine Begründung nötig)?

0 1

2N - 1

0
1

d) Wodurch unterscheidet sich die Herleitung einer Gaußquadraturformel mit N Punkten von anderen Quadraturformeln (z.B. Trapez- oder Simpsonregel)? Wie entsteht hierbei der höhere Grad?

Punkte werden zusätzlich zu den Gewichten optimiert. Dadurch entstehen doppelt soviele Gleichungen ($2N$). Hieraus folgt der Grad $2N - 1$

0
1

e) Welche Eigenschaft sollten die Gewichte einer Quadraturformel aufweisen, damit numerische Probleme vermieden werden?

Alle Gewichte sollten ≥ 0 sein (wobei > 0 zu bevorzugen ist).

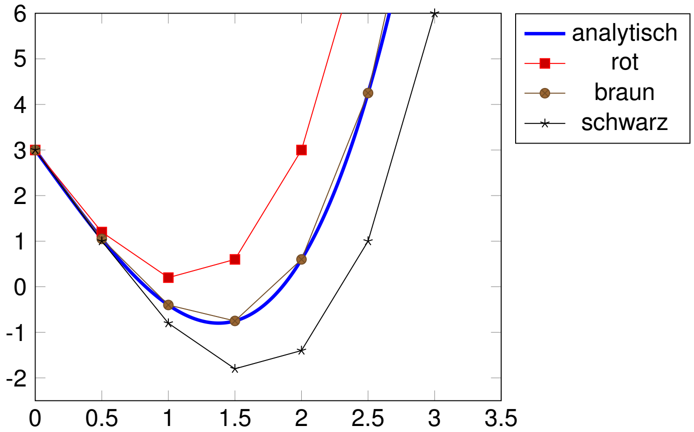
f) Wie oft sind kubische Splines im Allgemeinen maximimal stetig ableitbar (k mal stetig ableitbar heißt hier formal, dass das globale Polynom $\in C^k$ sein muss)?

- [x] 2 mal
- [ ] 3 mal
- [ ] unendlich oft
- [ ] 0 mal
- [ ] 1 mal

Aufgabe 5 **Gewöhnliche Differentialgleichungen (11 Punkte)**

Diese Aufgabe behandelt verschiedene Aspekte zu gewöhnlichen Differentialgleichungen.

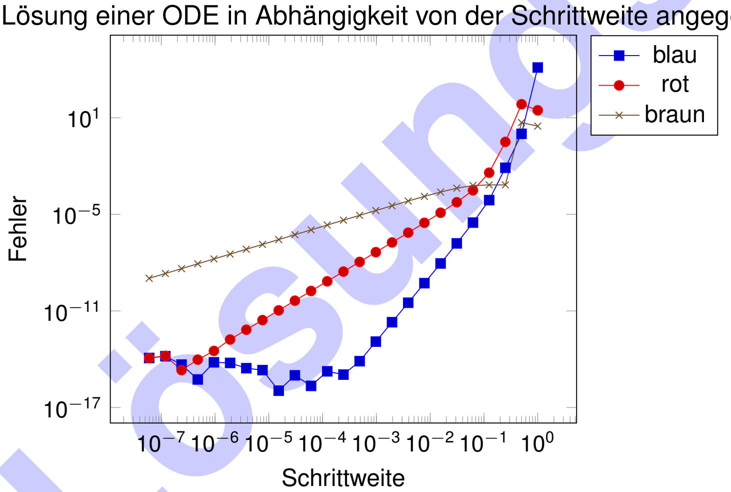
a) In der Grafik sind neben der analytischen Lösung einer Differentialgleichung, numerische Lösungen mit dem expliziten und dem impliziten Euler-Verfahren, sowie dem Runge-Kutta-Verfahren (RK4) zu sehen.



Ordnen Sie die Lösungen dem Verfahren zu, so dass erst das explizite Eulerverfahren, dann das RK4-Verfahren und dann das implizite Euler-Verfahren angeben ist (explizit, RK4, implizit).

- [ ] rot, schwarz, braun
- [ ] braun, schwarz, rot
- [x] schwarz, braun, rot
- [ ] braun, rot, schwarz
- [ ] schwarz, rot, braun
- [ ] rot, braun, schwarz

b) In der Grafik ist der globale Fehler verschiedener numerischer Verfahren (Euler, Heun, Runge-Kutta-4) zur Lösung einer ODE in Abhängigkeit von der Schrittweite angegeben.



Ordnen Sie die Graphen dem Verfahren zu, so dass erst das Eulerverfahren, dann das Heun Verfahren und zuletzt das Runge-Kutta-Verfahren angegeben ist (Euler, Heun, RK4).

- [ ] rot, braun, blau
- [ ] blau, rot, braun
- [ ] rot, blau, braun
- [ ] blau, braun, rot
- [ ] braun, blau, rot
- [x] braun, rot, blau

c) Erläutern Sie, warum sich der Fehler für das blaue Verfahren aus Teilaufgabe b) für kleine Schrittweiten nicht mehr ändert und für sehr kleine Schrittweiten wieder schlechter wird.

Ersteres geschieht auf Grund von numerischen Rechenfehlern, da wir uns im Bereich der Maschinengenauigkeit bewegen. Für sehr kleine Schrittweiten kommt es zur Absorption, da die hinzugefügten Werte sehr klein sind.

d) Gegeben ist die folgende Differentialgleichung:

$$\frac{\mathrm{d}}{\mathrm{d}x} y(x) = -4y(x)$$

Stellen Sie für diese Differentialgleichung das Heun-Verfahren auf und berechnen Sie die maximale Schrittweite h, für welche die Iterationsregel gegen den Fixpunkt $y = 0$ konvergiert.

$$y_{n+1} = y_n + \frac{h}{2} \cdot (-4y_n + (-4)(y_n + h(-4y_n)))$$
$$y_{n+1} = y_n - 4h \cdot y_n + 8h^2 y_n$$

$$y_{n+1} = (1 - 4h + 8h^2) y_n$$

stabil wenn:

$$abs(1 - 4h + 8h^2) < 1$$

alternativ:

$$0 < 1 - 4f + 8h^2 < 1$$

Fall 1: +

$$1 - 4h + 8h^2 = 1$$
$$-4h + 8h^2 = 0$$
$$h = 0 \text{ or } h = \frac{1}{2}$$

Fall 2: -

$$1 - 4h + 8h^2 = -1$$
$$h = \frac{4 \pm \sqrt{16 - 4 \cdot 8 \cdot 2}}{2 \cdot 8}$$

Fall 2 hat keine (reelwertige) Lösung (äquivalent für 0)

Aufgabe 6 Programmieraufgabe Lineare Gleichungssysteme + ODE (10 Punkte)

a) Entscheiden Sie sich für ein **iteratives** Verfahren um die Lösung des Gleichungssystems $Mx = b$ zu bestimmen, **nennen Sie dieses** (als Kommentar) und ergänzen Sie den unten stehenden Quelltext. Achten Sie dabei auf eine geeignete Abbruchbedigung der Iteration. Eine Rekursion soll nicht benutzt werden.

```java
public class LinearEquationSolver {

    private final double[][] M;

    public LinearEquationSolver(double[][] M) {
        this.M = M;
    }

    /**
     * Solves Mx = b. Uses starting value x0 for an iterative solution.
     *
     * @param b  Vector b.
     * @param x0 Starting value for iterative solution.
     * @return The solution x of the equation Mx=b.
     */
    public double[] solve(final double[] b, final double[] x0) {
        int length = x0.length;
        double[] x = x0.clone();
```

Jacobi oder Gauss-Seidel. Hier angegeben ist GS.

```java
        double error = 1e10;
        final double maxError = 1.e−10;
        while (error > maxError) {
            error = 0.;
            for (int i = 0; i < length; i++) {
                double sigma = 0.;
                for (int j = 0; j < length; j++) {
                    if (i != j) {
                        sigma += M[i][j] * x[j];
                    }
                }
                double tmp = (b[i] − sigma) / M[i][i];
                error += (tmp − x[i]) * (tmp − x[i]);
                x[i] = tmp;
            }
        }
```

```java
        return x;
    }
}
```

b)

Diese Aufgabe behandelt die numerische Lösung des Anfangswertproblems

$$\frac{\mathrm{d}}{\mathrm{d}t} y = A\,y$$
$$y, y_0 \in \mathbb{R}^n$$
$$A \in \mathbb{R}^{n\times n}.$$

0 1 2 3 4

Für die Lösung des Anfangswertproblems(AWP) ist das untenstehende Codegerüst gegeben. Dieses nutzt das **implizite Euler**-Verfahren um das AWP in ein lineares Gleichungssystem der Form **Mx** = **b** zu verwandeln, welches dann durch den linearen Gleichungssystemlöser aus der Teilaufgabe a) gelöst wird. Starten Sie mit der Iterationsvorschrift und stellen Sie die zu lösende Matrixgleichung auf. **Geben Sie** M**,** x **und** b **an. Ergänzen Sie dann den Quelltext**, so dass M entsprechend initialisiert wird.

$$y_{n+1} = y_n + dt \cdot A y_{n+1}$$
$$(1 - dt \cdot A) y_{n+1} = y_n$$
$$M = 1 - dt \cdot A, \quad x = y_{n+1}, \quad b = y_n$$

```java
public class LinearODESolver {
    private LinearEquationSolver eqSolver;

    /**
     * @param A Quadratic matrix.
     * @param dt This is the time step.
     */
    public LinearODESolver(double[][] A, double dt) {
        final int length = A.length;
        double[][] M = new double[length][length];
        // TODO initialize M
```

```java
        for (int i = 0; i < length; i++) {
            for (int j = 0; j < length; j++) {
                M[i][j] = (i == j ? 1. : 0) - dt * A[i][j];
            }
        }
```

```java
        this.eqSolver = new LinearEquationSolver(M);
    }

    /**
     * Advance y.
     * @param y current position
     * @return next position
     */
    public double[] nextStep(double[] y) {
        // solves Mx=b with starting value x0.
        return eqSolver.solve(/*b*/ y, /*x0*/ y);
    }
}
```

Zusätzlicher Platz für Lösungen. Markieren Sie deutlich die Zuordnung zur jeweiligen Teilaufgabe. Vergessen Sie nicht, ungültige Lösungen zu streichen.