Name, Vorname, Matrikelnummer:

Allgemeine Hinweise zur Klausur

- **Bearbeitungszeit**: 90 Minuten.
- **Hilfsmittel**:
  - **Ein** DIN-A4 Blatt, beidseitig **handschriftlich** beschrieben (keine Kopie!).
  - ggf. Fremdsprachenwörterbuch (Deutsch-. . . /. . . -Deutsch) ohne handschriftliche Eintragungen!
- **Allgemeine Hinweise**:
  - Trennen Sie unter keinen Umständen die Heftung der Blätter auf!
  - Falls der Platz zur Bearbeitung einer Teilaufgabe nicht ausreicht, markieren Sie dies bitte und benutzen die Rückseite des vorhergehenden Blattes.
  - Beschriften Sie den Kopf **dieses** Blattes mit Ihrem vollständigen Namen und Ihrer Matrikelnummer.
  - Beschriften Sie den Kopf **jedes anderen** Blattes mit Ihrem Kürzel (Initialen) und Ihrer Matrikelnummer für den Fall, dass sich die Heftung löst.
  - Die **letzte** Seite stellt ein leeres Zusatzblatt dar.
- **Punkteverteilung**:
  - Für die 5 Klausuraufgaben werden insgesamt **43** Punkte vergeben.
  - Die ungefähre Punkteverteilung innerhalb einer Aufgabe ist als Orientierungshilfe jeweils zu Beginn der Aufgabe angegeben. Es handelt sich **nicht** unbedingt um die endgültige Punkteverteilung.

Viel Erfolg!

1	2	3	4	5	$\sum$
/10	/9	/7	/7	/10	/43

1 Gleitkommazahlen und Kondition (4 + 2 + 2 + 2 = 10 Pkt.)

a) Wandeln Sie die Zahl $x = \frac{3}{7}$ 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.

$x = \frac{1}{15} = 0.\overline{011} = 1.\overline{101} \cdot 2^{-2}$

Bitfeld:

0 01111101 10110110110110110110111 (aufgerundet)
VZ EXP MANTISSE

b) In der Praxis verwenden viele Programmiersprachen (z.B. Python) denormalisierte Gleitkommazahlen, um den Zahlenbereich von float oder double zu vergrößern. Hierfür wird für den kleinstmöglichen Exponent $e = exp_{min}$ die Restriktion aufgehoben, dass vor den Mantissenbits eine 1 steht. Die Mantisse speichert in diesem Fall sowohl die führende Ziffer als auch die Nachkommastellen (jetzt natürlich mit einer Nachkommastelle weniger).

- Was ist nun die kleinste Zahl > 0, welche mit diesem Zahlenformat angegeben werden kann?

Hinweis: Sie können in diesem Beispiel **Sonderfälle in den Exponenten ignorieren**.

IEEE Standard $2^{exp_{min}} = 2^{-127}$
Denormalisiert $2^{exp_{min}-(m-1)} = 2^{-127-22} = 2^{-149}$

c) **Kondition**

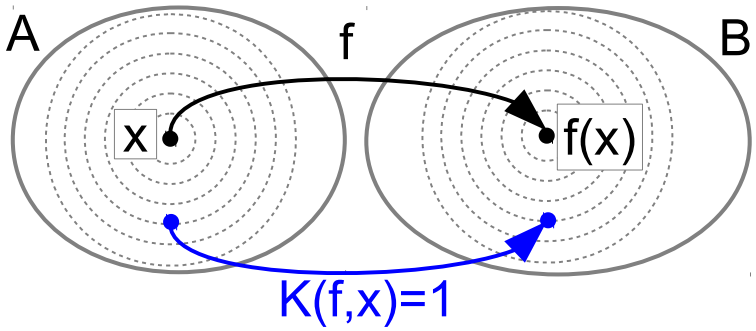

Abbildung 1: Veranschaulichung von f und $K(f, x) = 1$

Gegeben sei die Funktion $f : A \to B$. In Abb. 1 veranschaulicht der blaue Pfeil, markiert mit $K(f, x) = 1$, eine Kondition der Größe 1 von f an der Stelle x ($K(f, x) = 1$). Zeichnen Sie Pfeile in Abb. 2, die folgende Fälle veranschaulichen:

i) $K(f, x) = 0.5$

ii) $K(f, x) = 2$

Die Start- und Endpunkte der Pfeile sollen erkennbar sein!

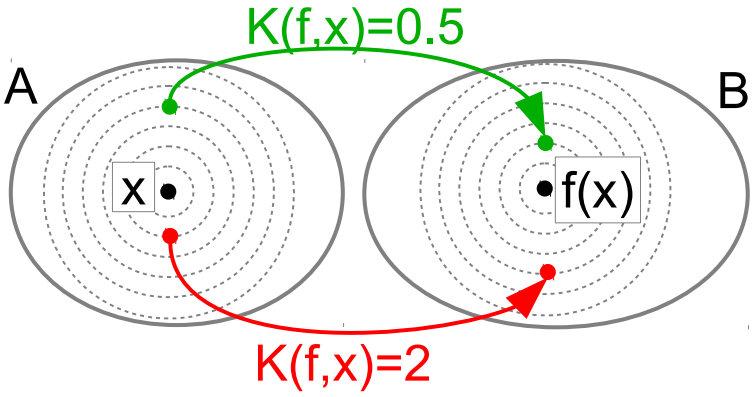

Abbildung 2: Veranschaulichung von $K(f, x) = 0.5$ und $K(f, x) = 2$

d) **Kondition einer Matrix**
Bestimmen Sie die Konditionszahl, das Lineare Gleichungssystem

$$Ax = b$$

zu lösen, für die Matrix A:

$$A = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}.$$

Falls Sie diese Aufgabe für A nicht lösen können, können Sie einen Teil der Punkte erreichen, wenn Sie die Aufgabe für die Matrix $B = 0.5 \cdot (A + A^T)$ lösen.

Dadurch, dass A weder diagonal, noch symmetrisch ist, brauchen wir die Singulärwerte von A:

$$AA^T = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix}.$$

$$\det(\sigma I - AA^T) = \det \begin{pmatrix} \sigma - 2 & -1 \\ -1 & \sigma - 1 \end{pmatrix} = (\sigma - 2)(\sigma - 1) - 1$$

$$\det(\sigma I - AA^T) = \sigma^2 - 3\sigma + 1 \Rightarrow \sigma_{1,2} = \frac{3 \pm \sqrt{5}}{2}$$

$$\kappa = \sqrt{\frac{\sigma_{max}}{\sigma_{min}}} = \sqrt{\frac{3 + \sqrt{5}}{3 - \sqrt{5}}}$$

Für Matrix B: $\kappa = \frac{1.5}{0.5} = 3$.

2 Quadratur (2 + 3 + 3 + 1 = 9 Pkt.)

a) In dieser Teilaufgabe betrachten wir die Funktion

$$f(x) = x^5 + 12 \cdot x^2 \tag{1}$$

welche wir im Intervall $[a, b] = [-1, 1]$ numerisch integrieren wollen, d.h. wir suchen eine Approximation des Integrals $I(f) = \int_{-1}^{1} f(x)dx$.

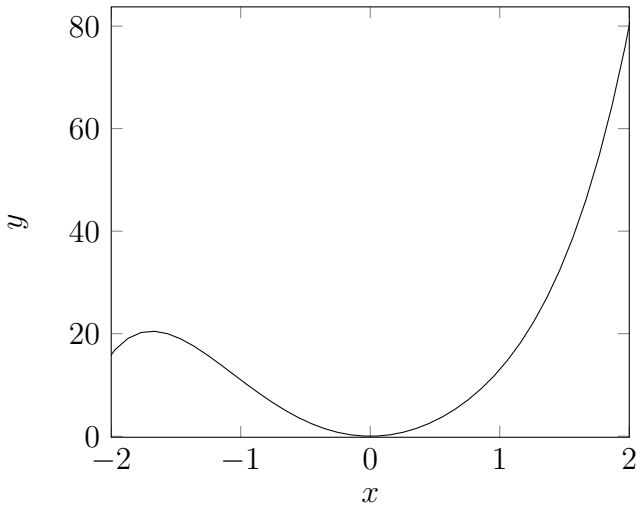

Abbildung 3: Plot der Funktion f

i) Berechnen sie I analytisch.

$$\int_{-1}^{1} x^5 + 12 \cdot x^2 dx = 8$$

ii) Berechnen Sie die Trapezsumme $Q_{TS}(f, h)$ für $N = 2$ (d.h. mit 2 Trapezen).

Markieren Sie die berechnete Fläche **auch grafisch** in Abb. 3.

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

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

$$= 5.5 + 0 + 6.5 = 12$$

iii) Berechnen Sie die Gaußquadratur $Q_G(f, 3)$ mit 3 Punkten. Aus der Literatur wissen Sie, dass hierfür gilt: $x_0 = -\sqrt{\frac{3}{5}}$, $x_1 = 0$, $x_2 = \sqrt{\frac{3}{5}}$, $w_0 = \frac{5}{9}$, $w_1 = \frac{8}{9}$, und $w_2 = \frac{5}{9}$. Was beobachten Sie bezüglich des Quadraturfehlers? Begründen Sie Ihre Beobachtung möglichst präzise.

$$Q_G(f, 3) = w_0 f(0) + w_1 f(1) + w_2 f(2) =$$

$$= \frac{5}{9}((-\sqrt{\frac{3}{5}})^5 + \frac{36}{5}) + 0 + \frac{5}{9}((\sqrt{\frac{3}{5}})^5 + \frac{36}{5}) =$$

$$= 8$$

Exakt da $n = 3$ und somit Polynom vom grad $3 * 2 - 1 = 5$ exakt integrierbar.

Die Funktion f aus Teilaufgabe a) spielt in den folgenden Teilaufgaben keine Rolle.

b) Sie bemerken für eine andere Funktion g(x), dass Ihre Approximation noch nicht gut genug ist. Deshalb wollen Sie die Anzahl der Stützstellen verdoppeln. 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.

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.

3 Gewöhnliche Differentialgleichungen (2 + 2 + 3 = 7 Pkt.)

In dieser Aufgabe soll die Differentialgleichung

$$y'(x) = f(x, y(x)) = \sqrt{1 - y^2}$$

numerisch untersucht werden. Der Anfangswert der gesuchten Funktion y bei $x_0 = 0$ ist gegeben durch

$$y_0 = y(x_0) = 0.$$

a) Berechnen Sie zunächst 2 Schritte mit dem explizitem Euler-Verfahren und Schrittweite $t = 1$. Was beobachten Sie, wenn Sie y_1 und y_2 vergleichen? Erklären Sie Ihre Beobachtung.

$$\begin{aligned} y_1 &= y_0 + 1\sqrt{1 - 0^2} = 1 \\ y_2 &= y_1 + 1\sqrt{1 - 1^2} = 1 \end{aligned}$$

gleich! Fixpunkt!

b) Berechnen Sie nun einen Schritt mit dem impliziten Euler-Verfahren und Schrittweite $t = 1$. Begründen Sie es, wenn Sie eine von zwei Lösungen auswählen.

$$\begin{aligned} y_1 &= y_0 + 1\sqrt{1 - y_1^2} = \sqrt{1 - y_1^2} \\ \Leftrightarrow y_1^2 &= 1 - y_1^2 \\ \Leftrightarrow y_1 &= \pm\tfrac{1}{\sqrt{2}} = \tfrac{1}{\sqrt{2}} \end{aligned}$$

Der Fall '-' kann verworfen werde, da $y_0 = 0$ und $y'(x) \geq 0$ wegen der Wurzel

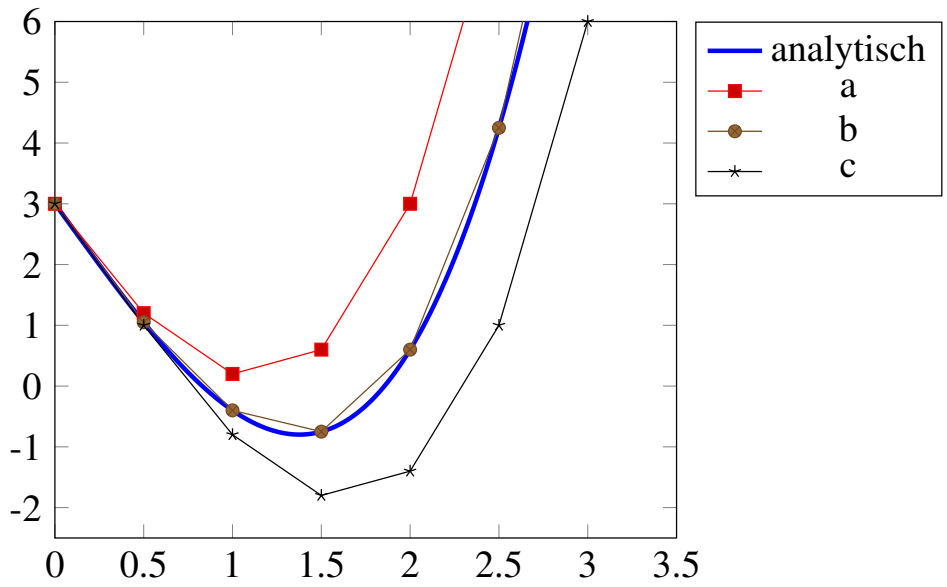

Abbildung 4: Unterschiedliche numerische Lösungen eines Anfangswertproblem.

c) Betrachten Sie den Plot in Abbildung 4. Dieser zeigt die analytische Lösung einer Differentialgleichung, sowie numerische Lösungen auf dem Intervall $[0, 3.5]$, die mit dem expliziten Eulerverfahren, dem implizitem Eulerverfahren und dem klassischen Runge-Kutta Verfahren erstellt wurden. Die Zeitschrittweite δt beträgt jeweils 0.5.

Weisen Sie jeder der drei in Abbildung 4 eingezeichneten numerischen Lösungen dasjenige Lösungsverfahren zu, mit dem sie berechnet wurden. Verbinden sie dazu unten auf dieser Seite das Verfahren mit der zugehörigen Linienfarbe. Begründen Sie auf der nächsten Seite **kurz** Ihre Entscheidung. Es ist zulässig die letzte Zuordnung durch das Ausschlussprinzip zu begründen.

Hinweis: Bei der analytischen Lösung handelt es sich um ein Polynom vom Grad 3.

explizites Euler Verfahren	☐	☐ a
implizites Euler Verfahren	☐	☐ b
klassisches Verfahren nach Runge/Kutta	☐	☐ c

Die Lösung c gehört zum expliziten Eulerverfahren, denn die Steigungen im Intervall $[x_k, x_{k+1}]$ entsprechen der Steigung der analytischen Lösung an der Stelle x_k.

Der Graph b gehört zum klassischen Runge/Kutta Verfahren. Es besitzt Ordnung 4 (muss nicht gezeigt werden), liefert also für Polynome bis zum Grad 4 exakte Ergebnisse.

Die Lösung a gehört zum implizitem Euler, denn die Steigungen im Intervall $[x_k, x_{k+1}]$ entsprechen der Steigung der analytischen Lösung an der Stelle x_{k+1}.

4 LU-Zerlegung (ca. 3 + 4 = 7 Pkt.)

a) **LU-Zerlegung**
Gegeben sei die Matrix A,

$$A = \begin{pmatrix} 1 & 4 & 1 \\ 4 & 20 & 6 \\ 1 & 6 & 258 \end{pmatrix},$$

Berechnen Sie die LU-Zerlegung von A!

Basierend auf Matrix-Multiplikation:

$$1 \cdot u_{11} = 1 \Rightarrow u_{11} = 1,$$

$$1 \cdot u_{12} = 4 \Rightarrow u_{12} = 4,$$

$$1 \cdot u_{13} = 1 \Rightarrow u_{13} = 1,$$

$$\dots$$

$$\begin{pmatrix} 1 & 0 & 0 \\ 4 & 1 & 0 \\ 1 & 0.5 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 4 & 1 \\ 0 & 4 & 2 \\ 0 & 0 & 256 \end{pmatrix} = \begin{pmatrix} 1 & 4 & 1 \\ 4 & 20 & 6 \\ 1 & 6 & 258 \end{pmatrix}.$$

b) **"Symmetrische" LU-Zerlegung**
Dadurch dass die Matrix A:

$$A = \begin{pmatrix} 1 & 4 & 1 \\ 4 & 20 & 6 \\ 1 & 6 & 258 \end{pmatrix}$$

symmetrisch und positiv-definit ist, eignet sie sich auch für eine "symmetrische" Zerlegung der Art

$$A = LL^T,$$

wobei L eine linke untere Dreiecksmatrix mit positiven Zahlen auf der Diagonalen ist.

i) Berechnen Sie die "symmetrische" Zerlegung von A! Zeigen Sie ihre Rechnungsschritte für mindestens eine Zeile oder Spalte.
Tipp: Berechnen Sie die Zerlegung basierend auf Matrix-Multiplikation. D.h. stellen Sie $A = LL^T$ auf und lösen Sie die Gleichung für alle Komponenten von L.

Basierend auf Matrix-Multiplikation:

$$l_{11}^2 = 1 \Rightarrow l_{11} = 1, (l_{ii} > 0)$$

$$1 \cdot l_{21} = 4 \Rightarrow l_{21} = 4,$$

$$1 \cdot l_{31} = 1 \Rightarrow l_{31} = 1,$$

$$\dots$$

$$\begin{pmatrix} 1 & 0 & 0 \\ 4 & 2 & 0 \\ 1 & 1 & 16 \end{pmatrix} \cdot \begin{pmatrix} 1 & 4 & 1 \\ 0 & 2 & 1 \\ 0 & 0 & 16 \end{pmatrix} = \begin{pmatrix} 1 & 4 & 1 \\ 4 & 20 & 6 \\ 1 & 6 & 258 \end{pmatrix}. \qquad (2)$$

ii) Welche Zerlegung kann schneller berechnet werden, LU oder “symmetrische” LU? Denken Sie an große $n \times n$ Matrizen. Begründen Sie kurz Ihre Antwort (1-2 Sätze sind ausreichend).

Die symmetrische Zerlegung kann schneller berechnet werden, denn es muss nach nur etwa die Hälfte Unbekannte gelöst werden.

5 Inverse Fast Fourier Transformation (4 + 5 + 1 = 10 Pkt.)

Ziel dieser Aufgabe ist es, die wesentlichen Bausteine der inversen Fast Fourier Transformation zu implementieren.
Wir betrachten die komplexen Koeffizienten $c \in \mathbb{C}^n$, aus welchen wir das Inputsignal $v \in \mathbb{C}^n$ rekonstruieren wollen. Eine Überblick über die IFFT kann in den folgenden Grafiken gefunden werden:

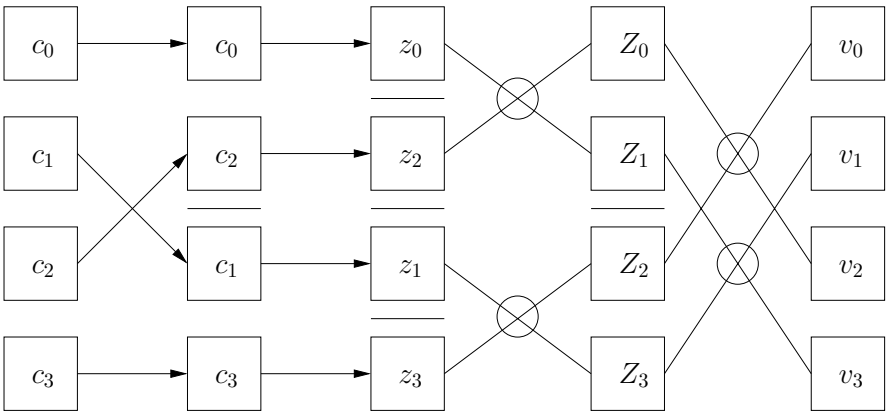

Abbildung 5: Schematischer Ablauf des IFFT-Algorithmus.

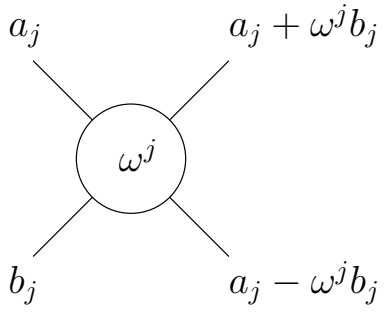

Abbildung 6: Der Butterfly-Operator.

In den folgenden Aufgabenteilen sollen die 2 Hauptschritte des Algorithmus implementiert werden: Die Sortier- und die Kombinationsphase.

Gegeben ist die folgende Implementierung der IFFT, welche die beiden Teilschritte *sortIFFT* und *combinationIFFT* aufruft.

```
public static complex[] computeIFFT(complex[] array, int n){
    sortIFFT(array, 0, n);
    complex w = getW(n);
    combinationIFFT(array, 0, n, w);
    return array;
}
```

a) Implementieren Sie die fehlenden Teile des Programmgerüsts für die Sortierphase. Die Hilfsfunktion *copy_array* kopiert hier den Inhalt des übergebenen Arrays und erstellt ein neues identisches Array. Ein Aufruf von *sortIFFT* soll hier die Elemente von *c* in dem Bereich der Elemente von *start* bis *end* - 1 direkt in *c* umgruppieren.

Hinweis: Die komplexen Zahlen können zur Vereinfachung wie normale floating-point Zahlen verwendet werden (*, +, ...).

```
public static void sortIFFT(complex[] c, int start, int end){
    complex[] c_copy = copy_array(c);
    //toDo


      for(int i = 0; i < (end - start)/2; i++){
          c[start + i] = c_copy[start + 2*i];
          c[start + (end - start)/2 + i] =
            c_copy[start+ 2*i + 1];
      }

    if(end - start > 2){
        //recursion
        //toDo

          sortIFFT(c, start, (end - start)/2);
          sortIFFT(c, start + (end - start)/2,
            end);

    }
}
```

b) Implementieren Sie die fehlenden Teile des Programgerüsts für die Kombinationsphase. Die Hilfsfunktion *copy_array* kopiert hier den Inhalt des übergebenen Arrays und erstellt ein neues identisches Array. Ein Aufruf von *sortIFFT* soll hier die Elemente des vorsortierten *c_s* in dem Bereich der Elemente von *start* bis *end* - 1 direkt in *c_s* kombinieren.

Hinweis: Die komplexen Zahlen können zur Vereinfachung wie normale floating-point Zahlen verwendet werden (*, +, ...). Die Funktion x^k kann mittels der Hilfsfunktion *pow(x,k)* berechnet werden.

```
public static void combinationIFFT(complex[] c_s, int start, int
   end, complex w){
    if(end - start > 2){
        //recursion
        //toDo

        combinationIFFT(c, start, (end - start)/2, w);
        combinationIFFT(c, start + (end - start)/2, end, w);

    }
    complex[] c_copy = copy_array(c_s);
    //toDo

    for(int i = 0; i < (end - start)/2; i++){
        c_s[start + i] = c_copy[start + i] + pow(w,i) * c_copy[
           start + i + (end - start)/2];
        c_s[start + i + (end - start)/2] = c_copy[start + i] -
           pow(w,i) * c_copy[start - i + (end - start)/2];
    }

}
```

c) Geben Sie für beide Teilschritte jeweils die Komplexität einer korrekten Implementierung an (in Landau Notation).

Beides $O(n \cdot \log(n))$

Leeres Zusatzblatt