TN
Teilnehmernummer
(von Platzkarte)

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

09. Februar 2016

Algorithmik kontinuierlicher Systeme — 09. Februar 2016

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	11
Max. Punktzahl	7	9	12	6	8	8	13	7	7	6	7
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 bei einem Vertrauensarzt nachgewiesen werden. Melden Sie sich bei der Aufsicht und lassen Sie sich das entsprechende Formular aushändigen.
- Die Bearbeitungszeit beträgt 90 Minuten.
- Überprüfen Sie die Prüfungsaufgaben auf Vollständigkeit (28 Seiten inklusive Deckblatt) und einwandfreies Druckbild.
- Vergessen Sie nicht, auf dem Deckblatt die Angaben zur Person inklusive der Teilnehmernummer (TN) von Ihrer Platzkarte 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, 09. Februar 2016 .....

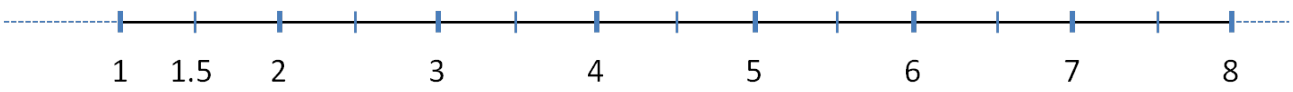
(Unterschrift)

1 Theoriefragen (7 Punkte)

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

Welche Komplexität hat die Matrix-Matrix-Multiplikation **AB** wenn **A** eine vollbesetzte $(n \times n)$-Matrix und **B** eine tridiagonale $(n \times n)$-Matrix 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 LR-Zerlegung $\mathbf{A} = \mathbf{LR}$ gegeben ist?	$\mathcal{O}($ $)$
Welche Konvergenzordnung hat das Gauss-Seidel-Verfahren zum Lösen eines linearen Gleichungssystems mit positiv definiter Koeffizientenmatrix?	
Wie groß ist der Approximationsfehler des stückweise linearen Interpolanten im Falle äquidistanter Schrittweite h ?	$\mathcal{O}($ $)$
Wieviele Kontrollpunkte besitzt eine Bézier-Kurve mit Grad n?	
Welche Konditionszahl besitzt eine Rotationsmatrix?	
Welche Rang besitzt eine invertierbare 3×3 Matrix **A**?	

b) Betrachten Sie $\mathcal{F}_{2,3}$, die Menge der Gleitpunktzahlen zur Basis $B = 2$ mit Mantissenlänge $t = 3$. Markieren Sie in der Abbildung (jeweils durch ein Kreuz) **alle** $a \in \mathcal{F}_{2,3}$ mit $1 \leq a \leq 8$.



2 Lineare Gleichungssysteme (9 Punkte)

2.1 Direkte Verfahren

a) Bestimmen Sie die LR-Zerlegung folgender Matrix $\mathbf{A} = \begin{bmatrix} 2 & -1 & 1 & 0 \\ -4 & 4 & -2 & 2 \\ 4 & -4 & 3 & -4 \\ 0 & 0 & -2 & 6 \end{bmatrix}$.

b) Bestimmen Sie einen Vektor $\vec{w}$, so dass die zugehörige Householder-Spiegelung $\mathbf{H}_{\vec{w}}$ bei Anwendung auf $\mathbf{A}$ (von Teilaufgabe a)) Nullen in der ersten Spalte generiert, d.h. $\mathbf{H}_{\vec{w}} \cdot \mathbf{A} = \begin{bmatrix} * & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \end{bmatrix}$

2.2 Iterative Verfahren

Die Lösung der diskretisierten Poissongleichung

$$4u_{i,j} - u_{i+1,j} - u_{i,j+1} - u_{i-1,j} - u_{i,j-1} = ...$$

in einem Rechteck mit einem äquidistanten Gitter mit Gitterabstand h wird mit iterativen Verfahren bestimmt: mit dem JACOBI-Verfahren (sh. Teilaufgabe c)), mit dem GAUSS-SEIDEL-Verfahren (sh. Teilaufgabe d)) und mit dem SOR-Verfahren (sh. Teilaufgabe e)).

Hinweis: Die Anzahl der Unbekannten (bzw. Gleichungen) ist proportional zu $\frac{1}{h^2}$

c) In der folgenden Tabelle ist zu den Gitterweiten $h = 2^{-3},\ 2^{-4}$ die Anzahl der Iterationen eingetragen, die nötig sind, um die angegebenen Fehlertoleranzen $\tau = 10^{-3},\ 10^{-4}$ mit dem JACOBI-Verfahren einzuhalten.

	$\tau = 10^{-3}$	$\tau = 10^{-4}$	$\tau = 10^{-5}$
$h = 2^{-3}$	154	207	
$h = 2^{-4}$	620	830	
$h = 2^{-5}$			

Ergänzen Sie diese Tabelle und schätzen Sie insbesondere die Anzahl der Iterationen, die notwendig ist, um bei Gitterweite $h = 2^{-5}$ die Fehlertolerenz $\tau = 10^{-5}$ einzuhalten.

d) Schätzen Sie nun für dieses Problem die Anzahl der Iterationen, die nötig sind, wenn man das Sytem mit dem GAUSS-SEIDEL-Verfahren iterativ löst und zwar für die Gitterweiten $h = 2^{-3},\ 2^{-4},\ 2^{-5}$ und Fehlertoleranzen $\tau = 10^{-3},\ 10^{-4},\ 10^{-5}$.

	$\tau = 10^{-3}$	$\tau = 10^{-4}$	$\tau = 10^{-5}$
$h = 2^{-3}$			
$h = 2^{-4}$			
$h = 2^{-5}$			

e) Nun wird das Problem mit dem **SOR**-Verfahren iterativ gelöst und zwar jeweils unter Verwendung des optimalen Relaxationsparameter ω_{opt}.

Ergänzen in der folgenden Tabelle die fehlenden fünf Wert (wiederum jeweils die geschätzte Anzahl der Iterationen, die nötig ist, um bei Gitterweite h die Fehlertoleranz τ einzuhalten).

	$\tau = 10^{-3}$	$\tau = 10^{-4}$	$\tau = 10^{-5}$
$h = 2^{-3}$	13	18	
$h = 2^{-4}$	25	35	
$h = 2^{-5}$			

3 Programmierung: LR-Zerlegung (12 Punkte)

Die Klasse `Solver` stellt grundlegende Funktionen zum Lösen von linearen Gleichungssystemen bereit. Dazu wird die Klasse `Matrix` verwendet. Sie sollen dabei einige Methoden der Klasse `Solver` 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 Solver {
public:
    //! Berechnet eine LR-Zerlegung für die quadratische Matrix A
    static void decomposeA(const Matrix &A, Matrix &L, Matrix &R);

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

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

    //! Loest das lineare System Ax = b mittels LR-Zerlegung
    static void solveSystem(const Matrix &A, const Matrix &b, Matrix &x);

    //! Berechnet die Determinante der Matrix A
    static float calcDeterminant(const Matrix &A);
};
```

Die Methode `decomposeA(const Matrix &A, Matrix &L, Matrix &R)` berechnet für die Matrix $\mathbf{A}$ eine LR-Zerlegung, die die folgende allgemeine Struktur besitzt:

$$\underset{\mathbf{A}}{\begin{bmatrix} \star & \cdots & \star \\ \vdots & \ddots & \vdots \\ \star & \cdots & \star \end{bmatrix}} = \underset{\mathbf{L}}{\begin{bmatrix} \star & & \mathbf{0} \\ \vdots & \ddots & \\ \star & \cdots & \star \end{bmatrix}} \cdot \underset{\mathbf{R}}{\begin{bmatrix} \star & \cdots & \star \\ & \ddots & \vdots \\ \mathbf{0} & & \star \end{bmatrix}}$$

Hinweis: Gehen Sie davon aus, dass sowohl auf der Diagonale von $\mathbf{L}$ als auch $\mathbf{R}$ beliebige Einträge $\neq 0$ stehen.

```
class Matrix {
public:
    //! Konstruktor: Baut eine uninitialisierte Matrix
    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;

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

a) Implementieren Sie die Methode `void forwardSubstitution(const Matrix &L, const Matrix &b, Matrix &y)`, die das Gleichungssystem $\mathbf{L}\vec{y} = \vec{b}$ löst.

Die Methode erhält als Eingabeparameter eine untere Dreiecksmatrix $\mathbf{L}$ und einen Vektor $\vec{b}$, der als $n \times 1$ Matrix dargestellt wird. Die Lösung wird in den Vektor $\vec{y}$ geschrieben, der ebenfalls als $n \times 1$ Matrix dargestellt wird. Sie können davon ausgehen, dass die Ergebnismatrix $\vec{y}$ bereits die richtige Größe hat.

```
void Solver::forwardSubstitution(const Matrix &L, const Matrix &b, Matrix &y) {



}
```

b) Implementieren Sie die Methode `void solveSystem(const Matrix &A, const Matrix &b, Matrix &x)`, die das Gleichungssystem $\mathbf{A}\vec{x} = \vec{b}$ mittels LR-Zerlegung löst.

Die Methode erhält als Eingabeparameter eine quadratische Matrix $\mathbf{A}$ und einen Vektor $\vec{b}$, der als $n \times 1$ Matrix dargestellt wird. Die Lösung wird in den Vektor $\vec{x}$ geschrieben, der ebenfalls als $n \times 1$ Matrix dargestellt wird. Verwenden Sie dazu passende Methoden der Klasse `Solver`.

Sie können davon ausgehen, dass die Matrix $\mathbf{A}$ quadratisch ist und $\vec{x}$ bereits die richtige Größe hat.

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



}
```

c) Implementieren Sie die Methode `float calcDeterminant(const Matrix &A)`, die die Determinante der Matrix **A** berechnet. Nutzen Sie eine LR-Zerlegung aus und verwenden Sie dazu passende Methoden der Klasse `Solver`. Sie können davon ausgehen, dass die Matrix **A** quadratisch ist.

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




}
```

d) Bei dieser Faktorisierung stehen sowohl in der Matrix **L** als auch in der Matrix **R** auf der Diagonale beliebige, von 0 verschiedene Einträge (siehe oben).

Wie müsste die Zerlegung aussehen, um eine möglichst effiziente Speicherung der im Allgemeinen vollbesetzten Dreiecksmatrizen zu ermöglichen? Geben Sie die Struktur von **L** und **R** an und beschreiben Sie kurz die Speicherung.

4 Speicherung dünn besetzter Matrizen (6 Punkte)

Gegeben ist die Matrix $\mathbf{M}$:

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

a) Sie sollen $\mathbf{M}$ im `CRS`-Format abspeichern (Compressed Row Storage). Geben Sie hierzu die interne Datenstruktur an. Die Indizierung beginnt bei 1.

b) Wie kann man aus dem `CRS`-Format die Anzahl der von Null verschiedenen Zeilen, Spalten und Elemente ermitteln?

c) Gegeben ist nun ein Zeilenvektor $\vec{b} = [0, 1, 0, 1, 0]$ und eine Matrix $\mathbf{B}$ im `CCS`-Format (Compressed Column Storage):

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

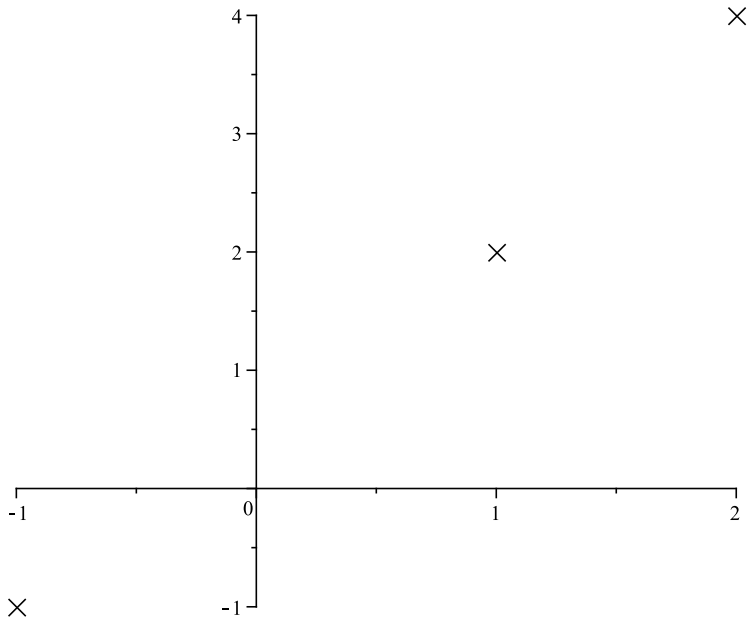
Bestimmen Sie den Zeilenvektor $\vec{b}\mathbf{B}$ ohne die Matrix zu rekonstruieren.
Zwischenschritte müssen erkennbar sein! Die Indizierung beginnt bei 1.

5 Polynominterpolation (8 Punkte)

Gegeben seien folgende Punkte:

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

a) **Skizzieren** Sie in der folgenden Abbildung den nearest neighbor Interpolanten zu den obigen Daten.



b) **Berechnen** Sie die Funktion $l(x) : [-1, 2] \mapsto \mathbb{R}$, welche obige Werte stückweise linear interpoliert.

Zur Erinnerung:

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

Im Weiteren sollen die obigen Daten durch ein Polynom interpoliert werden.

c) Bestimmen Sie die LAGRANGE-Polynome für die obigen Stützstellen $\{-1, 1, 2\}$.

$l_0(x) =$

$l_1(x) =$

$l_2(x) =$

d) Bestimmen Sie des Weiteren die Koeffizienten des Interpolationspolynoms bzgl. der LAGRANGE-Basis.

e) Bestimmen Sie die NEWTON-Polynome für die obigen Stützstellen $\{-1, 1, 2\}$.

$n_0(x) =$

$n_1(x) =$

$n_2(x) =$

f) Bestimmen Sie des Weiteren die Koeffizienten des Interpolationspolynoms bzgl. der NEWTON-Basis.

6 Bézier-Kurven (8 Punkte)

a) Betrachten Sie die Bézier-Kurve $C(t)\,,\ (0 \leq t \leq 1)$ mit den Kontrollpunkten

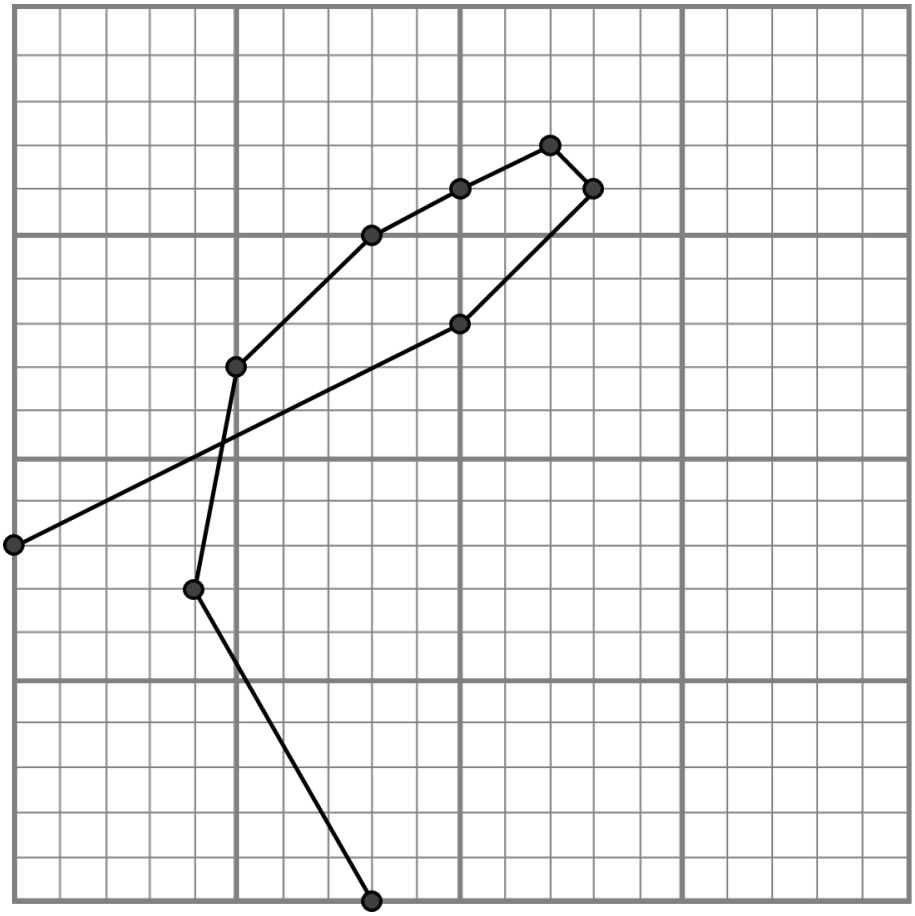
$$\vec{b}_0 = \begin{bmatrix} 8 \\ 16 \end{bmatrix}, \quad \vec{b}_1 = \begin{bmatrix} 0 \\ 8 \end{bmatrix}, \quad \vec{b}_2 = \begin{bmatrix} 8 \\ 0 \end{bmatrix}, \quad \vec{b}_3 = \begin{bmatrix} 32 \\ 56 \end{bmatrix}.$$

Werten Sie die Kurve $C(t)$ an der Stelle $t = 0.25$ aus.

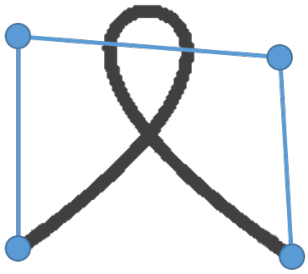
b) Die Bézier-Kurve von Teilaufgabe a) soll zunächst an der Diagonalen $y = x$ gespiegelt werden und anschließend um 2 nach links (entlang der X-Achse) verschoben werden. Bestimmen Sie die Kontrollpunkte der resultierenden Kurve.

Hinweis: Die Spiegelung an der Diagonale wird durch die Abbildung $\begin{bmatrix} x \\ y \end{bmatrix} \mapsto \begin{bmatrix} y \\ x \end{bmatrix}$ *beschrieben.*

c) Für eine BÉZIER-Kurve $B(t)$ wurde ein `midpoint subdivision` Schritt ausgeführt und die Kontrollpolygone der beiden Teilkurven in nachfolgender Abbildung gezeichnet. Bestimmen Sie die Kontrollpunkte $(\vec{c}_0, \vec{c}_1, \vec{c}_2, \vec{c}_3, \vec{c}_4)$ der Kurve $B(t)$ geometrisch und markieren Sie diese **deutlich** mit einem Kreuz.



d) Welche Formeigenschaften von BÉZIER-Kurven sind in nachfolgender Abbildung verletzt?
!!!Achtung: Falsche Antworten geben Punktabzug!!!



e) Man betrachte eine quadratische BÉZIER-Kurve $D(t) = \vec{d}_0 B_0^2(t) + \vec{d}_1 B_1^2(t) + \vec{d}_2 B_2^2(t)$.

Der Mittelpunkt $D(\frac{1}{2})$ der Kurve lässt sich als gewichtete Summe der Kontrollpunkte darstellen:

$$D(\tfrac{1}{2}) = w_0\vec{d}_0 + w_1\vec{d}_1 + w_2\vec{d}_2$$

Bestimmen Sie die Gewichte w_0, w_1 und w_2.

7 Programmierung: Bilineare Interpolation (13 Punkte)

In dieser Aufgabe soll bilineare Interpolation für Bilder in `C++` programmiert werden. Dazu wird die Template-Klasse `Image<T>` verwendet.

Ein `Image<T>` repräsentiert ein Bild der Höhe `height` und Breite `width`. Ein Pixel ist vom Typ `T`.

Im Gegensatz zu den Übungsaufgaben ist **keine** Fehlerbehandlung erforderlich.

Die Klasse `Image<T>` hat folgende Struktur:

```
template<class T>
class Image {
public:
        const unsigned int height;  //! Hoehe
        const unsigned int width;   //! Breite

        //! Konstruktor, initialisiert Attribute und legt Speicher an
        Image(unsigned int height, unsigned int width);

        //! Operator zum Lesezugriff an Stelle (i, j)
        const T &operator()(unsigned int i, unsigned int j) const;

        //! Operator zum Schreibzugriff an Stelle (i, j)
        T &operator()(unsigned int i, unsigned int j);

        //! Skaliert das Bild um Faktor hoch und fuellt die Zwischenraeume
        Image<T> Upsample() const;

        //! Interpoliert die umliegenden Pixel für u, v aus [0, 1]
        T GetInterpolatedPixel(float u, float v) const;

private:
        T *data;    //! Speicher
};
```

Die folgende Abbildung veranschaulicht das Speicherlayout des Bildes in der Variable `data`:

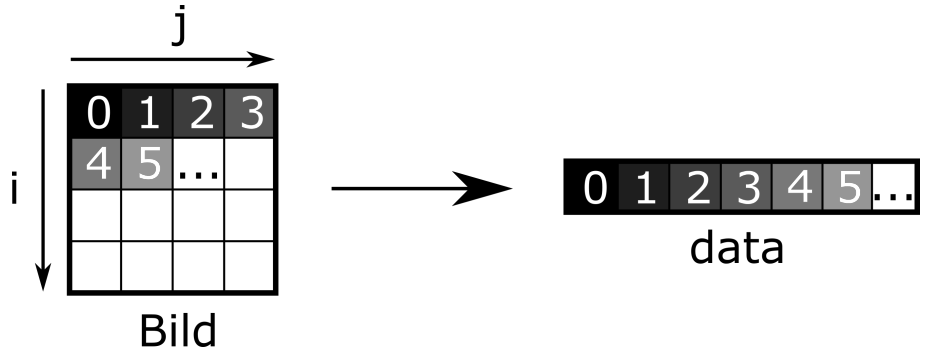


Abbildung 1: Layout eines 4×4-Bildes

Hinweis: Verändern Sie die Klassenstrukturen nicht, d.h. führen Sie keine neuen Attribute oder Methoden ein.

a) Implementieren Sie zunächst den Operator für den Lesezugriff `const T &operator()(unsigned int i, unsigned int j) const`, der eine (konstante) Referenz auf das Bild an der Stelle (i, j) zurückgibt. Beachten Sie das Speicherlayout aus Abbildung 1!

```
const T &Image::operator()(unsigned int i, unsigned int j) const {


}
```

b) Implementieren Sie den Operator für den Schreibzugriff `T &operator()(unsigned int i, unsigned int j)`, welcher eine Referenz auf das Bild an der Stelle (i, j) zurückliefert. Beachten Sie das Speicherlayout aus Abbildung 1!

```
T &Image::operator()(unsigned int i, unsigned int j) {


}
```

c) Die Methode `T getInterpolatedPixel(float u, float v) const` erhält zwei Koordinaten $(u, v) \in [0, 1]^2$ und berechnet für diese Position den bilinearen Interpolanten aus den vier umliegenden Pixeln. Dabei entspricht die Koordinate $(u, v) = (0, 0)$ dem Pixel $(0, 0)$ und die Koordinate $(u, v) = (1, 1)$ dem Pixel $(\text{height} - 1, \text{width} - 1)$.

Implementieren Sie die Methode, indem Sie sich zunächst für die übergebene Koordinate die zugehörige ganzzahlige Pixelposition berechnen. Rufen Sie dann die Werte der vier relevanten Pixel ab. Interpolieren Sie diese Pixel, indem Sie sich für jeden der Pixel die korrekten Interpolationsgewichte $\in [0, 1]$ bestimmen.

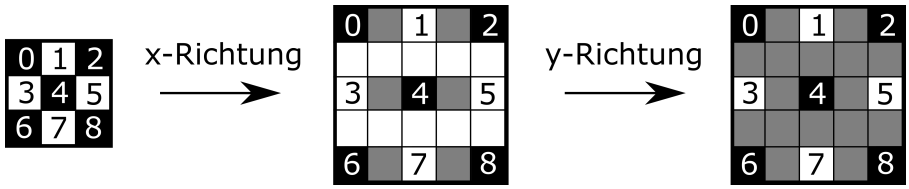
```
T Image::GetInterpolatedPixel(float u, float v) const {
    const Image<T> &image = *this;









}
```

d) Die Methode `Image<T> Upsample() const` soll das Bild auf die Dimensionen $(2 \cdot \text{height} - 1) \times (2 \cdot \text{width} - 1)$ hochskalieren und dabei die Werte in den Zwischenräumen durch bilineare Interpolation bestimmen.

Implementieren Sie diese Methode, indem Sie zunächst in x-Richtung das Ergebnisbild `result` entsprechend füllen. Führen Sie die Operation danach in y-Richtung durch.

Greifen Sie bei Ihrer Implementierung nicht auf die Methode `getInterpolatedPixel()` zurück.

Das folgende Bild veranschaulicht die Interpolation für ein 3×3 Bild:

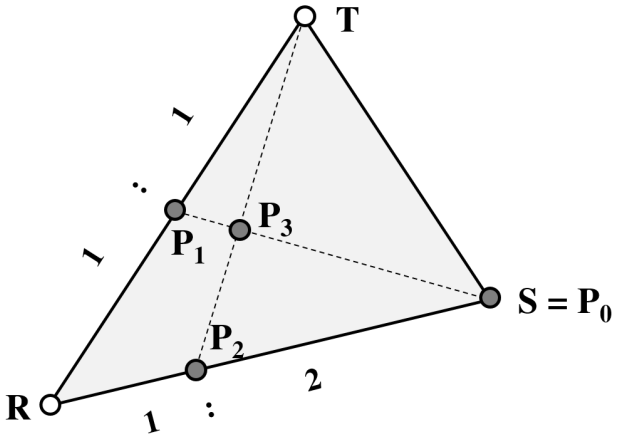


Tipp: Die Gewichtung der benachbarten Pixel ist hier ein konstanter Wert.

```
template<class T>
Image<T> Image<T>::Upsample() const {
    const Image<T> &image = *this;
    Image<T> result(2 * height - 1, 2 * width - 1);
```

```
    return result;
}
```

8 Baryzentrische Koordinaten (7 Punkte)



a) Bestimmen Sie die baryzentrischen Koordinaten $[\rho_i, \sigma_i, \tau_i]^T$ der vier Punkte $\mathbf{P}_i$, $i = 0, 1, 2, 3$, bzgl. des Dreiecks $\Delta(\mathbf{R}, \mathbf{S}, \mathbf{T})$.

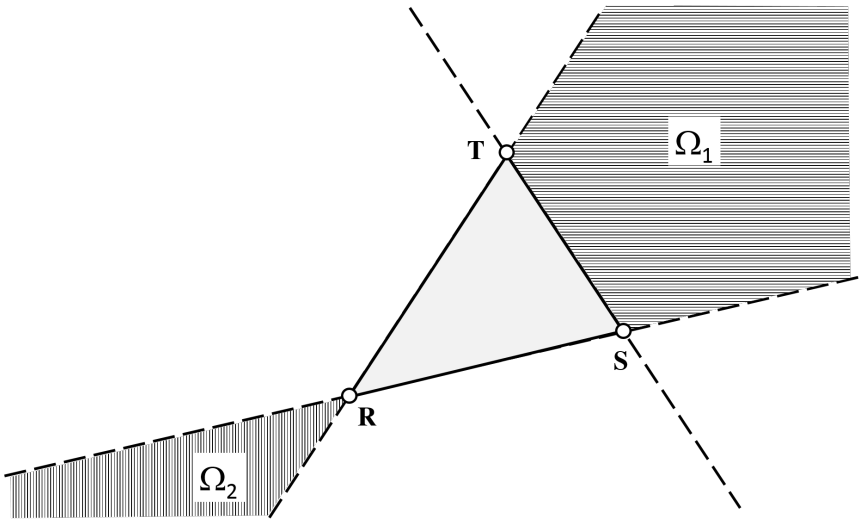
Beachten Sie dabei die in der Abbildung angegebenen Teilungsverhältnisse!

b) Betrachten Sie die Punkte $\mathbf{A} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$, $\mathbf{B} = \begin{bmatrix} 3 \\ 0 \end{bmatrix}$, $\mathbf{C} = \begin{bmatrix} 2 \\ 3 \end{bmatrix}$ sowie den Punkt $\mathbf{Q} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$.

Berechnen Sie die baryzentrischen Koordinaten des Punktes $\mathbf{Q}$ bzgl. des Dreiecks $\Delta(\mathbf{A}, \mathbf{B}, \mathbf{C})$.

c) In den Punkten $\mathbf{A}, \mathbf{B}, \mathbf{C}$ sind Werte gegeben: $f_A = -3$, $f_B = 1$, $f_C = 3$.
Bestimmen Sie mittels linearer Interpolation den Wert im Punkt $\mathbf{U} = \frac{1}{4}\mathbf{A} + \frac{1}{2}\mathbf{B} + \frac{1}{4}\mathbf{C}$.

d)



- Wie kann man mit Hilfe der baryzentrischen Koordinaten (bzgl. des Dreiecks $\Delta(\mathbf{R}, \mathbf{S}, \mathbf{T})$) das horizontal schraffierte Gebiet Ω_1 charakterisieren?

- Wie kann man mit Hilfe der baryzentrischen Koordinaten (bzgl. des Dreiecks $\Delta(\mathbf{R}, \mathbf{S}, \mathbf{T})$) das vertikal schraffierte Gebiet Ω_2 charakterisieren?

9 Faltung (7 Punkte)

a) Die unendliche, periodische Zahlenfolge

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

wird mit der Filtermaske $\frac{1}{3} \cdot [\,1, 1, 1\,]$ gefiltert. Bestimmen Sie das Ergebnis.

b) Eine Funktion f soll nacheinander mit h_1 und h_2 gefaltet werden.

1. Kommt es dabei auf die Reihenfolge an?

2. Die zweimalige Faltung $f \mapsto f * h_1 \mapsto (f * h_1) * h_2$ kann man durch eine Faltung erreichen. Mit welcher Funktion h gelingt das?

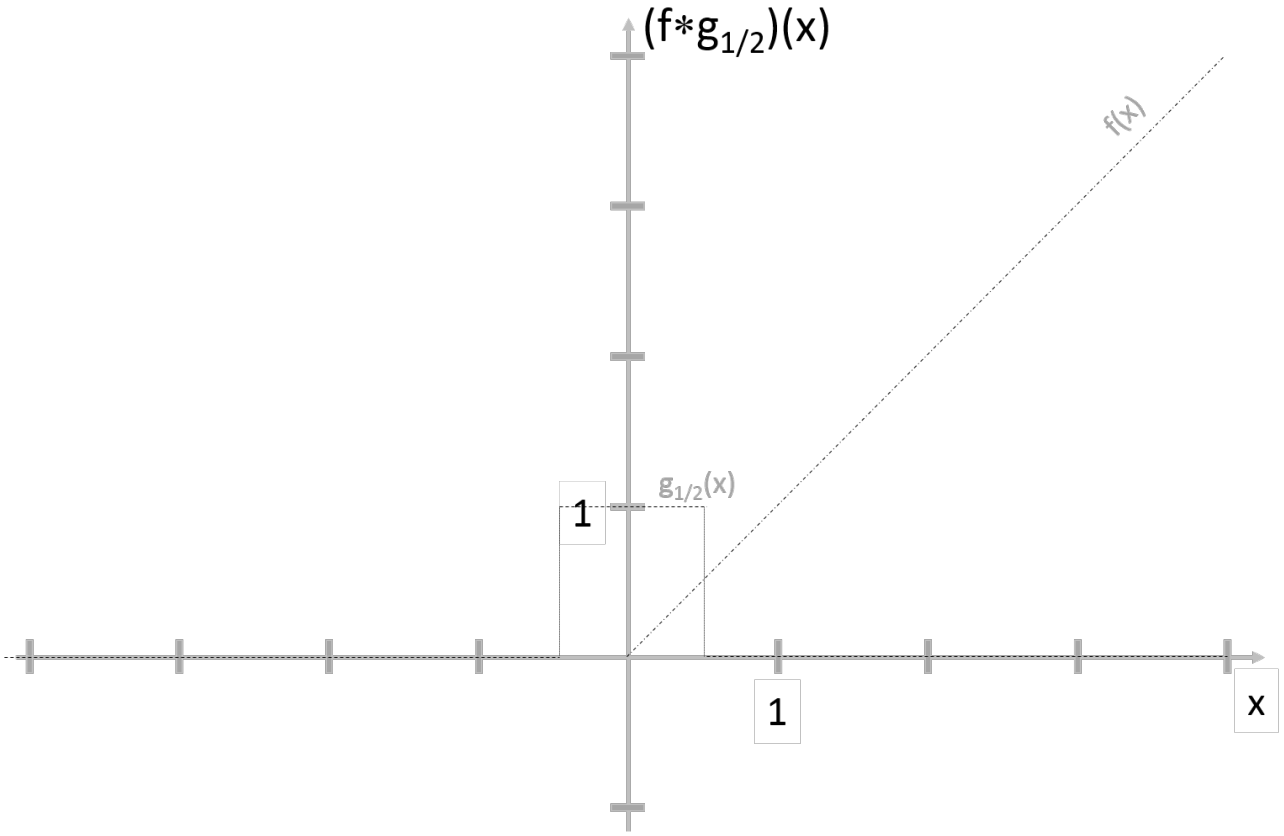
c) Was geschieht wenn man eine Funktion f mit der Dirac-"Funktion" δ_a faltet?

$$(f * \delta_a)(x) =$$

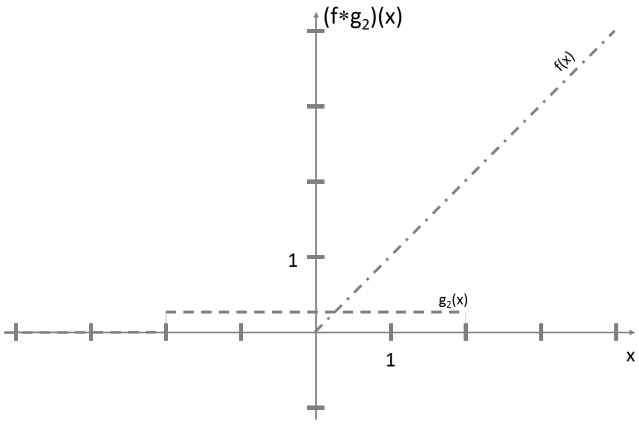
Gegeben ist die Funktion

$$f(x) := \begin{cases} 0 & \text{falls} \quad x \leq 0 \\ x & \text{falls} \quad x \geq 0 \end{cases} \quad , \quad \text{sowie für } a > 0 \text{ die Funktionen} \quad g_a(x) := \begin{cases} \frac{1}{2a} & \text{falls} \quad |x| \leq a \\ 0 & \text{sonst} \end{cases}$$

d) **Skizzieren** Sie die Funktionen $f * g_a$ für $a = \frac{1}{2}$



e) **Berechnen** Sie die Funktionen $f * g_2 \quad (a = 2)$



10 Hauptkomponentenanalyse (6 Punkte)

Gegeben seien die folgenden 2D-Datenpunkte $\vec{p}_i = \left[x_i, y_i\right]^T$:

$$\vec{p}_0 = \begin{bmatrix} 4 \\ 0 \end{bmatrix},\ \vec{p}_1 = \begin{bmatrix} -1 \\ -1 \end{bmatrix},\ \vec{p}_2 = \begin{bmatrix} 1 \\ 2 \end{bmatrix},\ \vec{p}_3 = \begin{bmatrix} -1 \\ 2 \end{bmatrix},\ \vec{p}_4 = \begin{bmatrix} 2 \\ 2 \end{bmatrix}\ .$$

a) Bestimmen Sie die Kovarianzmatrix $\mathbf{C}$ die zu den $\vec{p}_i$ gehört.

b) Gegeben sei die folgende Kovarianzmatrix:

$$\mathbf{B} = \begin{bmatrix} 4 & -3 \\ -3 & 4 \end{bmatrix}.$$

Bestimmen Sie nun die zu $\mathbf{B}$ gehörenden Hauptachsen.

11 Numerische Integration (7 Punkte)

Gegeben ist die Funktion $f(x) = 64x^3$. Das Integral $\int_0^1 f(x)\,dx$ soll mit Hilfe von numerischer Integration näherungsweise berechnet werden. Dabei sollen verschiedene Verfahren verwendet werden.

a) Verwenden Sie die iterierte Trapezregel für die Partition $\{0, \frac{1}{2}, 1\}$.

b) Verwenden Sie die iterierte Trapezregel für die Partition $\{0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1\}$.

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

d) Verwenden Sie die SIMPSON-Regel auf dem Intervall $[0, 1]$

e) Was ist Fehlerordnung der iterierten Trapezregel in Abhängigkeit der Schrittweite h?

f) Was ist Fehlerordnung der iterierten SIMPSON-Regel in Abhängigkeit der Schrittweite h?