TN
Teilnehmernummer
(von Platzkarte)

Lehrstuhl für Graphische Datenverarbeitung • Friedrich-Alexander Universität Erlangen-Nürnberg
Prof. Dr. Günther Greiner

21. Juli 2016

Klausur
Algorithmik kontinuierlicher Systeme – 21. Juli 2016

Angaben zur Person (Bitte in DRUCKSCHRIFT ausfüllen!):

Name ..... Vorname .....

Mat.-Nummer .....

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

- Die Bearbeitungszeit beträgt **90 Minuten**.
- Hilfsmittel (außer Schreibmaterial und Taschenrechner) sind nicht zugelassen.
- **Überprüfen Sie die Prüfungsangaben auf Vollständigkeit (28 Seiten inklusive Deckblatt) und einwandfreies Druckbild!**
- Schreiben Sie deutlich und ausschließlich mit blauer oder schwarzer Tinte. Unleserliche Antworten gehen **nicht** in die Bewertung ein. Benutzen Sie **nur für Zeichnungen** einen Bleistift.
- Sollte der Platz nicht reichen, so verwenden Sie zunächst die Zusatz-Seiten am Ende der Klausur. Fügen Sie einen Hinweis in Ihre Lösung ein, dass die Lösung auf den Zusatz-Seiten fortgesetzt wurde. Sollte der Platz immer noch nicht ausreichen, so müssen Sie bei der Aufsicht weitere Zusatz-Seiten anfordern und einheften lassen. **Es werden nur leere Blätter eingeheftet!**
- Falls Sie zusätzliches Schmierpapier benötigen, melden Sie sich bitte bei der Aufsicht.
  Das Schmierpapier wird eingesammelt, aber nicht bewertet!
- Wenn Sie die Prüfung aus gesundheitlichen Gründen abbrechen müssen, so muss Ihre Prüfungsunfähigkeit durch eine ärztliche Untersuchung nachgewiesen werden. Melden Sie sich bei der Aufsicht und lassen Sie sich das entsprechende Formular aushändigen.
- Die angegebene Punkteverteilung gilt unter Vorbehalt.

Erklärung

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

Erlangen, 21. Juli 2016 .....
(Unterschrift)

Nicht von der Kandidatin bzw. vom Kandidaten auszufüllen !

Aufgabe	1	2	3	4	5	6	7	8	9	10	**Gesamt**
Max. Punktzahl	6	7	5	14	9	11	7	10	13	8	90
Erreichte Punkte											
Zweitkorrektor											

1 Theoriefragen (6 Punkte)

a) Beantworten Sie die folgenden Fragen! Schreiben Sie ihre Antwort in die rechte Spalte der Tabelle!

Welche Komplexität hat die Matrix-Vektor-Multiplikation $\mathbf{A}\vec{x}$ wenn $\mathbf{A}$ eine vollbesetzte $(n \times n)$-Matrix und $\vec{x} \in \mathbb{R}^n$ ist?	$\mathcal{O}($ $)$
Welche Komplexität hat die Bestimmung der LR-Zerlegung einer $(n \times n)$-Matrix?	$\mathcal{O}($ $)$
Welche Komplexität hat das Bestimmen der Determinante einer $(n \times n)$-Matrix, wenn die QR-Zerlegung $\mathbf{A} = \mathbf{QR}$ gegeben ist?	$\mathcal{O}($ $)$
Welche Komplexität hat die Durchführung eines Iterationsschrittes des Gauss-Seidel-Verfahren für eine vollbesetzte $(n \times n)$-Matrix?	$\mathcal{O}($ $)$
Wie groß ist der Approximationsfehler der iterierten Simpson-Regel für die Schrittweite h ?	$\mathcal{O}($ $)$
Wieviele Kontrollpunkte entstehen bei einem midpoint-subdivision Schritt einer Bézier-Kurve vom Grad n ?	
Welche Komplexität hat die Bestimmung eines einzelnen Punktes auf einer Bézier-Kurve mit n Kontrollpunkten mit dem Algorithmus von De Casteljau?	$\mathcal{O}($ $)$
Wieviele Iterationen benötigt das CG-Verfahren bei exakter Arithmetik zum Lösen eines Gleichungssystems $\mathbf{A}\vec{x} = \vec{b}$, wobei $\mathbf{A}$ eine symmetrische, positiv definite $(n \times n)$-Matrix ist?	

Gegeben seien drei Matrizen $\mathbf{A} \in \mathbb{R}^{k \times l}$, $\mathbf{B} \in \mathbb{R}^{l \times m}$, $\mathbf{C} \in \mathbb{R}^{m \times n}$ mit $k > l > m > n$.

Welche Dimension besitzt die Ergebnismatrix $\mathbf{D} = \mathbf{A} \cdot \mathbf{B} \cdot \mathbf{C}$?	($\times$)
Welche der folgenden beiden Operationen besitzt den geringeren Aufwand? 1) $\mathbf{A} \cdot (\mathbf{B} \cdot \mathbf{C})$ 2) $(\mathbf{A} \cdot \mathbf{B}) \cdot \mathbf{C}$	

2 LR- und QR-Zerlegung (7 Punkte)

a) Bestimmen Sie die LR-Zerlegung der folgenden Matrix:

$$\mathbf{A} = \begin{bmatrix} 2 & -1 & 1 & -1 \\ -2 & 2 & -2 & 2 \\ 2 & -2 & 4 & -3 \\ 0 & 1 & -3 & 5 \end{bmatrix}$$

b) Die QR-Zerlegung der Matrix $\mathbf{B} = \begin{bmatrix} 1 & -3 & -4 & -2 \\ 1 & 1 & 0 & 2 \\ -1 & -1 & -4 & 0 \\ -1 & 3 & 0 & -4 \end{bmatrix}$ ist bekannt:

$$\mathbf{B} = \underbrace{\frac{1}{2}\begin{bmatrix} 1 & -1 & -1 & -1 \\ 1 & 1 & -1 & 1 \\ -1 & -1 & -1 & 1 \\ -1 & 1 & -1 & -1 \end{bmatrix}}_{Q} \underbrace{\begin{bmatrix} 2 & -2 & 0 & 2 \\ 0 & 4 & 4 & 0 \\ 0 & 0 & 4 & 2 \\ 0 & 0 & 0 & 4 \end{bmatrix}}_{R}$$

Bestimmen Sie unter Verwendung der QR-Zerlegung von $\mathbf{B}$ die Lösung der linearen Gleichung

$$\mathbf{B}\vec{x} = \vec{b} \quad \text{für} \quad \vec{b} = [\,4\,,\ 2\,,\ 4\,,2\,]^T \ .$$

c) Gegeben ist eine penta-diagonale $n \times n$-Matrix $\mathbf{A} = [a_{ij}]$.
D.h. $a_{ij} = 0$ falls $|i - j| > 2$.

Für diese Matrix soll mit Hilfe von Jacobi-Rotationen die QR-Zerlegung bestimmt werden.

Wieviele Jacobi-Rotationen werden dazu benötigt?

$$\mathbf{C} = \begin{bmatrix} * & * & * & 0 & 0 & & \cdots & \cdots & & 0 \\ * & * & * & * & 0 & & & & & \\ * & * & * & * & * & \ddots & & & & \vdots \\ 0 & * & * & * & * & \ddots & \ddots & & & \vdots \\ 0 & 0 & * & * & * & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 & 0 \\ \vdots & & & \ddots & \ddots & \ddots & * & * & * & 0 \\ \vdots & & & & \ddots & \ddots & * & * & * & * \\ & & & & & 0 & * & * & * & * \\ 0 & & \cdots & \cdots & & 0 & 0 & * & * & * \end{bmatrix}$$

Struktur einer penta-diagonalen Matrix

3 Dünnbesetzte Matrizen (5 Punkte)

a) Speichern Sie die folgende Matrix **A** im CCS-Format ab (Indizierung beginnt bei 1).

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 7 & 0 \\ -2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 4 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

b) Die 5×7-Matrix **B** sei im CRS-Format gegeben (Indizierung beginnt bei 1):

$$\begin{aligned} \texttt{val} &= [1, 5, 3, 5, -1, 8, 4, 7, -2] \\ \texttt{col\_ind} &= [2, 3, 4, 6, 3, 4, 7, 2, 5] \\ \texttt{row\_ptr} &= [1, 2, 5, 5, 7, 10] \end{aligned}$$

Multiplizieren Sie **B** mit dem Vektor $\vec{b} = [1, 0, -2, 1, -1, -1, 2]^T$ ohne die Matrix zu rekonstruieren! Zwischenschritte müssen erkennbar sein!

c) Wie kann man eine $n \times n$ Tridiagonal-Matrix möglichst effizient speichern?

4 Anwendungen der Singulärwertzerlegung (14 Punkte)

Die Klasse `SVD` stellt grundlegende Anwendungen der Singulärwertzerlegung einer **quadratischen** Matrix **A** zur Verfügung. Dazu wird die Klasse `Matrix` verwendet. Sie sollen dabei einige Methoden der Klasse `SVD` in `C++` implementieren. Entgegen der Übungen ist **keine** Fehlerbehandlung erforderlich. Verändern Sie die Klassenstrukturen nicht, d.h. führen Sie keine neuen Attribute / Methoden ein.

```
class SVD {
public:
    //! Konstruktor: Berechnet die Singulärwertzerlegung A = USV^T
    SVD(const Matrix &A);

    //! Nutzt die SVD aus, um den Rang von A zu berechnen
    unsigned int rang() const;

    //! Löst das lineare System Ax = b mittels Pseudoinverse
    Matrix solveSystem(const Matrix &b) const;

    //! Membervariablen
    Matrix U, S, V;
};
```

```
class Matrix {
public:
    //! Konstruktor: Baut eine Matrix, deren Einträge alle 0 sind
    Matrix(unsigned int height, unsigned int width);

    unsigned int getHeight() const;  //! Anzahl der Zeilen
    unsigned int getWidth() const;   //! Anzahl der Spalten

    //! Mutator (i = Zeile, j = Spalte)
    float& operator()(unsigned int i, unsigned int j);

    //! Akzessor (i = Zeile, j = Spalte)
    float operator()(unsigned int i, unsigned int j) const;

    //! Gibt die transponierte Matrix zurück
    Matrix transposed() const;

    ...  // weitere Konstruktoren und Methoden
};
```

Hinweis: Die Indizierung beginnt bei 0. Sie können davon ausgehen, dass die Operatoren `+, -, *, /, +=, -=, *=, /=` für die Klasse `Matrix` überladen sind.

a) Der Konstruktor `SVD(const Matrix &A)` benutzt mehrere QR-Zerlegungen, um die Singulärwertzerlegung zu bestimmen. Welchen Aufwand hat die Berechnung **einer** QR-Zerlegung für eine $(n \times n)$-Matrix?

$$\mathcal{O}(\qquad)$$

b) Implementieren Sie die Methode `unsigned int SVD::rang() const`, die den Rang der **quadratischen** Matrix **A** unter Ausnutzung der Singulärwertzerlegung bestimmt und diesen zurückgibt.

```
unsigned int SVD::rang() const {
     unsigned int n = S.getHeight();        // A ist quadratisch (n x n)









}
```

c) Implementieren Sie die Methode `Matrix SVD::solveSystem(const Matrix &b) const`, die das lineare System $\mathbf{A}\vec{x} = \vec{b}$ unter Ausnutzung der Singulärwertzerlegung mittels Pseudoinverse löst und den Ergebnisvektor $\vec{x}$ zurückgibt. Sowohl $\vec{b}$ als auch $\vec{x}$ werden als $(n \times 1)$-Matrix gespeichert.

Achten Sie darauf, dass die Methode nur den Aufwand $\mathcal{O}(n^2)$ hat.

```
Matrix SVD::solveSystem(const Matrix &b) const {
    unsigned int n = S.getHeight();        // A ist quadratisch (n x n)









}
```

d) Nun soll mit Hilfe der SVD-Klasse eine Hauptkomponentenanalyse (PCA) auf einer Punktewolke durchgeführt werden. Die PCA soll in den folgenden Funktionen schrittweise implementiert werden.

Ein Punkt $\vec{p} \in \mathbb{R}^m$ wird als Matrix $\mathbf{P} \in \mathbb{R}^{m \times 1}$ dargestellt.

Die Methode `Matrix computeMean(const std::vector<Matrix> &points)` soll für die übergebenen Punkte `points` den Mittelwert `mean` bestimmen und zurückgeben.

```
Matrix computeMean(const std::vector<Matrix> &points) {
    unsigned int m = points[0].getHeight();        // Dimension der Punkte
    unsigned int n = points.size();                // Anzahl der Punkte
    Matrix mean(m, 1);                             // Ergebnisvariable









}
```

Die Methode `Matrix computeCovariance(const std::vector<Matrix> &points)` soll für die übergebenen Punkte unter Ausnutzung der Methode `computeMean(...)` die Kovarianzmatrix `cov` bestimmen und zurückgeben.

```
Matrix computeCovariance(const std::vector<Matrix> &points) {
    unsigned int m = points[0].getHeight();        // Dimension der Punkte
    unsigned int n = points.size();                // Anzahl der Punkte
    Matrix cov(m, m);                              // Ergebnisvariable









}
```

In der Methode `computePCA(const std::vector<Matrix> &points, Matrix &EW, Matrix &EV)` soll die PCA unter Ausnutzung der SVD-Klasse durchgeführt werden und die Basisvektoren in `EV` zurückgegeben werden. Die Eigenwerte sollen in der Diagonalmatrix `EW` gespeichert werden.

Benutzen Sie die Methode `computeCovariance(...)`.

```
void computePCA(const std::vector<Matrix> &points, Matrix &EW, Matrix &EV) {

}
```

e) Nennen Sie **zwei** Anwendungen der PCA.

5 Interpolation (9 Punkte)

Gegeben seien folgende Punkte:

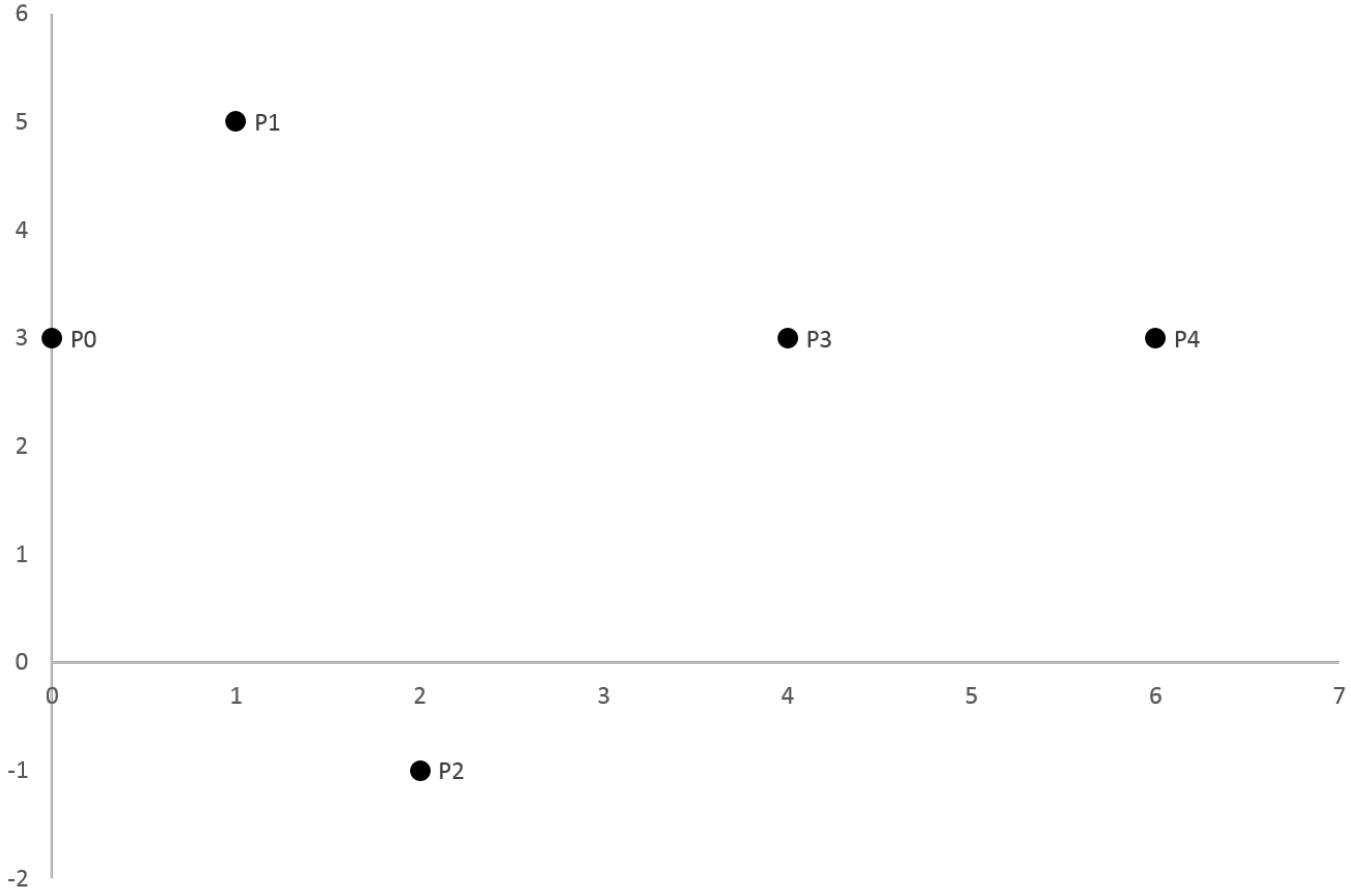
i	0	1	2	3
x_i	0	1	2	4
y_i	3	5	-1	3

a) Bestimmen Sie die LAGRANGE-Polynome und geben Sie die Koeffizienten des Interpolationspolynoms an. Hinweis: Die Basis-Funktionen müssen nicht aus multipliziert werden!

b) Bestimmen Sie die NEWTON-Polynome und geben Sie die Koeffizienten des Interpolationspolynoms an. Hinweis: Die Basis-Funktionen müssen nicht aus multipliziert werden!

c) Werten Sie das Interpolationspolynom an den Stellen $a = 1.5$ und $b = 4$ aus.

d) Skizzieren Sie die beiden mittleren Segmente des Catmull-Rom-Interpolanten.



6 Iterative Lösungsverfahren (11 Punkte)

Gegeben seien die 3×3-Matrix $\mathbf{A}$ sowie der Vektor $\vec{b}$ mit

$$\mathbf{A} = \begin{bmatrix} 2 & 1 & 1 \\ 1 & 3 & 2 \\ -1 & 1 & 1 \end{bmatrix} \qquad \vec{b} = \begin{bmatrix} 6 \\ 6 \\ -6 \end{bmatrix}.$$

a) Führen Sie **zwei** Schritte des JACOBI-Verfahrens zur Lösung von $\mathbf{A}\vec{x} = \vec{b}$ durch. Verwenden Sie als Startvektor $\vec{x}^0 = [0, 0, 0]^T$.

b) Die Lösung eines linearen Gleichungssystems $\mathbf{A}\vec{x} = \vec{b}$, mit $\mathbf{A} \in \mathbb{R}^{n\times n}$, $\vec{x}, \vec{b} \in \mathbb{R}^n$ wird iterativ mittels Jacobi-Verfahren gelöst.

In der folgenden Tabelle ist zu der Anzahl der Unbekannten n die Anzahl der Iterationen eingetragen, die nötig sind, um die angegebenen Fehlertoleranzen τ einzuhalten.

	$\tau = 10^{-3}$	$\tau = 10^{-4}$	$\tau = 10^{-5}$
$n = 10$	154	207	
$n = 20$	310		
$n = 30$			

Vervollständigen Sie diese Tabelle und schätzen Sie insbesondere die Anzahl der Iterationen, die notwendig ist, um bei $n = 30$ Unbekannten die Fehlertoleranz $\tau = 10^{-5}$ einzuhalten.

c) Schätzen Sie nun für dieses Problem die Anzahl der Iterationen, die nötig sind, wenn man das System mit dem Gauss-Seidel-Verfahren iterativ löst und zwar für $n = 40$ Unbekannte und Fehlertoleranz $\tau = 10^{-3}$.

	$\tau = 10^{-3}$
$n = 40$	

d) Nennen Sie **ein** Konvergenzkriterium für das Gauss-Seidel-Verfahren.

e) Implementieren Sie eine Methode `Matrix gaussSeidel(const Matrix &A, const Matrix &b, unsigned int nIterations)`, die für eine übergebene **quadratische** Matrix **A** und einen Vektor $\vec{b}$ das Gauss-Seidel-Verfahren mit `nIterations` Iterationen ausführt. Der Vektor $\vec{b}$ wird als $(n \times 1)$-Matrix gespeichert.

Verwenden Sie `C++`-Syntax.

Hinweis: Die Indizierung beginnt bei 0. Wie in den Programmierübungen können auf die Matrix-Elemente mit dem `operator()(unsigned int row, unsigned int column)` zugegriffen werden, z.B. liefert `A(0, 1)` das Element $a_{0,1}$.

Entgegen der Übungen ist **keine Fehlerbehandlung** erforderlich.

```
Matrix gaussSeidel(const Matrix &A, const Matrix &b, unsigned int nIterations) {
    unsigned int n = A.getHeight();        // A ist quadratisch (n x n)

    Matrix x(n, 1);









    return x;
}
```

f) Implementieren Sie eine Methode `bool isDiagonallyDominant(const Matrix &A)`, die überprüft, ob die **quadratische** Matrix **A** die Eigenschaft der **strikten** (starken) Diagonaldominanz erfüllt.

Verwenden Sie `C++`-Syntax.

Hinweis: Die Indizierung beginnt bei 0. Wie in den Programmierübungen können auf die Matrix-Elemente mit dem `operator()(unsigned int row, unsigned int column)` zugegriffen werden, z.B. liefert `A(0, 1)` das Element $a_{0,1}$.

Um den Absolutbetrag $|x|$ eines Skalars x zu berechnen, können Sie `abs(x)` verwenden.

Entgegen der Übungen ist **keine Fehlerbehandlung** erforderlich.

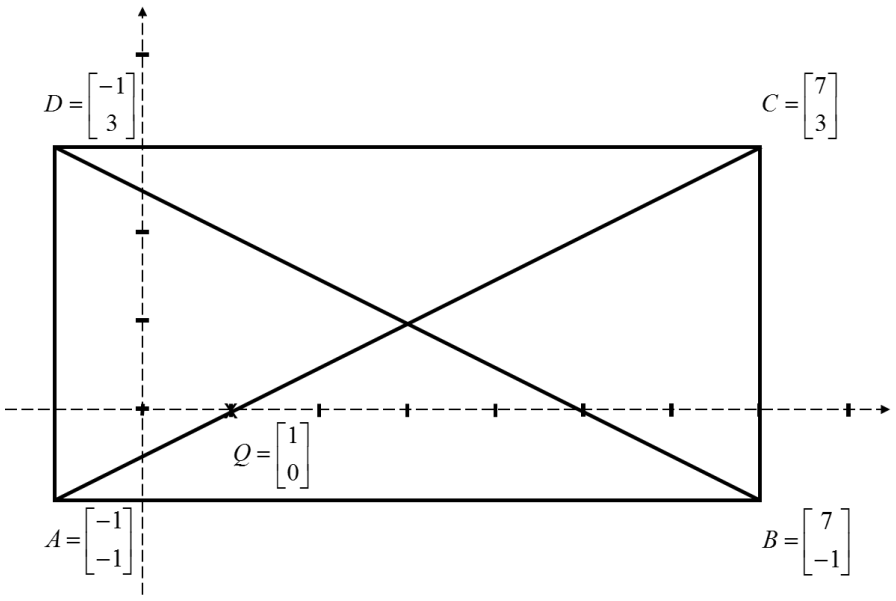
```
bool isDiagonallyDominant(const Matrix &A) {
    unsigned int n = A.getHeight();        // A ist quadratisch (n x n)
```

```
}
```

7 Multivariate Interpolation, Baryzentrische Koordinaten (7 Punkte)

Gegeben sind die vier Punkte $A = \begin{bmatrix} -1 \\ -1 \end{bmatrix}$, $B = \begin{bmatrix} 7 \\ -1 \end{bmatrix}$, $C = \begin{bmatrix} 7 \\ 3 \end{bmatrix}$, $D = \begin{bmatrix} -1 \\ 3 \end{bmatrix}$.

Sie bilden ein Rechteck sowie mehrere Dreiecke: $\Delta(A, B, C)$, $\Delta(B, C, D)$, $\Delta(C, D, A)$ und $\Delta(D, A, B)$.



a) Bestimmen Sie die baryzentrischen Koordinaten des Punktes $Q = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ bezüglich der folgenden Dreiecke.

Tipp: Die Lösung kann geometrisch bestimmt werden.

- Bezüglich des Dreiecks $\Delta(A, B, C)$ ($Q = \alpha_1 A + \beta_1 B + \gamma_1 C$) :

- Bezüglich des Dreiecks $\Delta(D, A, B)$ ($Q = \delta_2 D + \alpha_2 A + \beta_2 B$) :

- Bezüglich des Dreiecks $\Delta(B, C, D)$ ($Q = \beta_3 B + \gamma_3 C + \delta_3 D$) :

b) In den Punkten A, B, C und D sind Werte gegeben und zwar: $f_A = -4$, $f_B = 4$, $f_C = 8$ bzw. $f_D = 4$.

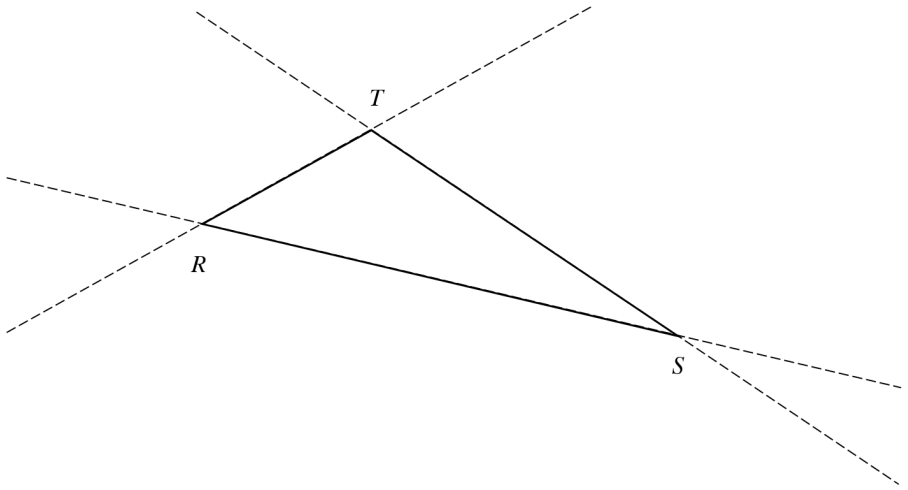
- Bestimmen Sie mittels bilinearer Interpolation im Rechteck den Wert im Punkt Q.

- Bestimmen sie mittels linearer Interpolation im Dreieck $\Delta(D, A, B)$ den Wert an der Stelle Q.

c) Markieren Sie in der nachfolgenden Abbildung den Bereich der Punkte $P = \rho R + \sigma S + \tau T$, deren baryzentrische Koordinaten folgende Bedingung erfüllen:

$$\rho \geq 0 \wedge \sigma < 0 \wedge \tau \geq 0$$

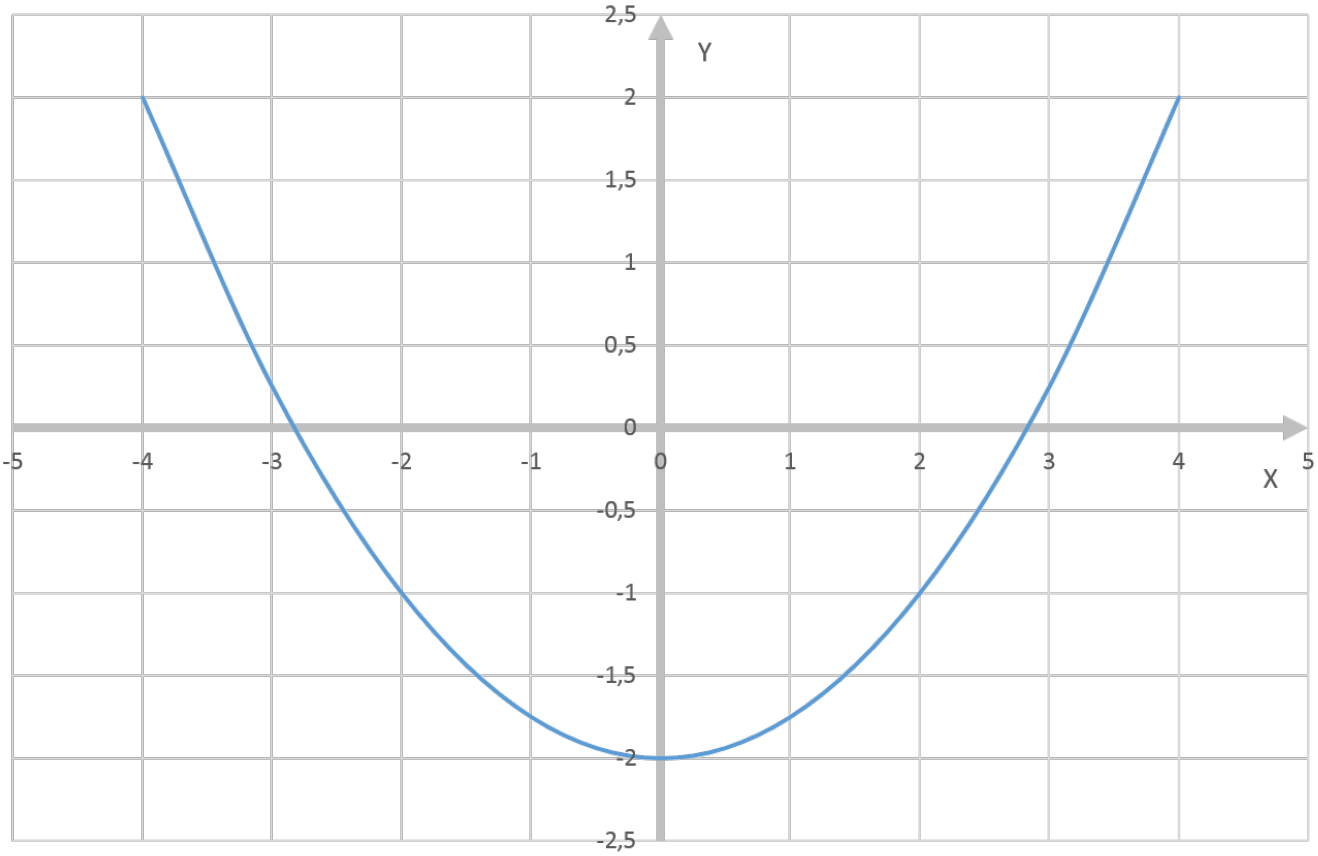
Achten Sie insbesondere auf den Rand dieses Gebietes.



8 Gemischtes (10 Punkte)

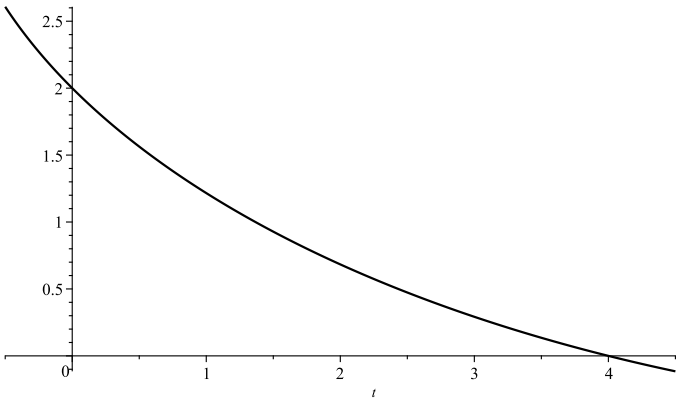
8.1 Nullstellensuche (2 Punkte)

a) Im Folgenden soll mittels Sekantenverfahren eine Nullstelle der unten eingezeichneten Parabel ermittelt werden. Gegeben sind dabei die Startwerte $x_0 = -2$ sowie $x_1 = 4$. Führen Sie **zeichnerisch** zwei Schritte des Sekantenverfahrens durch. Ermitteln Sie also zeichnerisch x_2 und x_3. Achten Sie darauf, dass die Zwischenschritte erkennbar sind.



8.2 Kondition (3 Punkte)

b)



Die oben skizzierte differenzierbare Funktion wird durch einen (nicht näher bekannten) Algorithmus ausgewertet. Es gelte $f(0) = 2$, $f'(0) = -1$ sowie $f(4) = 0$, $f'(4) = -\frac{1}{4}$. Geben Sie an, ob dieses Problem gut oder schlecht konditioniert ist und zwar für den Fall, dass $x \approx 0$:

und für den Fall $x \approx 4$:

c) Bestimmen Sie die Konditionszahlen der folgenden Matrix $\mathbf{A} = \begin{bmatrix} a & 1 \\ 0 & a \end{bmatrix}$, ($a$ ist eine positive Konstante) für eine von Ihnen gewählte Matrixnorm.
Geben Sie auch an, welche Matrixnorm Sie verwenden.

8.3 Nichtlineare Optimierung (5 Punkte)

d) Gegeben ist der Gradient $\nabla F(x, y) = (xy^2 + x + y - 4, x^2y + x + 2y - 8)^T$ der Funktion $F(x, y)$. Diese soll mit einem Abstiegsverfahren minimiert werden. Führen Sie einen Schritt des NEWTON-Verfahrens mit Startwert $(x_0, y_0) = (0, 0)$ durch.

e) Gegeben ist das quadratische Funktional $Q(\vec{x}) = \vec{x}^T \begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 0 \\ 1 & 0 & 1 \end{bmatrix} \vec{x} + [\,-4,\, 16,\, 6\,]\vec{x} - 7$.

Dieses soll mit dem CG-Verfahrens minimiert werden.

Hierfür wurden bereits zwei Schritte mit den Suchrichtungen $\vec{s}_0 = [\,2,\, -1,\, 0\,]^T$ und $\vec{s}_1 = [\,0,\, 1,\, 0\,]^T$ durchgeführt.

Bestimmen Sie die nächste Suchrichtung $\vec{s}_2$.

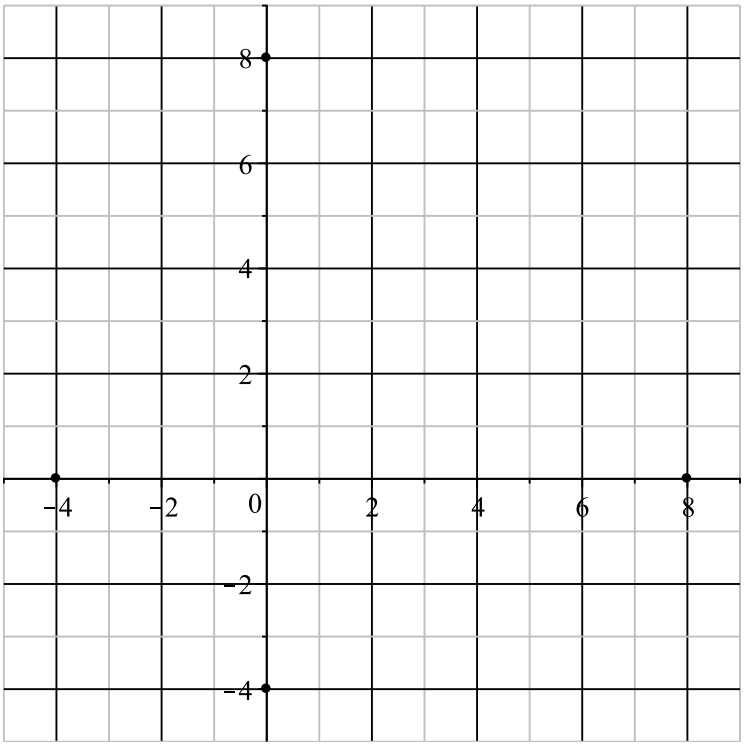
9 Bézier-Kurven (13 Punkte)

a) Betrachten Sie die kubische Bézier-Kurve $C(t) = \sum_{i=0}^{3} \mathbf{b}_i B_i^3(t)$ mit den Kontrollpunkten

$$\mathbf{b}_0 = \begin{bmatrix} -4 \\ 0 \end{bmatrix} \ , \quad \mathbf{b}_1 = \begin{bmatrix} 8 \\ 0 \end{bmatrix} \ , \quad \mathbf{b}_2 = \begin{bmatrix} 0 \\ 8 \end{bmatrix} \ , \quad \mathbf{b}_3 = \begin{bmatrix} 0 \\ -4 \end{bmatrix} \ .$$

Berechnen Sie mit Hilfe des Algorithmus von deCasteljau den Kurvenpunkt $C(\frac{1}{2})$.

b) Skizieren Sie die Bézier-Kurve $C(t)$ aus Teilaufgabe b).



Zur Erinnerung:

$$\mathbf{b}_0 = \begin{bmatrix} -4 \\ 0 \end{bmatrix} \ , \quad \mathbf{b}_1 = \begin{bmatrix} 8 \\ 0 \end{bmatrix} \ , \quad \mathbf{b}_2 = \begin{bmatrix} 0 \\ 8 \end{bmatrix} \ , \quad \mathbf{b}_3 = \begin{bmatrix} 0 \\ -4 \end{bmatrix} \ .$$

c) Die Kurve $C(t)$ aus Teilaufgabe b) wird um den Ursprung um 90° im Uhrzeigersinn rotiert. Bestimmen Sie die Kontrollpunkte der rotierten Kurve.

d) In den folgenden Methoden soll der **Midpoint-Subdivision-Algorithmus** in `C++` implementiert werden.

- Es gibt eine **rekursive Methode** `subdivide(...)` (siehe nächste Seite), die die Subdivision auf den aktuellen Kontrollpunkten `cp` durchführt, solange die aktuelle Rekursionstiefe `curDepth` die maximale Rekursionstiefe `maxDepth` nicht erreicht hat. Der Basisfall der Rekursion ist bereits im Codegerüst vorgegeben: er fügt die aktuellen Kontrollpunkte an den Ergebnisvektor an.
- Der **Einstieg in die Rekursion** soll in der Methode `evalSubdivision(...)` stattfinden.

Entgegen der Übungen ist keine Fehlerbehandlung erforderlich. Duplikate können ignoriert werden.

Hinweis: Gehen Sie davon aus, dass für die Klasse `Point2D` alle arithmetischen Operatoren überladen sind. Sie können alle Methoden der Standard-Template-Library (STL) verwenden, insbesondere `size()` und `push_back()`.

```
vector<Point2D> evalSubdivision(const vector<Point2D> &cp, unsigned int maxDepth) {




}
```

```
void subdivide(vector<Point2D> &res, int maxDepth, int curDepth, const vector<Point2D> &cp){
    if (curDepth >= maxDepth) {
        res.insert(res.end(), cp.begin(), cp.end());
        return;
    }

    unsigned int n = cp.size();        // n: Anzahl der Kontrollpunkte









}
```

10 Faltung (8 Punkte)

a) Die unendliche, periodische Zahlenfolge

$$[\ \cdots, \mathbf{-3}, \mathbf{-2}, \mathbf{-1}, \mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}, -3, -2, -1, 0, 1, 2, 3, -3, -2, -1, 0 \cdots\ ]$$

wird mit dem SOBEL-Filter $[-1, 0, 1]$ gefiltert. Bestimmen Sie das Ergebnis.

b) Ist der 2D-SOBEL-Filter $\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$ separierbar?

c) Gegeben sind die Funktionen

$$g(x) = \begin{cases} 1 & \text{falls} \quad 0 \le x \le 1 \\ 0 & \text{sonst} \end{cases}, \text{ sowie die Hütchenfunktion } h(x) = \begin{cases} 1+x & \text{falls} \quad -1 \le x \le 0 \\ 1-x & \text{falls} \quad 0 \le x \le 1 \\ 0 & \text{sonst} \end{cases}.$$

Beantworten Sie die folgenden Fragen zum Faltungsprodukt $f = g * h$:

- Auf welchem Intervall sind die Funktionswerte $f(x)$ ungleich Null?
- An welcher Stelle x_0 nimmt die Funktion $f(x)$ das Maximum an?
- Welches ist der maximale Funktionswert $f(x_0)$?

d) Die Parabel $p(x) = x^2$ wird mit der Funktion $q(x) = \begin{cases} \frac{1}{2} & \text{falls } |x| \leq 1 \\ 0 & \text{sonst} \end{cases}$ gefaltet.

Berechnen Sie das Ergebnis $[p * q](x)$.