Allgemeine Informationen: Dieses Aufgabenblatt enthält schriftliche und/oder Programmieraufgaben. Bitte kombinieren Sie alle Lösungen zu den schriftlichen Aufgaben zu einem einzelnen PDF Dokument, welches Sie nach folgendem Schema benennen:`{lastname}-written.pdf`. Sie können Ihre Lösungen auch scannen oder fotografieren. Achten Sie in diesem Fall auf die Lesbarkeit. Es werden JPEG/PNG Bilddateien akzeptiert welche wie folgt benannt werden müssen:`{exercisenumber}-{lastname}-written.{jpeg/png}`. Stellen Sie sicher, dass alle Rechenschritte nachvollziehbar sind und kombinieren Sie nicht zu viele kleine Schritte zu einem einzelnen. Die Programmieraufgaben müssen in *Julia* gelöst sein und Ihr Quellcode sollte nach folgendem Schema benannt sein: `{exercisenumber}-{lastname}.jl`.

(1) (2.5 Punkte) Gegeben ist das folgende System linearer Gleichungen:

$$\begin{array}{rrrrr} 16x & -4y & -4z & & = 32 \\ -4x & 17y & z & -4w & = 0 \\ -4x & y & 17z & -4w & = -4 \\ & -4y & -4z & 18w & = -3 \end{array}$$

Dieses kann in Matrixform d.h. $\mathbf{Ax} = \mathbf{b}$ wie folgt geschrieben werden:

$$\begin{bmatrix} 16 & -4 & -4 & 0 \\ -4 & 17 & 1 & -4 \\ -4 & 1 & 17 & -4 \\ 0 & -4 & -4 & 18 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = \begin{bmatrix} 32 \\ 0 \\ -4 \\ -3 \end{bmatrix}.$$

a) (1 Punkt) $\mathbf{A}$ ist positiv-definit. Kann die *Cholesky-Zerlegung* für $\mathbf{A}$ angewendet werden und warum bzw. warum nicht? Falls ja, berechnen Sie diese per Hand. Falls die Zerlegung nicht angewandt werden kann berechnen Sie stattdessen die *LU-Zerlegung* von $\mathbf{A}$ per Hand. Arbeiten Sie in beiden Fällen mit Brüchen und Wurzeln nicht mit Dezimalzahlen. Erklären Sie die Methode welche nicht verwendet wird. Was ist der Unterscheid der beiden?

b) (1 Punkt) Lösen Sie das Gleichungssystem durch Vor- bzw. Rückwärtseinsetzen unter Verwendung der Ergebnisse der Cholesky- oder der LU-Zerlegung (siehe 1.a). Arbeiten Sie mit Brüchen und Wurzeln nicht mit Dezimalzahlen. Wie unterscheidet sich das Vor- und Rückwärtseinsetzen zwischen den beiden Zerlegungen?

c) (0.5 Punkte) Berechnen Sie per Hand eine Iteration des *Gauß-Seidel*-Verfahrens.
`Hinweis:` Starten Sie mit dem Nullvektor als Initialisierung.

(2) (2.5 Punkte) Das Ziel dieser Aufgabe ist es die Temperaturverteilung in einem quadratischen Raum zu berechnen. Implementieren Sie hierzu verschiedene Algorithmen zum Lösen linearer Gleichungssysteme.

Als Ausgangspunkt betrachten wir die stationäre Wärmeleitungsgleichung in 2D (*Laplace Gleichung*), welche wir unter Berücksichtigung gewisser Randwerte (z.B.: Temperatur der Wände, Fenster, Heizkörper und Türen) lösen wollen:

$$\nabla^2 u(x,y) = \frac{\partial^2 u(x,y)}{\partial x^2} + \frac{\partial^2 u(x,y)}{\partial y^2} = 0,$$

mit u als *Temperaturfunktion*. Um diese elliptische partielle Differentialgleichung zu lösen, wird der Raum durch eine Menge an 2D-Punkten diskretisiert, welche in einem Gitter ausgerichtet sind (gleichmäßige Abstände h).

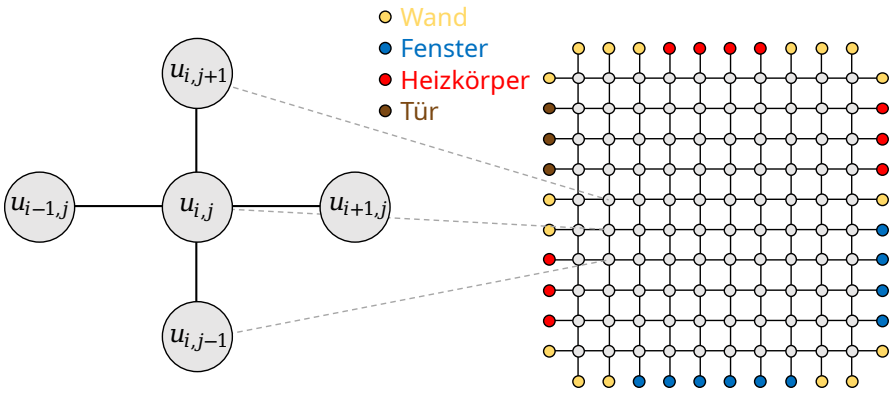

Abbildung 1: Setup für welches die Temperaturverteilung simuliert werden soll.

Mit Hilfe der Zentralen Differenz zweiter Ordnung, können wir nun die Raumableitung approximieren:

$$\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{h^2} = 0$$
$$\frac{u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j}}{h^2} = 0.$$

Wir erhalten somit ein lineare Gleichungssystem für die unbekannten Temperaturen an den Gitterpunkten, welches wir mit Hilfe der bekannten Randwerte lösen können.

a) (0.5 Punkte) In Abbildung 2 ist ein kleines Beispielgitter abgebildet. Geben Sie das zu lösende Gleichungssystem, für die unbekannten Temperaturen $\mathbf{u} = (u_{1,1}, u_{2,1}, u_{1,2}, u_{2,2})^\intercal$ in Matrixform ($\mathbf{Au} = \mathbf{b}$) an.

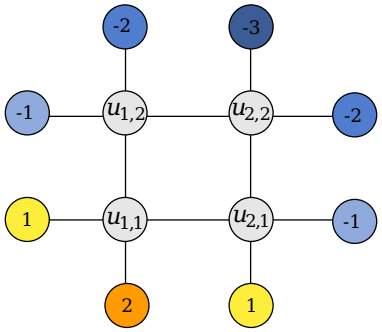

Abbildung 2: Kleines Beispiel Gitter mit den Temperaturen am Rand (orange/blaue Punkte) gegeben.

b) (1 Punkt) Implementieren Sie die Iterationsschritte für das *Jacobi*-Verfahren und das *SOR*-Verfahren (*Successive Over-Relaxation*). Verwenden Sie dazu das Julia Template **2-iterative_solvers.jl**, welches bereits die Implementation des *Gauß-Seidel*-Verfahrens beinhaltet. Nachdem alle Tests (siehe Code-Block am Ende von **2-iterave_solvers.jl**) korrekt ausgeführt werden, lösen Sie mit Hilfe Ihrer Implementierung die diskretisierte Laplace Gleichung. Modifizieren Sie dafür das Julia Template **2-steady_state_heat.jl**. Abbildung 3 zeigt das Ergebnis der Simulation für $N = 100$.
`Hinweis:` Sollten Sie Probleme mit der Ausführungsdauer des Programms haben, können Sie N verkleinern.

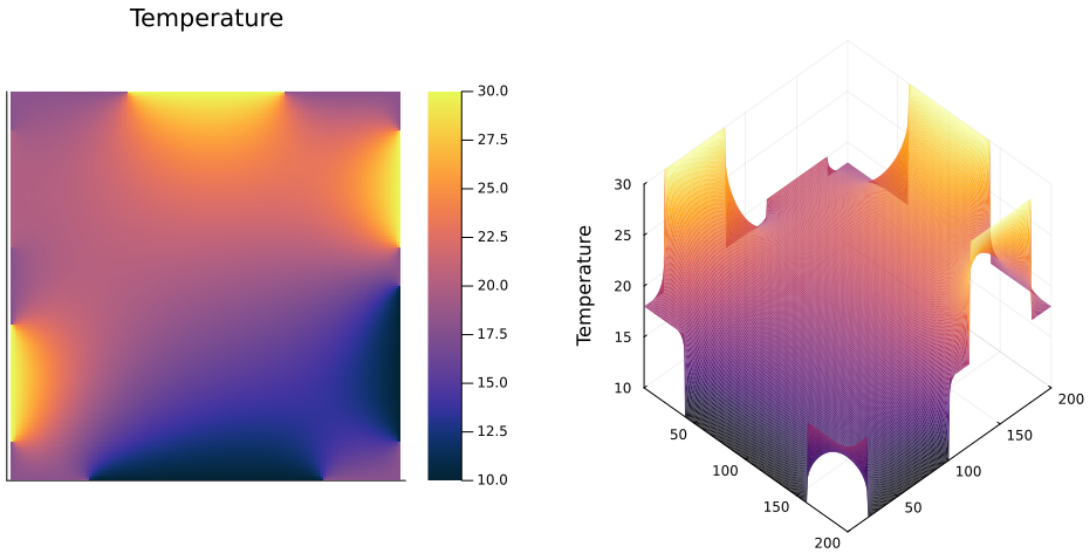

Abbildung 3: Stationäre Temperaturverteilung für das Setup definiert in Abbildung 1, berechnet auf einem Grid mit $N = 100$.

c) (1 Punkt) Implementieren Sie die Berechnung der Euklidischen Norm des Residuums um die Konvergenzgeschwindigkeit der iterativen Löser zu verfolgen:

$$||\mathbf{b} - \mathbf{A}\mathbf{x}^{(\mathbf{k})}||,$$

wobei $\mathbf{x}^{(\mathbf{k})}$ der aktuelle Lösungsvektor ist – siehe Funktion **errorNorm**.
Lösen Sie die stationäre Wärmeleitungsgleichung (mit unterschiedlichen Problemgrößen N) und vergleichen Sie die Konvergenzgeschwindigkeit aller Verfahren. Erweitern Sie hierfür den Konvergenz-Plot um die Konvergenz des Jacobi und des SOR Verfahrens. Speichern Sie die Plots als PNGs ab und legen Sie diese ihrer Abgabe bei (verwenden Sie die Funktion **savefig**).
Welche Methode konvergiert am schnellsten? Was ist der Grund für die verschiedenen Konvergenzzeiten? Antworten Sie in schriftlicher Form.

(3) (2 Punkte) Gegeben ist eine parabolische partielle Differentialgleichung, nämlich die 1D Diffusionsgleichung mit einem konstanten Diffusionskoeffizienten α.

$$\frac{\partial u(x,t)}{\partial t} = \alpha \frac{\partial^2 u(x,t)}{\partial x^2}, \qquad -1 \leq x \leq 1, \quad 0 \leq t \leq 1$$

Des Weiteren, sind homogene *Dirichlet*-Randbedingungen $u(-1,t) = u(1,t) = 1$, und die anfängliche Wärmeverteilung $u(x,0) = x^2 - |x|$ gegeben. Um ein solches Problem numerisch zu lösen, diskretisieren wir das stetige Interval $[-1,1]$ durch Punktwerte, d.h. $u(ih,t) \approx u(x_i,t),\ i \in \{0,...,I\}$ wobei I die Anzahl an Punkten und $h = 2/I$ deren Abstand zueinander beschreiben. Beachten Sie, dass aufgrund der Randbedingungen $u(x_0,t) = u(x_I,t) = 1$ ist.
Analog zum letzten Aufgabenblatt, approximieren wir die rechte Seite mit der Zentralen Differenz zweiter Ordnung, und die zeitliche Ableitung mit der Vorwärts Differenz:

$$\frac{u(x_i,t_{n+1}) - u(x_i,t_n)}{\tau} = \alpha \frac{u(x_{i+1},t_n) - 2u(x_i,t_n) + u(x_{i-1},t_n)}{h^2}.$$

Nachdem wir für $u(x_i,t_{n+1})$ lösen bekommen wir das *Explizite Euler*-Verfahren:

$$u(x_i,t_{n+1}) = u(x_i,t_n) + \tau\alpha \frac{u(x_{i+1},t_n) - 2u(x_i,t_n) + u(x_{i-1},t_n)}{h^2}. \tag{1}$$

Eine Implementierung dieses Verfahrens wird Ihnen im Julia Template **3-diffusion.jl** zur Verfügung gestellt. Abbildung 4 zeigt das Ergebnis davon.

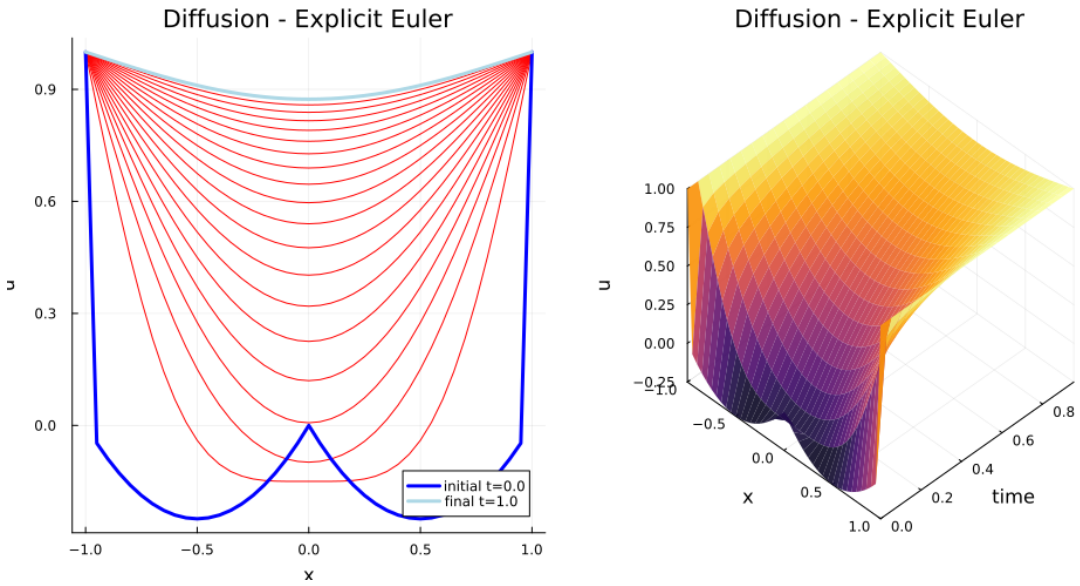

Abbildung 4: Ergebnis der Diffusionssimulation mit dem Expliziten Euler-Verfahren für $N = 1000,\ I = 40,\ \alpha = 1$.

Ein Problem des *Expliziten Euler*-Verfahrens ist das instabile Verhalten bei größeren Zeitschritten τ (Sie können dies selber beobachten mit $N < 800$). Das Ziel der Aufgabe ist die Herleitung und Implementierung des stabileren *Impliziten Euler*-Verfahrens für dieses Problem. Allerdings, ist dabei das Lösen eines linearen Gleichungssystems für jeden Zeitschritt notwendig um die neuen Werte $u(x_i, t_n),\ i \in \{1, ..., I-1\}$ zu bestimmen.

a) (0.5 Punkte) Verwenden Sie die *Rückwärts Differenz* um die zeitliche Ableitung der Diffusionsgleichung zu approximieren. Als Nächstes, isolieren Sie die Unbekannten, $u(x_{i-1}, t_n), u(x_i, t_n), u(x_{i+1}, t_n)$, wobei $u(x_i, t_{n-1})$ vom vohergegangenen Zeitschritt bekannt ist.

b) (0.5 Punkte) Gegeben ist nun die räumliche Diskretisierung mit $I = 5$ und den Randbedingungen $u(x_0, t) = u(x_5, t) = 1$. Des Weiteren sind die Werte des vohergegangenen Zeitschrittes $u(x_1, t_{n-1}), ..., u(x_4, t_{n-1})$ bekannt.
Geben Sie das zu lösende Gleichungssystem in Matrixform ($\mathbf{Au} = \mathbf{b}$), mit den Unbekannten $\mathbf{u} = (u(x_1, t_n), ..., u(x_4, t_n))^T$ an.
`Hinweis:` Die Matrix $\mathbf{A}$ wird entlang der Diagonale die selben Einträge aufweisen.

c) (1 Punkt) Implementieren Sie das *Implizite Euler*-Verfahren im Template **diffusion.jl**. Sie müssen dafür die Systemmatrix $\mathbf{A}$ konstruieren und in jedem Zeitschritt den Ergebnisvektor richtig initialisieren und das Gleichungssystem lösen um die Unbekannten $\mathbf{u}$ zu finden (Sie können dafür den Backslash \ Operator verwenden).
`Hinweis:` Die Anzahl der Unbekannten ist $I - 2$ da die Werte am Rand bekannt sind.