Programmierblatt 4.

Abgabewoche: **09.12–13.12.2024**

Nachdem wir uns bisher mit dem stationären, elliptischen Poisson-Problem befasst haben, möchten wir uns auf dem letzten Programmierblatt der Lösung von zeitabhängigen Problemen widmen. Exemplarisch betrachten wir dazu ein hyperbolisches Problem, die *Wellengleichung*.

Wellengleichung

Auf einem Gebiet $\Omega \subset \mathbb{R}^2$ und einem Intervall $[0,T]$ mit $T < \infty$ betrachten wir die Gleichung

$$\frac{\partial^2}{\partial t^2} u(\mathbf{x},t) - \Delta_{\mathbf{x}}\, u(\mathbf{x},t) = 0 \qquad \text{für } (\mathbf{x},t) \in \Omega \times (0,T].$$

Dabei agiert der $\Delta_{\mathbf{x}}$-Operator nur auf der Ortsvariablen $\mathbf{x}$. Die Lösung $u(\mathbf{x},t)$ beschreibt dann eine Grösse, welche Wellenphänomene beschreibt. Zur eindeutigen Lösbarkeit der Gleichung müssen entsprechende Anfangs- und Randbedingungen vorgegeben werden. Der Einfachheit halber beschränken wir uns auf homogene Dirichlet- und homogene Neumann-Randbedingungen.[1] Das komplette System von Gleichungen, dass wir betrachten, lautet also:

$$\begin{aligned}
\frac{\partial^2}{\partial t^2} u(\mathbf{x},t) - \Delta_{\mathbf{x}}\, u(\mathbf{x},t) &= 0 && \text{für } (\mathbf{x},t) \in \Omega \times (0,T],\\
u(\mathbf{x},t) &= 0 && \text{für } (\mathbf{x},t) \in \Gamma_D \times (0,T],\\
\langle \nabla_{\mathbf{x}}\, u(\mathbf{x},t), \mathbf{n}(\mathbf{x}) \rangle &= 0 && \text{für } (\mathbf{x},t) \in \Gamma_N \times (0,T],\\
u(\mathbf{x},0) &= u_0(\mathbf{x}) && \text{für } \mathbf{x} \in \Omega,\\
\frac{\partial}{\partial t} u(\mathbf{x},0) &= v_0(\mathbf{x}) && \text{für } \mathbf{x} \in \Omega.
\end{aligned}$$

Ortsdiskretisierung

Betrachtet man die Wellengleichung für einen fixierten, aber beliebigen Zeitpunkt $t \in (0,T]$, so kann man die (örtlich) schwache Formulierung bilden und davon die Galerkin-Approximation für einen vorgegebenen Finite-Elemente-Raum V_h des Gebiets Ω bilden. Die zeitabhängigen Koeffizienten der Finite-Element-Approximation erfüllen dann eine gewöhnliche Differentialgleichung zweiter Ordnung. Zusammen mit den Anfangswerten ergibt dies dann folgendes Anfangswertproblem:

$$\begin{aligned}
\mathbf{M}_h \frac{\partial^2}{\partial t^2} \mathbf{u}(t) + \mathbf{A}_h \mathbf{u}(t) &= \mathbf{0} && \text{für } t \in (0,T],\\
\mathbf{u}(0) &= \Pi_h\, u_0,\\
\frac{\partial}{\partial t} \mathbf{u}(0) &= \Pi_h\, v_0.
\end{aligned}$$

Hierbei ist $\mathbf{M}_h = [\langle \phi_i, \phi_j \rangle_{L^2(\Omega)}]_{i,j}$ die Massenmatrix und $\mathbf{A}_h = [\langle \nabla\phi_i, \nabla\phi_j \rangle_{L^2(\Omega)}]_{i,j}$ die Steifgkeitsmatrix, während der Operator Π_h die Anfangswerte in den Finite Elemente-Raum V_h abbildet (z.B. per Projektion oder Interpolation) und deren Koeffizienten zurück gibt.

[1] Akustisch entsprechen die homogenen Dirichlet-Daten einer schallweichen Randbedingung (offenes Fenster) während die homogenen Neumann-Daten eine schallharte Randbedingung (Wand) darstellen.

Zeitdiskretisierung

Für die Diskretisierung des Anfangswertsystems benutzen wir Finite Differenzen. Für $N \in \mathbb{N}$ sei hierfür $\{t_0, t_1, \dots, t_N\}$ ein regelmässiges Gitter für das Zeitintervall $[0, T]$, das heisst $t_0 = 0$, $t_N = T$ und $t_i = i\delta$ mit $\delta = T/N$. Wir setzen $\mathbf{u}^{(i)} \approx \mathbf{u}(t_i)$ für $i = 0, \dots, N$. Mit dem Ersetzen der zweiten Ableitung durch die zentrale Differenz zweiter Ordnung,

$$\frac{\partial^2}{\partial t^2}\mathbf{u}(t_i) \approx \frac{\mathbf{u}(t_{i+1}) - 2\mathbf{u}(t_i) + \mathbf{u}(t_{i-1})}{\delta^2},$$

erreicht man die Gleichung des klassischen *leapfrog-Verfahrens*,

$$\mathbf{M}_h\mathbf{u}^{(i+1)} - (2\mathbf{M}_h - \delta^2\mathbf{A}_h)\mathbf{u}^{(i)} + \mathbf{M}_h\mathbf{u}^{(i-1)} = \mathbf{0}.$$

Benutzt man stattdessen die Vorwärts- und Rückwärts-Differenzen zweiter Ordnung,

$$\frac{\partial^2}{\partial t^2}\mathbf{u}(t_{i-1}) \approx \frac{\mathbf{u}(t_{i+1}) - 2\mathbf{u}(t_i) + \mathbf{u}(t_{i-1})}{\delta^2} \quad \text{und} \quad \frac{\partial^2}{\partial t^2}\mathbf{u}(t_{i+1}) \approx \frac{\mathbf{u}(t_{i+1}) - 2\mathbf{u}(t_i) + \mathbf{u}(t_{i-1})}{\delta^2},$$

so erhält man die verwandten Gleichungen

$$\mathbf{M}_h\mathbf{u}^{(i+1)} - 2\mathbf{M}_h\mathbf{u}^{(i)} + (\mathbf{M}_h + \delta^2\mathbf{A}_h)\mathbf{u}^{(i-1)} = \mathbf{0}$$

und

$$(\mathbf{M}_h + \delta^2\mathbf{A}_h)\mathbf{u}^{(i+1)} - 2\mathbf{M}_h\mathbf{u}^{(i)} + \mathbf{M}_h\mathbf{u}^{(i-1)} = \mathbf{0}.$$

Um die Symmetrie der Gleichungen um den Punkt i zu erhalten, addieren wir sie und kommen auf die Gleichung

$$\left(\mathbf{M}_h + \tfrac{1}{2}\delta^2\mathbf{A}_h\right)\mathbf{u}^{(i+1)} - 2\mathbf{M}_h\mathbf{u}^{(i)} + \left(\mathbf{M}_h + \tfrac{1}{2}\delta^2\mathbf{A}_h\right)\mathbf{u}^{(i-1)} = \mathbf{0},$$

also eine Art von impliziter Modifikation des klassischen leapfrog-Verfahrens.

Durch bilden der Konvexkombination der Gleichung des leapfrog-Verfahrens mit der der impliziten Modifikation erreichen wir schliesslich für $\theta \in [0, 1]$ die Gleichung

$$\left(\mathbf{M}_h + \tfrac{\theta}{2}\delta^2\mathbf{A}_h\right)\mathbf{u}^{(i+1)} - \left(2\mathbf{M}_h - (1-\theta)\delta^2\mathbf{A}_h\right)\mathbf{u}^{(i)} + \left(\mathbf{M}_h + \tfrac{\theta}{2}\delta^2\mathbf{A}_h\right)\mathbf{u}^{(i-1)} = \mathbf{0}.$$

Aufgelöst nach $\mathbf{u}^{(i+1)}$ ergibt sich also

$$\mathbf{u}^{(i+1)} = \left(\mathbf{M}_h + \tfrac{\theta}{2}\delta^2\mathbf{A}_h\right)^{-1}\left(\left(2\mathbf{M}_h - (1-\theta)\delta^2\mathbf{A}_h\right)\mathbf{u}^{(i)} - \left(\mathbf{M}_h + \tfrac{\theta}{2}\delta^2\mathbf{A}_h\right)\mathbf{u}^{(i-1)}\right)$$

und wir erhalten ein *θ-leapfrog-Schema*,

$$\mathbf{u}^{(i+1)} = \mathbf{S}_{h,\delta,\theta}^{-1}\mathbf{R}_{h,\delta,\theta}\mathbf{u}^{(i)} - \mathbf{u}^{(i-1)},$$

wobei $\mathbf{S}_{h,\delta,\theta} := \mathbf{M}_h + \frac{\theta}{2}\delta^2\mathbf{A}_h$ und $\mathbf{R}_{h,\delta,\theta} := 2\mathbf{M}_h - (1-\theta)\delta^2\mathbf{A}_h$ sind.

Offensichtlich ist das θ-leapfrog-Schema für jedes $\theta \in \mathbb{R}$ mindestens konsistent von der Ordnung 2 und zum Beispiel für $\theta = 0$ ein explizites Verfahren respektive für $\theta = 1$ ein implizites Verfahren. Da die Wellengleichung als Anfangswertproblem steif ist (analog wie bei der Wärmeleitungsgleichung, siehe Übungsblatt 9, Aufgabe 1), muss das θ-leapfrog-Schema mit $\theta = 0$ bei zu grossen Zeitschrittweiten δ instabil werden. Dabei hängt die Schranke für die Zeitschrittweite von der Ortsdiskretisierung ab. Diese Einschränkung wird generell die Courant–Friedrichs–Lewy–Bedingung genannt. Wir werden deshalb generell $\theta = 1$ wählen wollen.

Diskretisierung der Anfangsbedingungen

Um mit dem θ-leapfrog-Schema sukzessive die Approximationen $\mathbf{u}^{(i+1)}$ berechnen zu können, benötigen wir am Anfang die *zwei* ersten Startwerte $\mathbf{u}^{(0)}$ und $\mathbf{u}^{(1)}$. Der Einfachheit halber nehmen wir an, dass die Anfangsbedingungen u_0 und v_0 hinreichend glatte Funktionen sind. Dann können wir für den ersten Startwert $\mathbf{u}^{(0)}$ einfach die Koeffizienten der Punktinterpolation von u_0 in dem Finite-Element-Raum nehmen.

Um den zweiten Startwert $\mathbf{u}^{(1)}$ zu bestimmen, betrachten wir das θ-leapfrog-Schema bei $i = 0$:

$$\mathbf{u}^{(1)} = \mathbf{S}_{h,\delta,\theta}^{-1}\mathbf{R}_{h,\delta,\theta}\mathbf{u}^{(0)} - \mathbf{u}^{(-1)}.$$

In dieser Gleichung ist neben $\mathbf{u}^{(1)}$ vorerst auch $\mathbf{u}^{(-1)}$ unbekannt. Wenn $\mathbf{v}_0$ die Koeffizienten der Punktinterpolation von v_0 in dem Finite-Element-Raum ist, dann wissen wir aber, dass

$$\frac{\partial}{\partial t}\mathbf{u}(t_0) \approx \mathbf{v}_0$$

gilt, womit wir mit der symmetrischen Differenz erster Ordnung auf die Approximation

$$\frac{\mathbf{u}(t_1) - \mathbf{u}(t_{-1})}{2\delta} \approx \mathbf{v}_0$$

kommen. Wir verlangen deshalb, dass

$$\frac{\mathbf{u}^{(1)} - \mathbf{u}^{(-1)}}{2\delta} = \mathbf{v}_0$$

gelten soll, womit wir das $\mathbf{u}^{(-1)}$ eliminieren können, und sich $\mathbf{u}^{(1)}$ schliesslich als

$$\mathbf{u}^{(1)} = \frac{1}{2}\mathbf{S}_{h,\delta,\theta}^{-1}\mathbf{R}_{h,\delta,\theta}\mathbf{u}^{(0)} + \delta\mathbf{v}_0$$

bestimmen lässt.

Aufgabe 1. Schreiben Sie eine Funktion

```
mgsys = mgsys_wave_assemble(mgmesh, T, N, theta),
```

welche die Massen- und Steifigkeitsmatrizen sowie die Systemmatrizen, $\mathbf{S}_{h,\delta,\theta}$ und $\mathbf{R}_{h,\delta,\theta}$, im `sparse`-Format für alle Dreiecksnetze in dem übergebenen Mehrgitter-Dreiecksnetz berechnet und in dem Struct `mgsys` als Cell-Arrays `mgsys.M`, `mgsys.A` `mgsys.R` und `mgsys.S` zurückgibt. Weiter sei in dem Struct `mgsys` ein leeres Cell-Array `mgsys.b`, in welchem wir jeweils für alle Dreiecksnetze in dem übergebenen Mehrgitter-Dreiecksnetz die diskrete rechte Seite speichern werden, sowie `mgsys.L` die Anzahl Level im Mehrgitter-Dreiecksnetz und `mgsys.delta` der Zeitschritt δ.

Schreiben Sie weiter eine Funktion

```
mgsys = mgsys_wave_reset_rhs(mgsys, mgmesh, u),
```

welche die rechte Seite $\mathbf{b} = \mathbf{R}_{h,\delta,\theta}\mathbf{u}$, der jeweils zu lösenden linearen Gleichung

$$\mathbf{S}_{h,\delta,\theta}\mathbf{w} = \mathbf{R}_{h,\delta,\theta}\mathbf{u},$$

auf dem feinsten Level berechnet und auf alle gröberen restringiert. Diese $\mathbf{b}$ speichern Sie jeweils in dem Struct `mgsys` in dem Cell-Array `mgsys.b`.

Aufgabe 2. Schreiben Sie eine Funktion

```
w = mgsys_wave_solve_backslash(mgmesh, mgsys),
```

welche die Gleichung

$$\mathbf{S}_{h,\delta,\theta}\mathbf{w} = \mathbf{b}$$

auf dem feinsten Level vermittels des \-Solvers löst.

Schreiben Sie weiter eine Funktion

```
w = mgsys_wave_solve_nested_dpcg(mgmesh, mgsys, eps_tol),
```

welche das gleiche mit dem diagonal vorkonditioniertes CG-Verfahren berechnet. Hierbei starten Sie aber mit dem gröbstem Level und benutzen für die feineren Level sukzessive als Startwert die Prolongation der Lösung zum eins gröberen Level.

Aufgabe 3. Schreiben Sie eine Funktion

```
mgsmoother = mgsmoother_wave_extract(mgsys, alpha),
```

welche die jeweiligen Matrizen $\{\mathbf{B}_j\}_{j=1}^{J}$ des jeweiligen Glätters[2] im `sparse`-Format in den Feldern des Struct `mgsmoother` als Cell-Arrays `mgsmoother.Bv` und `mgsmoother.Bn` zurückgeben. Dabei sei `mgsmoother.Bv` der Vor- und `mgsmoother.Bn` der Nachglätter. Die Auswahl des Glätters passiere (analog wie auf Programmierblatt 3) mit Hilfe des Arguments `alpha`: Für ein numerisches `alpha` geben sie den zugehörigen Richardson-Glätter zurück, für die Werte `alpha='j'` und `alpha='gs'` den Jacobi-Glätter respektive den Gauss-Seidel-Glätter.

Schreiben Sie weiter eine Funktion

```
w = mgsmoother_wave_vcycle(w, b, ll, mgmesh, mgsys, mgsmoother, K),
```

die einen V-Zyklus auf `w` mit K Vor- und Nachglättungen anwendet und eine Funktion

```
w = mgsys_wave_solve_nested_vcycle(mgmesh, mgsys, mgsmoother, K, J),
```

welche die Gleichung

$$\mathbf{S}_{h,\delta,\theta}\mathbf{w} = \mathbf{b}$$

mit J Iterationen des V-Zyklus mit K Vor- und Nachglättungen löst. Hierbei starten Sie aber mit dem gröbstem Level und benutzen für die feineren Level sukzessive als Startwert die Prolongation der Lösung zum eins gröberen Level.

Aufgabe 4. Schreiben Sie ein Skript, welches die Wellengleichung zeitgleich mit den drei verschiedenen Löser auf dem fünffach verfeinertem "basel"-Mehrgitter-Dreiecksnetz mithilfe des θ-leapfrog-Schemas löst und als Animation darstellt[3]. Wählen Sie dabei folgende Parameter,

```
N = 200; T = .005*N; theta = .6; eps_tol = 1e-8; alpha = 'gs'; K = 2; J = 1;
```

und als Startwerte die Funktionen, die vermittels

```
d = .02; c = [.05; .1];
u0 = (x) (norm(x-c) < d) *10* ((d-norm(x-c))./d)^2;
v0 = (x) 0;
```

definiert werden.

[2] Hierbei ist zu beachten, dass die Glätter aus den Systemmatrizen $\mathbf{S}_{h,\delta,\theta}$ gebildet werden.

[3] Benutzen Sie dazu die Funktionen `mgmesh_wave_plot_initialise.m` und `mgmesh_wave_plot_update.m` von der Vorlesungswebseite.