Prof. Dr. Günther Greiner
Lehrstuhl für Graphische Datenverarbeitung
der Universität Erlangen–Nürnberg

14. Februar 2012

Algorithmik kontinuierlicher Systeme — 14. Februar 2012

Angaben zur Person (Bitte in DRUCKSCHRIFT ausfüllen!):

Name, Vorname: .....

Geburtsdatum: .....

Matrikelnummer: .....

Studienfach: .....

Nicht von der Kandidatin bzw. vom Kandidaten auszufüllen !

Bewertung:

Aufgabe	1	2	3	4	5	6	7	8	9	10
Max. Punktzahl	4	10	10	6	12	10	12	8	9	9
Erreichte Punkte										

Gesamtpunktzahl	
Note	

Organisatorische Hinweise

Die folgenden Hinweise bitte aufmerksam lesen und die Kenntnisnahme durch Unterschrift bestätigen!

- Bitte legen Sie Ihren Studentenausweis und einen Lichtbildausweis zur Personenkontrolle bereit.
- Hilfsmittel (außer Schreibmaterial **und** Taschenrechner) sind nicht zugelassen. Andere elektronische Geräte sind auszuschalten.
- Fragen zu den Prüfungsaufgaben werden grundsätzlich nicht beantwortet.
- Die Lösung einer Aufgabe muss auf das jeweilige Aufgabenblatt geschrieben werden. Sollte der Platz nicht reichen, so verwenden Sie die Zusatz-Seiten am Ende der Klausur. Fügen Sie einen Hinweis in Ihre Lösung ein, dass die Lösung auf den Zusatz-Seiten fortgesetzt wurde und beschriften Sie diese mit Namen und Aufgabennummer.
- Es können durch die Aufsicht zusätzlich Seiten eingeheftet werden, sollte mehr Platz benötigt werden. Bitte beschriften Sie den Kopf dieser Seiten mit Ihrem Namen und der Aufgabennummer. Streichen Sie alles, was nicht bewertet werden soll doppelt aus.
- Auf Ihrem Platz befinden sich einige lose Blätter Schmierpapier. Bei Bedarf können Sie zusätzliches Schmierpapier von der Aufsicht anfordern. Das Schmierpapier muss abgegeben werden, es wird aber nicht bewertet.
- Wenn Sie die Prüfung aus gesundheitlichen Gründen abbrechen müssen, so muss Ihre Prüfungsunfähigkeit durch eine Untersuchung in der Universitätsklinik nachgewiesen werden. Melden Sie sich bei der Aufsicht und lassen Sie sich das entsprechende Formular aushändigen.
- Die Bearbeitungszeit beträgt 90 Minuten, es sind alle zehn Aufgaben mit 90 Punkten zu bearbeiten.
- Überprüfen Sie die Prüfungsaufgaben auf Vollständigkeit (28 Seiten inklusive Deckblatt) und einwandfreies Druckbild.
- Vergessen Sie nicht, auf dem Deckblatt die Angaben zur Person einzutragen und die Erklärungen auf dieser Seite zu unterschreiben.
- Viel Erfolg!

Erklärungen

Durch meine Unterschrift bestätige ich den Empfang der vollständigen Klausurunterlagen und die Kenntnisnahme der obigen Informationen.

Erlangen, 14. Februar 2012 .....
(Unterschrift)

Ich bin damit einverstanden, dass mein Prüfungsergebnis unter Angabe der Matrikelnummer anonymisiert veröffentlicht wird:

ja: ☐ nein: ☐

Erlangen, 14. Februar 2012 .....
(Unterschrift)

1 Komplexität (4 Punkte)

Bestimmen Sie die Komplexität der folgenden Operationen. Dabei sollen Sie annehmen, dass die Länge der Spaltenvektoren n und die Größe von Matrizen $n \times n$ ist.

Lösen von $\mathbf{A}\vec{x} = \vec{b}$, für Tridiagonalmatrizen $\mathbf{A}$	**O** ( )
Auswerten eines Polynominterpolanten vom Grad n gegeben bzgl. der Lagrange-Basis	**O** ( )
Bestimmung der QR-Zerlegung mittels Jacobi-Rotationen	**O** ( )
Bestimmung der quadrierten Frobenius Norm $\|\|\mathbf{A}\|\|_F^2$ einer vollbesetzten Matrix $\mathbf{A}$	**O** ( )
Berechnung des Skalarprodukts zweier Vektoren	**O** ( )
Berechnung der Inversen $\mathbf{A}^{-1}$ für Diagonalmatrizen $\mathbf{A}$	**O** ( )
Lösen von $\mathbf{A}\vec{x} = \vec{b}$, für untere Dreiecksmatrizen $\mathbf{A}$	**O** ( )
Bestimmen der Einsnorm $\|\|\vec{x}\|\|_1$ eines Vektors $\vec{x}$	**O** ( )

2 Speicherung dünn besetzter Matrizen (10 Punkte)

Gegeben ist die Matrix **M**:

$$\mathbf{M} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 5 & 0 & 6 & 0 \\ 0 & -2 & -1 & 0 & 0 \end{bmatrix}.$$

a) Sie sollen **M** im `CRS`-Format abspeichern (Compressed Row Storage). Geben Sie hierzu die interne Datenstruktur an. Die Indizierung beginnt stets bei 0.

b) Wie ergibt sich die Dimension einer $m \times n$ Matrix aus dem `CRS`-Format?

c) Gegeben ist nun eine Matrix mit Zeilenanzahl 4 im `CCS`-Format (Compressed Column Storage):

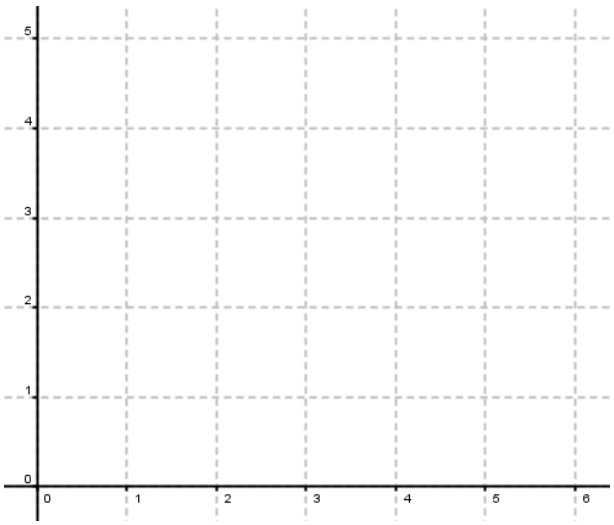
Wertearray: `4 3 7 -2 8 4`
Zeilenindexarray: `1 2 3 1 2 3`
Spaltenpointerarray: `0 0 1 3 5 6`

Geben Sie die Matrix an. Die Indizierung beginnt stets bei 0.

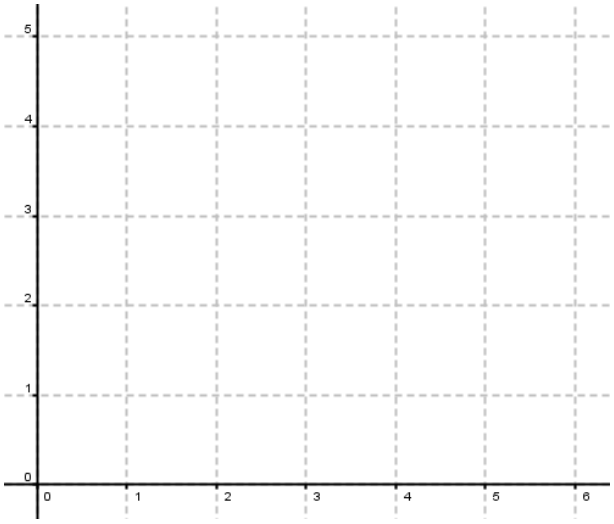
3 Numerische Integration (10 Punkte)

Gegeben sei die Funktion $f(x) = (x-3)^2 + 1$. Das Integral $I_1^5(f) = \int_1^5 f(x)\,dx$ soll näherungsweise mittels Numerischer Integration bestimmt werden.

a) Wenden Sie die iterierte Trapezregel für die Partition $\{1, 3, 5\}$ an um $I_1^5(f)$ näherungsweise zu bestimmen (Rechnung!). Zeichnen Sie zusätzlich den Graphen von $f(x)$ und schraffieren Sie die von der Integrationsregel berechnete Fläche.

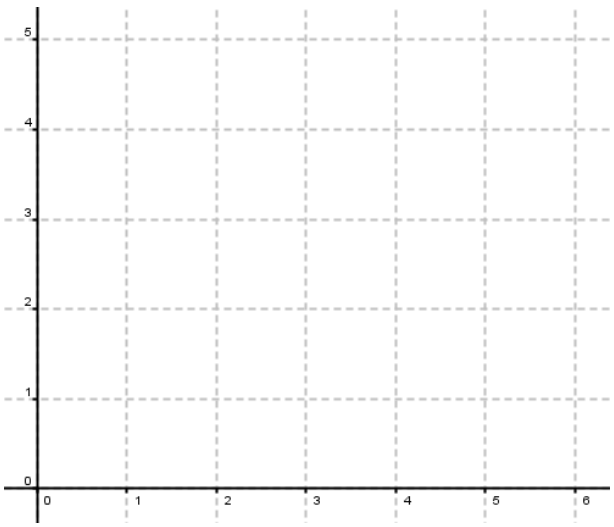


b) Wenden Sie die iterierte Trapezregel für die Partition $\{1, 2, 3, 4, 5\}$ an um $I_1^5(f)$ näherungsweise zu bestimmen (Rechnung!). Zeichnen Sie zusätzlich den Graphen von $f(x)$ und schraffieren Sie die von der Integrationsregel berechnete Fläche.



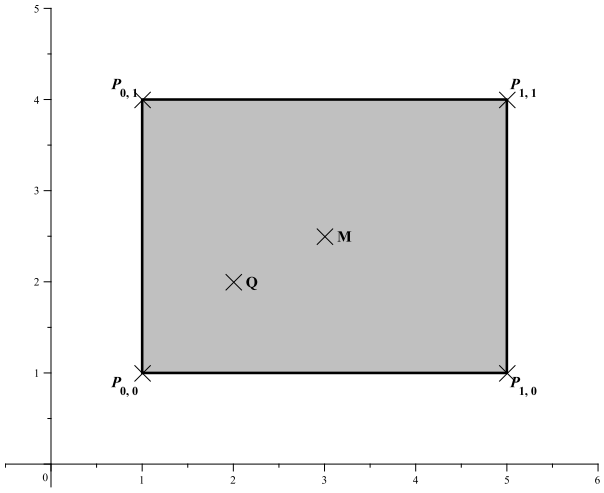
c) Benutzen Sie die Ergebnisse von a) und b) und führen Sie einen Schritt des ROMBERG-Verfahrens durch.

d) Wenden Sie die SIMPSON-Regel für die Partition $\{1, 3, 5\}$ an um $I_1^5(f)$ näherungsweise zu bestimmen (Rechnung!). Zeichnen Sie zusätzlich den Graphen von $f(x)$ und schraffieren Sie die von der Integrationsregel berechnete Fläche.



4 Bilineare und Transfinite Interpolation (6 Punkte)

a) In den Ecken eines Rechtecks $P_{0,0} = [1,1]$, $P_{1,0} = [5,1]$, $P_{1,1} = [5,4]$, $P_{0,1} = [1,4]$ sind vier Werte f_{00}, f_{10}, f_{11} und f_{01} gegeben, diese sollen bilinear interpoliert werden:



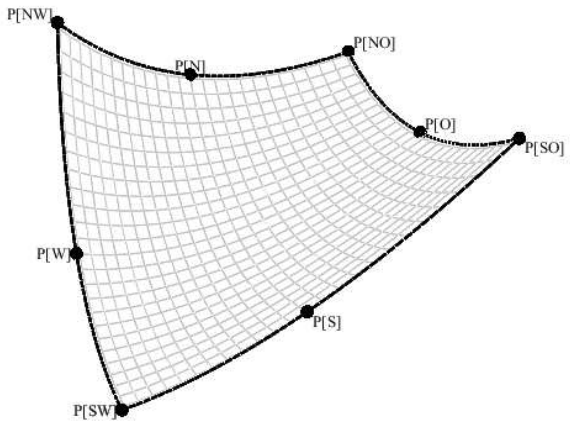
Die Werte des Interpolanten in einem Punkt P kann man als gewichtete Summe schreiben:
$f_P = w_{00}^P f_{00} + w_{10}^P f_{10} + w_{11}^P f_{11} + w_{01}^P f_{01}$

Bestimmen Sie die Gewichte für die beiden folgenden Punkte:

$Q = [2,2]:$ $\quad w_{00}^Q =$ $\qquad , w_{10}^Q =$ $\qquad , w_{11}^Q =$ $\qquad , w_{01}^Q =$ $\qquad ;$

$M = [3,2.5]:$ $\quad w_{00}^M =$ $\qquad , w_{10}^M =$ $\qquad , w_{11}^W =$ $\qquad , w_{01}^M =$ $\qquad ;$

b) Man betrachte das COONS-Patch zu vier gegebenen Randkurven $C_S(s)$, $C_O(t)$, $C_N(s)$ und $C_W(t)$, siehe dazu die folgende Abbildung (Notation: **S**üd, **O**st, **N**ord, **W**est):



Der Mittelpunkt des COONS-Patches, d.h. der Punkt $F(\frac{1}{2}, \frac{1}{2})$, kann als Linearkombination der vier Kurven-Mittelpunkte

$P_S = C_S(\frac{1}{2})$, $P_O = C_O(\frac{1}{2})$, $P_N = C_N(\frac{1}{2})$, $P_W = C_W(\frac{1}{2})$,

und der vier Eckpunkte:

$P_{SW} = C_S(0) = C_W(0)$, $P_{SO} = C_S(1) = C_O(0)$, $P_{NO} = C_O(1) = C_N(1)$, $P_{NW} = C_W(1) = C_N(0)$,

beschrieben werden:

$$F(\frac{1}{2}, \frac{1}{2}) = w_{SW}P_{SW} + w_S P_S + w_{SO}P_{SO} + w_O P_O + w_{NO}P_{NO} + w_N P_N + w_{NW}P_{NW} + w_W P_W$$

Bestimmen Sie die acht Gewichte w_{SW}, ... w_W.

5 Singulärwertzerlegung (12 Punkte)

Gegeben sei die Singulärwertzerlegung (SVD) der Matrix $\mathbf{A} = \mathbf{USV}^T$:

$$\underbrace{\frac{1}{12}\begin{bmatrix}-3 & -3 & -3 & -3\\ -6 & -6 & 6 & 6\\ -2 & 2 & 2 & -2\end{bmatrix}}_{\mathbf{A}} = \underbrace{\begin{bmatrix}0 & -1 & 0\\ -1 & 0 & 0\\ 0 & 0 & 1\end{bmatrix}}_{\mathbf{U}} \cdot \underbrace{\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & \frac{1}{2} & 0 & 0\\ 0 & 0 & \frac{1}{3} & 0\end{bmatrix}}_{\mathbf{S}} \cdot \underbrace{\begin{bmatrix}\frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2}\\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2}\\ -\frac{1}{2} & \frac{1}{2} & \frac{1}{2} & -\frac{1}{2}\\ -\frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & \frac{1}{2}\end{bmatrix}^T}_{\mathbf{V}^T} .$$

a) Bestimmen Sie für die Matrix $\mathbf{A}$:

- die Singulärwerte:

- den Rang:

- den Bildraum:

- den Kern (welche Vektoren spannen den Kern auf?):

- und die Konditionszahl bzgl. der Spektralnorm-Norm $|||\cdot|||_2$:

Zur Erinnerung:

$$\underbrace{\frac{1}{12}\begin{bmatrix}-3 & -3 & -3 & -3\\ -6 & -6 & 6 & 6\\ -2 & 2 & 2 & -2\end{bmatrix}}_{\mathbf{A}} = \underbrace{\begin{bmatrix}0 & -1 & 0\\ -1 & 0 & 0\\ 0 & 0 & 1\end{bmatrix}}_{\mathbf{U}} \cdot \underbrace{\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & \frac{1}{2} & 0 & 0\\ 0 & 0 & \frac{1}{3} & 0\end{bmatrix}}_{\mathbf{S}} \cdot \underbrace{\begin{bmatrix}\frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2}\\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2}\\ -\frac{1}{2} & \frac{1}{2} & \frac{1}{2} & -\frac{1}{2}\\ -\frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & \frac{1}{2}\end{bmatrix}^T}_{\mathbf{V}^T}.$$

b) Lösen Sie die folgende Gleichung mittels der Pseudo-Inversen.

$$\mathbf{A} \cdot \vec{x} = \begin{bmatrix}1\\ 0\\ -2\end{bmatrix}.$$

Zur Erinnerung:

$$\underbrace{\frac{1}{12}\begin{bmatrix}-3 & -3 & -3 & -3\\ -6 & -6 & 6 & 6\\ -2 & 2 & 2 & -2\end{bmatrix}}_{\mathbf{A}} = \underbrace{\begin{bmatrix}0 & -1 & 0\\ -1 & 0 & 0\\ 0 & 0 & 1\end{bmatrix}}_{\mathbf{U}} \cdot \underbrace{\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & \frac{1}{2} & 0 & 0\\ 0 & 0 & \frac{1}{3} & 0\end{bmatrix}}_{\mathbf{S}} \cdot \underbrace{\begin{bmatrix}\frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2}\\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2}\\ -\frac{1}{2} & \frac{1}{2} & \frac{1}{2} & -\frac{1}{2}\\ -\frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & \frac{1}{2}\end{bmatrix}^T}_{\mathbf{V}^T} .$$

c) Bestimmen Sie die Matrix vom Rang 1, die **A** im Sinne der FROBENIUS-Norm am besten approximiert.

6 Programmierung: LU-Zerlegung (10 Punkte)

Die Klasse `Solver` stellt grundlegende Funktionen zum Lösen von linearen Gleichungssystemen bereit. Dabei soll die Klasse `Matrix` verwendet werden. Sie sollen dabei die entsprechenden Methoden implementieren (verwenden Sie `C++` Syntax). Entgegen der Übungen ist **keine** Fehlerbehandlung erforderlich.

```
class Solver {
  public:
    Solver();
    ~Solver();

    //! Berechnet die LU Zerlegung der quadratischen Matrix m
    void decomposeLU(const Matrix &m, Matrix &l, Matrix &u);

    //! Berechnet ly = b wobei y das Ergebnis ist; l ist eine untere Dreiecksmatrix
    void forwardSubstitution(const Matrix &l, const Matrix &b, Matrix &y);

    //! Berechnet ux = y wobei x das Ergebnis ist; u ist eine obere Dreiecksmatrix
    void backwardSubstitution(const Matrix &u, const Matrix &y, Matrix &x);

    //! Loest das lineare System mx = b mittles LU-Zerlegung
    void solveSystem(const Matrix &m, const Matrix &b, Matrix &x);

    //! Berechnet die Determinante der Matrix m
    float calcDeterminant(const Matrix &m);
};

class Matrix {
public:
    //! Konstruktor: Baut eine uninitialisierte Matrix
    Matrix(unsigned int height, unsigned int width);

    //! Copy Konstruktor
    Matrix(const Matrix &other);

    //! Destruktor
    ~Matrix();

    unsigned int getHeight();                                    //! Anzahl der Zeilen
    unsigned int getWidth();                                     //! Anzahl der Spalten
    void setZero();                                              //! Setzt alle Matrixwerte auf 0
    void setIdentity();                                          //! Initialisiert als Einheitsmatrix
    float&  operator()(unsigned int m, unsigned int n);          //! Mutator (m=Zeile, n=Spalte)
    float   operator()(unsigned int m, unsigned int n) const;    //! Akzessor (m=Zeile, n=Spalte)
    Matrix& operator=(const Matrix &other);                      //! Zuweisungsoperator
    ...
private:
    ...
};
```

Bitte wenden!

a) Implementieren Sie die Methode `void Solver::decomposeLU(const Matrix &m, Matrix &l, Matrix &u)`. Diese soll eine LU-Zerlegung der Matrix `const Matrix &m` berechnen und in den entsprechenden Parametern `Matrix &l`, `Matrix &u` speichern. Sie können davon ausgehen, dass die übergebene Matrix quadratisch ist. Sie können ebenfalls davon ausgehen, dass die Ergebnismatrizen `Matrix &l` und `Matrix &u` schon die richtige Größe haben.

```
void Solver::decomposeLU(const Matrix &m, Matrix &l, Matrix &u) {









}
```

b) Implementieren sie die Methode `void Solver::solveSystem(const Matrix &m, const Matrix &b, Matrix &x)`. Diese soll das Gleichungssystem $m \cdot x = b$ lösen (`Matrix &x` soll berechnet werden). Verwenden Sie dazu passende Methoden der Klasse `Solver`. Sie können davon ausgehen, dass die Ergebnismatrix `Matrix &x` schon die richtige Größe hat.

```
void Solver::solveSystem(const Matrix &m, const Matrix &b, Matrix &x) {









}
```

c) Implementieren Sie die Methode `float Solver::calcDeterminant(const Matrix &m)`. Diese soll die Determinante von `const Matrix &m` berechnen und als Rückgabewert der Funktion zurückgeben. Sie können dazu passende Methoden der Klasse `Solver` verwenden.

```
float Solver::calcDeterminant(const Matrix &m) {









}
```

7 Programmierung: Bildbearbeitung (12 Punkte)

In dieser Aufgabe sollen verschiedene Operationen auf Graustufenbildern, repräsentiert durch die Klasse `Image`, implementiert werden (verwenden Sie `C++` Syntax). Die Bilder sind immer quadratisch und die zugehörigen Grauwerte sind als `float` gespeichert. Entgegen der Übungen ist **keine** Fehlerbehandlung erforderlich.

```
class Image {
public:
    //! Konstruktor: Baut ein uninitialisiertes, QUADRATISCHES Bild der Groesse dim x dim
    Image(unsigned int dim);

    //! Copy Konstruktor
    Image(const Image &other);

    //! Destruktor
    ~Image();

    //! Gibt die Anzahl der Zeilen/Spalten zurueck (das Bild ist quadratisch)
    unsigned int getDimension();

    float& operator()(unsigned int m, unsigned int n);       //! Mutator  (m=Zeile, n=Spalte)
    float  operator()(unsigned int m, unsigned int n) const; //! Akzessor (m=Zeile, n=Spalte)
    Image& operator=(const Image &other);                    //! Zuweisungsoperator
    ...
private:
    ...
};
```

Bitte wenden!

a) Implementieren Sie die folgende Funktion, welche für das übergebene Bild `const Image &image` die zweiten diskreten Ableitungen in x-Richtung (Breite) berechnet. Verwenden Sie dazu einen passenden Differenzenquotienten zweiter Ordnung. Achten Sie dabei auf korrekte Randbehandlung, indem Sie die entsprechenden zweiten Ableitungen auf 0 setzen. Sie können davon ausgehen, dass das Bild `Image &result` schon die richtige Größe hat.

```
void computeDerivativeXX(const Image &image, Image &result) {




}
```

b) Implementieren Sie die folgende Funktion, welche für das übergebene Bild `const Image &image` den diskreten Laplace-Operator berechnet. Sie können davon ausgehen, dass das Bild `Image &result` schon die richtige Größe hat. Verwenden Sie dazu die obige Funktion aus Teilaufgabe a), sowie die bereits gegebene Funktion `void computeDerivativeYY(const Image &image, Image &result)`.

```
void computeLaplacian(const Image &image, Image &result) {




}
```

c) Implementieren Sie folgende Funktion, welche die partielle Differentialgleichung $\Delta I = 0$ auf dem Gebiet Ω mit Hilfe des Jaccobi Verfahrens löst. Dabei ist I das übergebene Bild `const Image &source`. Das Gebiet Ω ist durch die Maske `const Image &mask` gegeben. Diese hat die selbe Größe wie das übergebene Bild. Dabei sind Werte der Maske im Inneren von Ω gleich $1.0f$, und außerhalb $0.0f$. Sowohl die Randbedingungen (außerhalb von Ω) als auch die Startwerte (innerhalb von Ω) sind bereits in I entsprechend gesetzt. Führen Sie `unsigned int iterations` Jaccobi Schritte durch. Das Ergebnis soll am Ende als Rückgabewert der Funktion zurückgegeben werden.

```
Image solveLaplacianEquation(const Image &source, const Image &mask, unsigned int iterations) {

}
```

8 Nichtlineare Optimierung (8 Punkte)

Betrachten Sie die Matrix

$$\mathbf{A} = \begin{bmatrix} 2 & -1 \\ -1 & 1 \end{bmatrix}$$

und das quadratische Funktional

$$Q(\vec{x}) = \vec{x}^T \mathbf{A} \vec{x} + 2\vec{b}^T \vec{x} + \gamma, \qquad \vec{b} = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \ \gamma = 2 \ .$$

a) Führen Sie zwei Schritte des Gradientenverfahrens zur Bestimmung des Minimums von Q durch. Beginnen Sie im Punkt $\vec{x}_0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ und verwenden Sie die optimale Schrittweite $t_i := -\frac{(\mathbf{A}\vec{x}_i + \vec{b})^T \vec{s}_i}{\vec{s}_i^T \mathbf{A} \vec{s}_i}$ für die Suchrichtung $\vec{s}_i$.

Zur Erinnerung:

$$\mathbf{A} = \begin{bmatrix} 2 & -1 \\ -1 & 1 \end{bmatrix}$$

b) Bestimmen Sie eine zu $\begin{bmatrix} 1 \\ 1 \end{bmatrix}$ $\mathbf{A}$-orthogonale Richtung.

c) Erklären Sie für eine konvergente Folge (x_i) mit Grenzwert x^* die Begriffe:

- Konvergenzordnung:

- lineare Konvergenz:

- quadratische Konvergenz:

9 Interpolation (9 Punkte)

a) Die folgenden Daten sollen interpoliert werden:

i	0	1	2	3
x_i	1	3	5	7
y_i	0	-2	2	-2

Bestimmen Sie den (stückweise) linearen Interpolanten $L : [1, 7] \to \mathbb{R}$

$$L(x) = \begin{cases} \\ \\ \\ \\ \\ \\ \end{cases}$$

b) Was ist der Unterschied zwischen Interpolation und Approximation?

c) Gegeben sei die Funktion $g(x) = 4x^5 - 3x^2 + 8$. Wieviele Datenpunkte sind nötig, um die Funktion mittels Polynominterpolation exakt zu rekonstruieren?

d) Gegeben sei die Funktion $f(x) = \sin(x)$. Werten Sie zunächst die Funktion an den Stellen $x_0 = 0.0$, $x_1 = \frac{1}{2}\pi$, $x_2 = \frac{3}{2}\pi$, $x_3 = 2\pi$ aus. Berechnen Sie nun die Koeffizienten des Polynoms $p(x)$ in der Newton-Basis welches die Daten $(x_i, f(x_i))$ $(0 \leq i \leq 3)$ interpoliert. Verwenden Sie dabei das Dreiecks-Schema von Aitken-Neville. Geben Sie anschließend das Polynom $p(x)$ an (kein Ausmultiplizieren nötig).

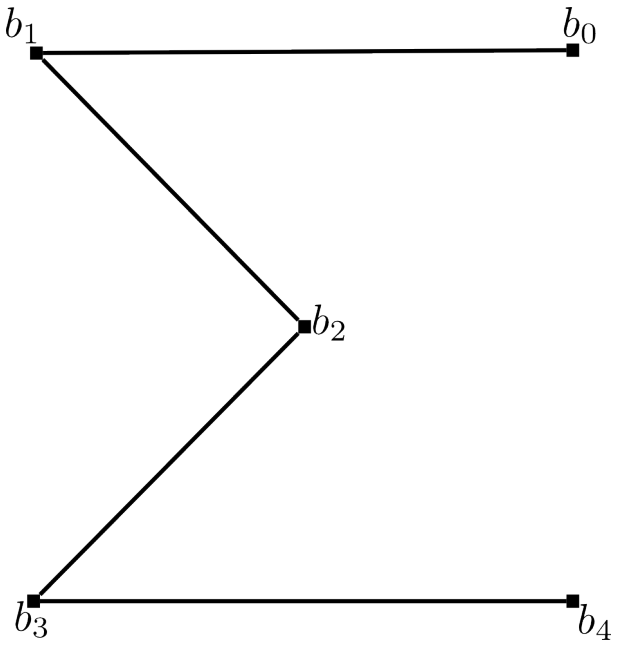
10 Bézier-Kurven (9 Punkte)

a) Betrachten Sie die BEZIER-Kurve $C(t)$ vom Grad 4 mit folgenden Kontrollpunkten:

$$\vec{b}_0 = \begin{bmatrix} 88 \\ 88 \end{bmatrix}, \quad \vec{b}_1 = \begin{bmatrix} 0 \\ 88 \end{bmatrix}, \quad \vec{b}_2 = \begin{bmatrix} 44 \\ 44 \end{bmatrix}, \quad \vec{b}_3 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad \vec{b}_4 = \begin{bmatrix} 88 \\ 0 \end{bmatrix}.$$

Werten Sie diese Kurve mit Hilfe des DECASTELJAU-Algorithmus an der Stelle $t = \frac{1}{2}$ aus (Rechnung!).

b) Führen Sie den DECASTELJAU-Algorithmus zur Berechnung von $C(\frac{3}{4})$ **geometrisch** durch. Ergänzen Sie dazu die nachfolgende Abbildung.



c) Nun sei eine BEZIER-Kurve $C(u) = \sum_{i=0}^{n} b_i B_i^n(u)$ im 3D-Raum vom Grad n gegeben. Zeigen Sie, dass $\Phi(C(u)) = \sum_{i=0}^{n} \Phi(b_i) B_i^n(u)$ (Formeigenschaft *Affine Invarianz*) gilt. Dabei ist Φ eine affine Abbildung $\Phi(x) = \mathbf{A}x + t$ mit $\mathbf{A} \in \mathbb{R}^{3\times3}$ und $t \in \mathbb{R}^3$.

d) Nennen Sie drei weitere Formeigenschaften von BEZIER-Kurven neben der *Affinen Invarianz*.