TECHNISCHE
UNIVERSITÄT
WIEN
Vienna University of Technology

SEMINARARBEIT

Polynomielle Eigenwertprobleme

ausgeführt am

Institut für
Analysis und Scientific Computing
TU Wien

unter der Anleitung von

Associate Prof. Dipl.-Math. Dr.rer.nat. Lothar Nannen

durch

Judith Deimel
Matrikelnummer: 11819985

und

Rebecca Weiß
Matrikelnummer: 11819986

Wien, am 26. Februar 2021

Inhaltsverzeichnis

1 Einleitung **1**

2 Definition **2**

3 Motivation **3**

4 Diskretisierung **5**

5 Numerische Lösung **7**
5.1 Vom polynomiellen zum linearen Eigenwertproblem 7
5.2 Berechnung der Inversen $(A - \rho B)^{-1}$ 8

6 Implementierung **12**

7 Anhang **15**
7.1 Code 15

Literaturverzeichnis **16**

1 Einleitung

Diese Arbeit beschäftigt sich mit dem numerischen Lösen von polynomiellen Eigenwertproblemen. Zu Beginn wird beschrieben, wie man von einer gewöhnlichen Differentialgleichung zu einem polynomiellen Eigenwertproblem kommt. Dazu ist eine Diskretisierung notwendig. Anschließend wird das polynomielle Eigenwertproblem in ein verallgemeinertes lineares Eigenwertproblem umgeformt. Dadurch werden allerdings die Dimensionen der Matrizen größer, weshalb eine effizientere numerische Lösungsmethode erforderlich ist. Die effiziente Berechnung der Inversen ist das Hauptziel der Arbeit. Mithilfe des Schur-Komplements ist es möglich, die Größe der zu invertierenden Matrix von der Polynomordnung unabhängig zu machen. Zum Abschluss wird die Implementierung beschrieben und ein Laufzeitvergleich durchgeführt.

Als Vorlage wurde der Abschnitt *Polynomielle Eigenwertprobleme* des Skriptes „Eigenwertprobleme“ [Nan18] verwendet. Die Beweise sind ebenfalls diesem Werk entnommen und teilweise ergänzt.

2 Definition

Unter einem polynomiellen Eigenwertproblem versteht man die Suche nach einem Eigenpaar (λ, u) mit $u \neq 0$, sodass $L(\lambda, u) = 0$, wobei der Operator L linear von u und polynomiell von $\lambda \in \mathbb{C}$ abhängt. Im Folgenden sei der Operator $L(\lambda)$ von der Form

$$\sum_{j=0}^{n} \lambda^j L_j(\lambda),$$

wobei $\forall j : L_j : V \to V$ beschränkte, lineare Operatoren auf dem Hilbertraum $(V, (\cdot, \cdot))$ sind.

3 Motivation

Wir wollen uns nun damit beschäftigen, warum wir uns überhaupt mit polynomiellen Eigenwertproblemen befassen. Dazu betrachten wir zur Motivation das Beispiel einer partiellen Differentialgleichung, die wir auf ein polynomielles Eigenwertproblem zurückführen und dieses dann mit unserem im nächsten Abschnitt entwickelten Löser numerisch lösen werden.

Wir betrachten die sogenannte *Telegraphen-Gleichung* der Form

$$\frac{\partial^2}{\partial x^2}u = \frac{1}{c^2}\frac{\partial^2}{\partial t^2}u + \gamma\frac{\partial}{\partial t}u + ku \tag{3.1}$$

mit Konstanten $c, \gamma, k > 0$. Mithilfe dieser Gleichung kann man zum Beispiel die Spannung $u(x,t)$ in einem Übertragungskabel simulieren, wobei die Konstanten vom Widerstand, der Induktion, der Kapazität und dem Stromverlust im Kabel abhängen.

Wir verwenden zuerst einen zeitharmonischen Ansatz, um die instationäre Gleichung in eine stationäre Gleichung umzuformen. Sei also $u(x,t) = \Re\{\exp(-i\omega t)v(x)\}$ mit $\omega \in \mathbb{C}$. Wenn wir in dieser Darstellung die benötigten partiellen Ableitungen bilden und in die Gleichung 3.1 einsetzen, erhalten wir

$$\Re\{\exp(-i\omega t)\Delta v(x)\} = \Re\{\frac{(-i\omega)^2}{c^2}\exp(-i\omega t)v(x) - \gamma i\omega\exp(-i\omega t)v(x) + k\exp(-i\omega t)v(x)\}$$

bzw.

$$\Re\Big\{\big(-\Delta v(x) - \frac{\omega^2}{c^2}v(x) - \gamma i\omega v(x) + kv(x)\big)\exp(-i\omega t)\Big\} = 0$$

Dies wiederum ist äquivalent zur Gleichung

$$(-\Delta + k\mathrm{id})v(x) - \gamma i\omega v(x) - \frac{\omega^2}{c^2}v(x) = 0 \tag{3.2}$$

.

Es handelt sich um ein quadratisches Eigenwertproblem in ω.
Um die Gleichung 3.2 weiter umformen zu können machen wir einen kurzen Exkurs in die Theorie der partiellen Differentialgleichungen.

Wir nehmen im Weiteren sogenannte Neumann-Randbedingungen an. Das heißt wir verlangen von einer Lösung $v \in C^2(\overline{\Omega})$ der Gleichung 3.2, dass sie zusätzlich

$$\frac{\partial v}{\partial n}(x) = 0 \qquad \forall x \in \partial\Omega \tag{3.3}$$

erfüllt. Die Gleichungen 3.2 und 3.3 bilden gemeinsam die *starke* Formulierung des Problems. Um diese nun tatsächlich zu lösen, ist es üblich die *schwache* Formulierung zu betrachten.

Für die Herleitung dieser sind die Sobolevräume von großer Bedeutung. Da ein tieferer Einblick in die Theorie der Sobolevräume den Rahmen dieser Arbeit sprengen würde, wollen wir hier nur eine Definition angeben.

Definition 1. [1] *Sei $\Omega \subset \mathbb{R}^n$ offen. Der Abschluss von $X = \{u \in C^\infty(\Omega) : \|u\|_{H^k(\Omega)} < \infty\}$ bezüglich der Norm $\|\cdot\|_{H^k(\Omega)}$ ist der Sobolevraum $H^k(\Omega)$. Der Abschluss von $C_0^\infty(\Omega)$ bezüglich der Norm $\|\cdot\|_{H^k(\Omega)}$ ist der Sobolevraum $H_0^k(\Omega)$. Wir schreiben :*

$$H^k(\Omega) = \overline{X}, \quad H_0^k(\Omega) = \overline{C_0^\infty(\Omega)}$$

Die Norm $\|\cdot\|_{H^k(\Omega)}$ ist auf $C^\infty(\overline{\Omega})$ definiert durch

$$\|u\|_{H^k(\Omega)} = \sqrt{(u,u)_{H^k}}$$

wobei

$$(u,v)_{H^k} := \sum_{|\alpha| \le k} \int_\Omega D^\alpha u D^\alpha v \, dx$$

.

Diese Räume haben die Eigenschaft vollständig bezüglich ihrer Norm und damit Hilberträume zu sein.

Mithilfe dieser Räume können wir nun das schwache Problem formulieren. Dazu geht man folgendermaßen vor: Wir multiplizieren zunächst die Gleichung 3.2 mit einer Testfunktion $w \in C_0^\infty(\Omega)$ und integrieren über Ω. Mit partieller Integration gilt

$$-\int_\Omega (\Delta v) w \, dx = \int_\Omega \nabla v \cdot \nabla w \, dx - \underbrace{\int_{\partial\Omega} \frac{\partial v}{\partial n} w \, ds}_{=0}$$

Das Randintegral verschwindet aufgrund der Annahme von 3.3.

Damit haben wir insgesamt folgenes Problem:
Finde $(\omega, v) \in \mathbb{C} \times H^1(\Omega)$, so dass

$$\underbrace{\int_\Omega (\nabla v \cdot \nabla w + kvw) dx}_{=:a(v,w)} - \omega \, i \underbrace{\int_\Omega \gamma v w dx}_{=:b_1(v,w)} - \omega^2 \underbrace{\int_\Omega \frac{1}{c^2} v w dx}_{=:b_2(v,w)} = 0 \quad \forall w \in H^1(\Omega) \tag{3.4}$$

bzw.

$$a(v,w) + \omega b_1(v,w) + \omega^2 b_2(v,w) = 0 \quad \forall w \in H^1(\Omega) \tag{3.5}$$

Wir wollen hier noch kurz argumentieren, warum es überhaupt sinnvoll ist homogene Neumann-Randbedingungen vorauszusetzen. Wir haben am Anfang dieses Abschnittes angesprochen, dass man die Telegraphengleichung dazu nutzen kann, die Spannung in einem Übertragungskabel zu modellieren. Nun gibt die Normalenableitung $\partial v/\partial n$ den Spannungsaustausch mit der Umgebung an. Nun ist es sinnvoll ein isoliertes Kabel zu betrachten, bei dem solch ein Spannungsaustausch nicht stattfinden kann.

[1] Diese Definition stammt aus [Jü20]

4 Diskretisierung

Wir haben im letzten Abschnitt unser Motivationsbeispiel auf die Form
Suche $(\omega, v) \in \mathbb{C} \times H^1(\Omega)\backslash\{0\}$, so dass:

$$a(v,w) + \omega b_1(v,w) + \omega^2 b_2(v,w) = 0, \quad w \in H^1(\Omega) \tag{4.1}$$

gebracht. Wir wollen dieses Problem nun mithilfe der Finite-Elemente-Methode diskretisieren und die Bilinearformen durch Matrizen darstellen.

Die Finite-Elemente-Methode besteht darin, das Problem 4.1 auf einem endlich dimensionalen Unterraum zu betrachten und dort zu lösen.

Dafür brauchen wir zunächst den Begriff der Triangulierung. Wir verwenden hier die Definition aus [FP21].

Definition 2. *Eine Menge $\mathcal{T}$ heißt **Triangulierung** von Ω, wenn*

- *$\mathcal{T}$ eine endliche Menge an nicht-degenerierten Dreiecken ist,*
- *der Abschluss von Ω von $\mathcal{T}$ überdeckt wird, d.h. $\overline{\Omega} = \bigcup_{T\in\mathcal{T}} T$*
- *für alle $T, T' \in \mathcal{T}$ mit $T \neq T'$ gilt $|T \cap T'| = 0$, der Schnitt also Maß 0 hat.*

Wir sprechen von einer konformen bzw. regulären Triangulierung, falls der Schnitt von zwei Elementen $T, T' \in \mathcal{T}$ mit $T \neq T'$

- *leer,*
- *ein gemeinsamer Knoten (Eckpunkt),*
- *oder eine gemeinsame Kante ist.*

Für solch eine reguläre Triangulierung $\mathcal{T}$ können wir den Raum der elementweise affinen Funktionen betrachten.

Proposition 1. *Sei $\mathcal{T}$ eine reguläre Triangulierung von Ω. Wir definieren*

$$\mathcal{S}^1(\mathcal{T}) := \{v_h \in C(\Omega) |\, \forall T \in \mathcal{T} : v_h|_T \text{ affin}\}$$

als den Raum der elementweise affinen und global stetigen Funktionen. Dann gilt:

1. *$\mathcal{S}^1(\mathcal{T})$ ist ein N-dimensionaler Unterraum von $H^1(\Omega)$, wobei $N = \#\mathcal{K}$ die Anzahl der Knoten ist.*
2. *Für jeden Knoten $z \in \mathcal{K}$ existiert eine eindeutige **Hutfunktion***

$$\zeta_z \in \mathcal{S}^1(\mathcal{T}) \quad \text{mit} \quad \zeta_z(z') = \delta_{zz'} \quad \forall z' \in \mathcal{K} \tag{4.2}$$

3. Die Menge $\{\zeta_z | z \in \mathcal{K}\}$ *ist eine Basis von* $\mathcal{S}^1(\mathcal{T})$.

Beweis. Die Proposition sowie ihr Beweis ist in [FP21] zu finden. □

Mithilfe dieser Basis von Hutfunktionen wollen wir nun die Bilinearformen aus 4.1 durch Matrizen darstellen.

Allgemein können wir auf einem endlichdimensionalen Unterraum X_h von $H^1(\Omega)$ die Einschränkung von einer Bilinearform $a : H^1(\Omega) \times H^1(\Omega) \to \mathbb{R}$ betrachten.

Lemma 2. *Sei* $\{\phi_1, \dots, \phi_N\}$ *eine Basis des endlichdimensionalen Unterraumes* $X_h \subset H^1(\Omega)$. *Dann kann man die Einschränkung der Bilinearform* $a : H^1(\Omega) \times H^1(\Omega) \to \mathbb{R}$ *auf* $X_h \times X_h$ *durch die Matrix* A *mit den Einträgen*

$$A_{jk} := a(\phi_j, \phi_k) \quad j, k = 1 \dots, N \tag{4.3}$$

darstellen.

Beweis. Diese Überlegung lässt sich leicht beweisen indem wir einfach zwei Elemente $u = \sum_{i=1}^N u_i\phi_i$, $v = \sum_{i=1}^N v_i\phi_i$ aus X_h nehmen und sehen

$$a(u, v) = \sum_{j,k=1}^{N} u_j v_k a(\phi_j, \phi_k) = \sum_{j=1}^{N} u_j (A_j v) = \tilde{u}^T A \tilde{v} \tag{4.4}$$

wobei A_j die j-te Zeile der Matrix A und $\tilde{u} = (u_1, \dots, u_N)^T$ resp. $\tilde{v} = (v_1, \dots, v_N)^T$ den Koeffizientenvektor von u resp. v bezeichnet. □

Damit haben wir das diskrete Problem:
Suche $(\omega_h, v_h) \in \mathbb{C} \times \mathcal{S}^1(\mathcal{T}) \backslash \{0\}$, so dass:

$$a(v_h, w) + \omega_h b_1(v_h, w) + \omega_h^2 b_2(v_h, w) = 0 \quad \forall w \in \mathcal{S}^1(\mathcal{T}) \tag{4.5}$$

bzw.

Suche $(\omega_h, v_h) \in \mathbb{C} \times \mathbb{C}^N \backslash \{0\}$

$$v_h^T A w + \omega_h v_h^T B_1 w + \omega_h^2 v_h^T B_2 w = 0 \quad \forall w \in \mathbb{C}^N \tag{4.6}$$

falls $A, B_1, B_2 \in \mathbb{C}^{N \times N}$ die Matrizen zu den Einschränkungen der Bilinearformen auf $\mathcal{S}^1(\mathcal{T})$ bezeichnen.

Wir sehen, dass die Dimension der Matrizen von der Dimension des Unterraumes $\mathcal{S}^1(\mathcal{T})$ abhängt. Diese steigt mit der Feinheit der Triangulierung, denn ein feineres Gitter mit mehr Knoten impliziert eine größere Anzahl von Hutfunktionen, die an einem Knoten den Wert 1 und an den anderen den Wert 0 annehmen.

Das heißt wir haben es mit großen Matrizen zu tun. Nun haben die gewählten Basisfunktionen von $\mathcal{S}^1(\mathcal{T})$ - die Hutfunktionen - einen lokalen Träger, womit die Einträge der Matrix, die ja durch die Anwendung der Bilinearform auf zwei Basisfunktionen definiert werden, sehr oft Null sind. Wir arbeiten also mit Sparse-Matrizen.

5 Numerische Lösung

5.1 Vom polynomiellen zum linearen Eigenwertproblem

Wir haben in den letzten beiden Abschnitten motiviert, wieso wir uns überhaupt mit polynomiellen EWPs beschäftigen und wie man auf solche kommen kann. Nun wollen wir uns der Entwicklung eines numerischen Lösers widmen.

Wir wollen also das folgende Problem numerisch lösen.

$$\text{Suche } (\lambda, u) \in \mathbb{C} \times \mathbb{C}^N \backslash \{0\} : \; L(\lambda)u = 0, \quad \text{wobei } L(\lambda) := \sum_{j=0}^{n} \lambda^j L_j. \tag{5.1}$$

In Fall des Beispiel aus Abschnitt 3 sind die L_j die Matrixdarstellungen der Bilinearformen a, b_1, b_2.

Das Problem 5.1 können wir nun - ähnlich wie bei der Transformation von einer Differentialgleichung in ein System erster Ordnung - auf ein verallgemeinertes lineares Eigenwertproblem umformen.

Dazu setzen wir $y = (u, \lambda u, \lambda^2 u, \ldots, \lambda^{n-1} u)^T$. Damit lautet das umgeformte Problem

$$\text{Suche } (\lambda, y) \in \mathbb{C} \times \mathbb{C}^{nN} \backslash \{0\} : \quad Ay = \lambda By \tag{5.2}$$

mit den Matrizen $A, B \in \mathbb{C}^{nN \times nN}$

$$A := \operatorname{diag}(L_0, \mathrm{id}, \ldots, \mathrm{id}), \quad B := \begin{pmatrix} -L_1 & -L_2 & \cdots & -L_n \\ \mathrm{id} & 0 & \vdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ 0 & 0 & \mathrm{id} & 0 \end{pmatrix}$$

Proposition 3. *Das Problem 5.2 ist äquivalent zum Problem 5.1.*

Beweis. Die Aussage folgt durch Vergleichen der linken und rechten Seite in 5.2. In den ersten N Zeilen haben wir die Gleichung

$$L_0 u = -\lambda \sum_{j=1}^{n} L_j \lambda^{j-1} u = -\sum_{j=1}^{n} \lambda^j L_j u$$

Hier handelt es sich genau um die Gleichung, die wir auch in Problem 5.1 haben.

Für die restlichen Einträge überlegt man sich leicht, dass hier die folgende Gleichung steht:

$$\begin{pmatrix} \lambda u \\ \lambda^2 u \\ \cdots \\ \lambda^{n-1} u \end{pmatrix} = \begin{pmatrix} \lambda u \\ \lambda^2 u \\ \cdots \\ \lambda^{n-1} u \end{pmatrix}$$

Somit sehen wir, dass für eine Lösung (λ, u) des polynomiellen Eigenwertproblems (λ, y) mit $y := (u, \lambda u, \ldots, \lambda^{n-1}u)$ eine Lösung des linearen EWPs ist. Umgekehrt für eine Lösung (λ, y) des linearen Problems ist $(\lambda, y[1, \ldots, N])$ eine Lösung des polynomiellen. Die Probleme sind also tatsächlich äquivalent. □

Wir können also das polynomielle EWP lösen, indem wir es auf ein lineares zurückführen.
Das Problem 5.2 können wir nun mit dem *Shift-and-Invert*-Prinzip umformulieren: Sei $\rho \in \mathbb{C}$ kein Eigenwert von A. Dann gilt für $\lambda = \frac{1}{\mu} + \rho$

$$Ay = \lambda By = \Big(\frac{1}{\mu} + \rho\Big)By \iff (A - \rho B)^{-1}By = \mu y$$

.

Der Shiftparameter ρ steuert zu dem auch, welche Eigenwerte wir berechnen. Denn im Allgemeinen werden durch die verschiedenen Methoden die betragsmäßig größten Eigenwerte berechnet. Diese sind zum einen numerisch ziemlich instabil, zum anderen sind wir in der Praxis eher an den kleinsten interessiert.

Ein Paradebeispiel stammt aus der Musik. Wenn wir uns für die Schwingungen z.B. einer eingespannten Saite interessieren und welche Töne daraus entstehen, so brauchen wir den Grundton und die nächsten paar Obertöne. Diese entsprechen den Eigenschwingungen zu den kleinsten Eigenwerten. Wir sind also hauptsächlich an diesen interessiert.

Mithilfe des Shiftparameters können wir dieses Problem aber nun umgehen. Wir berechnen nämlich im umgeformten linearen Problem die größten Eigenwerte μ und damit jene λ des verallgemeinerten linearen EWP, die besonders nahe an ρ liegen. Um möglichst kleine Eigenwerte zu erhalten werden wir ρ also in der Regel sehr nahe an Null, z.B. 10^{-9} wählen.

Man könnte nun das lineare Problem mit bereits bekannten Methoden einfach lösen. Allerdings sind die Matrizen A, B aus $\mathbb{C}^{nN \times nN}$. Dieses System ist um einiges größer das ursprüngliche. Die Inverse $(A - \rho B)^{-1}$ explizit zu berechnen wäre also sehr teuer. Dies wollen wir umgehen und leiten deshalb eine Methode her, bei der wir für die Bildung der Inversen letzendlich wieder nur eine Matrix der Dimension $N \times N$ invertieren müssen.

5.2 Berechnung der Inversen $(A - \rho B)^{-1}$

Um die Inverse effizient berechnen zu können, werden wir das Schur-Komplement anwenden, welches eine Darstellung der Inversen einer Blockmatrix ermöglicht.

Definition 3. *Sei $C \in \mathbb{C}^{(n+m)\times(n+m)}$ eine Blockmatrix folgender Form:*

$$C = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix},$$

wobei $C_{11} \in \mathbb{C}^{n\times n}$, $C_{12} \in \mathbb{C}^{n\times m}$, $C_{21} \in \mathbb{C}^{m\times n}$ und $C_{22} \in \mathbb{C}^{m\times m}$. Die Matrix $S \in \mathbb{C}^{n\times n}$ heißt Schur-Komplement von C_{22} in C, wenn $S = C_{11} - C_{12}C_{22}^{-1}C_{21}$ gilt.

Lemma 4. *Sei S das Schur-Komplement von C_{22} in C. Damit gilt folgende Darstellung für die Inverse von C:*

$$\begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}^{-1} = \begin{pmatrix} I_n & 0 \\ -C_{22}^{-1}C_{21} & I_m \end{pmatrix} \begin{pmatrix} S^{-1} & 0 \\ 0 & C_{22}^{-1} \end{pmatrix} \begin{pmatrix} I_n & -C_{12}C_{22}^{-1} \\ 0 & I_m \end{pmatrix}$$

Beweis. Wir betrachten das lineare Gleichungssystem

$$\begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} t_1 \\ t_2 \end{pmatrix}. \tag{5.3}$$

Nun schreiben wir die beiden Zeilen einzeln an und formen (unter der Voraussetzung, dass C_{22} invertierbar ist) um:

$$C_{11}x + C_{12}y = t_1 \tag{5.4}$$

$$C_{21}x + C_{22}y = t_2 \quad \Rightarrow \quad y = C_{22}^{-1}(t_2 - C_{21}x) \tag{5.5}$$

Die Darstellung 5.5 von y setzen wir in 5.4 ein und formen um:

$$C_{11}x + C_{12}(C_{22}^{-1}(t_2 - C_{21}x)) = t_1 \Leftrightarrow \tag{5.6}$$

$$\underbrace{(C_{11} - C_{12}C_{22}^{-1}C_{21})}_{=:S} x = t_1 - C_{12}C_{22}^{-1}t_2. \tag{5.7}$$

Unter Voraussetzung der Invertierbarkeit von S erhalten wir aus 5.7 die Darstellung $x = S^{-1}(t_1 - C_{12}C_{22}^{-1}t_2)$, welche wir in 5.5 einsetzen können:

$$y = C_{22}^{-1}(t_2 - C_{21}(S^{-1}(t_1 - C_{12}C_{22}^{-1}t_2))). \tag{5.8}$$

Die Darstellung 5.8 für y sowie die obige Darstellung für x können wir ausmultiplizieren und erhalten dann eine Form, die wir in das von links mit C^{-1} multiplizierte Gleichungssystem 5.3 einsetzen können:

$$\begin{aligned} x &= S^{-1}t_1 - S^{-1}C_{12}C_{22}^{-1}t_2 \\ y &= -C_{22}^{-1}C_{21}S^{-1}t_1 + (C_{22}^{-1} + C_{22}^{-1}C_{21}S^{-1}C_{12}C_{22}^{-1})t_2 \end{aligned}$$

Daraus ergibt sich:

$$\begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}^{-1} = \begin{pmatrix} S^{-1} & -S^{-1}C_{12}C_{22}^{-1} \\ -C_{22}^{-1}C_{21}S^{-1} & C_{22}^{-1} + C_{22}^{-1}C_{21}S^{-1}C_{12}C_{22}^{-1} \end{pmatrix}$$

Durch Ausführen der Matrixmultiplikationen zeigt man sofort, dass diese Darstellung äquivalent zur gewünschten Zerlegung ist. □

Diese Darstellung der Inversen aus Lemma 4 kann nun wiederholt auf die Inverse der Matrix

$$A - \rho B = \begin{pmatrix} A_0 + \rho A_1 & \rho A_2 & \cdots & \rho A_n \\ -\rho\mathrm{id} & \mathrm{id} & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ 0 & 0 & -\rho\mathrm{id} & \mathrm{id} \end{pmatrix}$$

angewendet werden. Nach dem ersten Schritt erhält man folgende Darstellung der Inversen

$$(A-\rho B)^{-1} = \underbrace{\begin{pmatrix} \mathrm{id} & 0 \\ (0,\cdots,0,\rho\mathrm{id}) & \mathrm{id} \end{pmatrix}}_{L_1} \begin{pmatrix} D_1^{-1} & 0 \\ 0 & \mathrm{id} \end{pmatrix} \underbrace{\begin{pmatrix} \mathrm{id} & \begin{pmatrix} -\rho A_n \\ 0 \\ \vdots \\ 0 \end{pmatrix} \\ 0 & \mathrm{id} \end{pmatrix}}_{R_1} \tag{5.9}$$

wobei

$$D_1 = \begin{pmatrix} A_0+\rho A_1 & \rho A_2 & \cdots & \rho A_{n-2} & \rho A_{n-1}+\rho^2 A_n \\ -\rho\mathrm{id} & \mathrm{id} & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & -\rho\mathrm{id} & \mathrm{id} & 0 \\ 0 & \cdots & 0 & -\rho\mathrm{id} & \mathrm{id} \end{pmatrix}.$$

Analog wie im vorherigen Schritt können wir nun Lemma 4 für die Berechnung der Inversen der Matrix D_1 anwenden und erhalten eine analoge Darstellung wie in 5.9 mit dem Unterschied, dass die Matrix D_2 nun folgende Form hat:

$$D_2 = \begin{pmatrix} A_0+\rho A_1 & \rho A_2 & \cdots & \rho A_{n-3} & \rho A_{n-2}+\rho^2 A_{n-1}+\rho^3 A_n \\ -\rho\mathrm{id} & \mathrm{id} & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & -\rho\mathrm{id} & \mathrm{id} & 0 \\ 0 & \cdots & 0 & -\rho\mathrm{id} & \mathrm{id} \end{pmatrix}$$

Die Dimension der zu invertierenden Matrix D_i verringert sich dabei in jedem Schritt. Wir wenden Lemma 4 n-mal an. Dies führt dazu, dass Matrix D_n, der Inverse zu berechnen ist, Dimension $N \times N$ hat. Die Größe der zu berechnenden Inversen hängt somit nicht mehr von der Polynomordnung ab. Ausmultiplizeren der Matrizen L_i bzw. R_i, die sich aus der wiederholten Anwendung von Lemma 4 ergeben, liefert schließlich folgende Darstellung der Inversen:

$$(A-\rho B)^{-1} = \begin{pmatrix} \mathrm{id} & & & & \\ \rho\mathrm{id} & \mathrm{id} & & & \\ \rho^2\mathrm{id} & \ddots & \ddots & & \\ \vdots & \ddots & \ddots & \ddots & \\ \rho^{n-1}\mathrm{id} & \cdots & \rho^2\mathrm{id} & \rho\mathrm{id} & \mathrm{id} \end{pmatrix} \begin{pmatrix} P(\rho)^{-1} & & & \\ & \mathrm{id} & & \\ & & \ddots & \\ & & & \mathrm{id} \end{pmatrix} \begin{pmatrix} \mathrm{id} & E_2 & \cdots & E_n \\ & \mathrm{id} & & \\ & & \ddots & \\ & & & \mathrm{id} \end{pmatrix} \tag{5.10}$$

wobei $E_j := -\sum_{l=j}^{n} \rho^{l+1-j} A_l$ und $P(\rho) = A_0 + \rho A_1 + \cdots + \rho^n A_n$.

Um die Implementierung noch effizienter gestalten, rechnen wir nicht explizit mit diesen drei Matrizen. Wir betrachten nur den linearen Operator, der durch die Matrix definiert wird. Dazu führen wir zuerst die Matrixmultiplikation in 5.10 aus und erhalten somit:

$$(A-\rho B)^{-1} = \begin{pmatrix} P^{-1} & P^{-1}E_2 & P^{-1}E_3 & \cdots & P^{-1}E_n \\ \rho P^{-1} & \mathrm{id}+\rho P^{-1}E_2 & \rho P^{-1}E_3 & \cdots & \rho P^{-1}E_n \\ \rho^2 P^{-1} & \rho\mathrm{id}+\rho^2 P^{-1}E_2 & \mathrm{id}+\rho^2 P^{-1}E_3 & \cdots & \rho^2 P^{-1}E_n \\ \vdots & \vdots & \vdots & & \vdots \\ \rho^{n-1}P^{-1} & \rho^{n-2}\mathrm{id}+\rho^{n-1}P^{-1}E_2 & \rho^{n-3}\mathrm{id}+\rho^{n-1}P^{-1}E_3 & \cdots & \mathrm{id}+\rho^{n-1}P^{-1}E_n \end{pmatrix}$$

Wie bereits zu Beginn des Kapitels beschrieben, müssen wir die Matrix $(A-\rho B)^{-1}B =$

$$\begin{pmatrix} -P^{-1}(A_1-E_2) & -P^{-1}(A_2-E_3) & \cdots & -P^{-1}(A_{n-1}-E_n) & -P^{-1}A_n \\ -\rho P^{-1}(A_1-E_2)+\mathrm{id} & -\rho P^{-1}(A_2-E_3) & \cdots & -\rho P^{-1}(A_{n-1}-E_n) & -\rho P^{-1}A_n \\ -\rho^2 P^{-1}(A_1-E_2)+\rho\mathrm{id} & -\rho^2 P^{-1}(A_2-E_3)+\mathrm{id} & \cdots & -\rho^2 P^{-1}(A_{n-1}-E_n) & -\rho^2 P^{-1}A_n \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ -\rho^{n-1}P^{-1}(A_1-E_2)+\rho^{n-2}\mathrm{id} & -\rho^{n-1}P^{-1}(A_2-E_3)+\rho^{n-3}\mathrm{id} & \cdots & -\rho^{n-1}P^{-1}(A_{n-1}-E_n)+\mathrm{id} & -\rho^{n-1}P^{-1}A_n \end{pmatrix} \tag{5.11}$$

betrachten, um das verallgemeinerte Eigenwertproblem zu lösen. Betrachten wir nun also den linearen Operator $y \mapsto (A-\rho B)^{-1}By =: \tilde{y}$. Sei also $y = (y_1, \cdots, y_n)^T$ ein Vektor, wobei $\forall i = 1, \cdots, n$ gilt, dass y_1 Länge N hat. Multiplikation mit 5.11 ergibt, dass $\tilde{y}_1$ die Form

$$\tilde{y}_1 = -P^{-1} \cdot \Big(\sum_{n=1}^{N}((A_i - E_{i+1})y_i) + A_n \cdot y_n\Big)$$

hat. Aufgrund der regelmäßigen Struktur der Matrix 5.11 gilt $\forall i = 2, \cdots, n: \ \tilde{y}_i = y_{i-1} + \rho\tilde{y}_{i-1}$.

6 Implementierung

In diesem Abschnitt betrachten wir ein paar besondere Teile in der tatsächlichen Implementierung des im vorherigen Abschnitt entwickelten Lösers. Außerdem betrachten wir Laufzeitplots, die uns zeigen sollen, dass die neue Art die Inverse $(A-\rho B)^{-1}$ zu berechnen tatsächlich einen Speedup bringt. Für den vollständigen Code sei auf den Anhang verwiesen.

Wir verwenden nun den Löser des vorherigen Abschnittes, um die Telegraphengleichung mit homogenen Neumann-Randbedingungen auf einem 2×1- Rechteck zu lösen. Die Implementierung wurde in der Programmiersprache Python gemacht.

Für die Diskretisierung und das Aufstellen der benötigten Bilinearformen wurde die Software *Netgen/NGSolve* (`https://ngsolve.org/`) verwendet. Damit ist es uns möglich, für unser Gebiet mit wenigen Zeilen eine Triangulierung mit bestimmtem maximalen Durchmesser der Elemente zu generieren und die Bilinearformen auf dieser zu assemblieren. Diese werden intern als NGSolve-Bilinearformen abgespeichert. Generell haben wir in unserer Implementierung vor allem mit NGSolve-Datentypen gearbeitet (z.B. Matrizen), da die Operationen wie das Bilden der Inverse auf diesen für unsere Zwecke eindeutig effizienter funktioniert, als auf den gewöhnlichen Sparse-Matrizen aus dem *scipy.sparse*-Package.

Wir bemerken beispielsweise, dass alle Bilinearformen - oder besser die Matrixdarstellung davon - dieselbe Struktur haben. Wie wir im Abschnitt 4 bereits gesehen haben, handelt es sich aufgrund der lokalen Träger der Hutfunktionen um Sparse-Matrizen. Da aber alle Bilinearformen bezüglich dieser Basis diskretisiert werden, haben die Matrizen an denselben Stellen Nulleinträge. Im Falle von NGSolve-Matrizen wird diese Struktur dem Compiler mitgeteilt. Damit wird die Matrixaddition, die beim Aufstellen von $P(\rho)$ benötigt wird, um einiges effizienter. Es muss nämlich nicht mehr zuerst festgestellt werden, wo sich die nichttrivialen Einträge befinden und ggf. mehr Speicher allokiert werden, wie dies z.B. bei den *scipy.sparse*-Matrizen der Fall ist.

Weiters - wie im vorherigen Abschnitt schon angesprochen wurde - wollen wir verhindern, die Inverse des vergrößerten, linearen EWP explizit aufzustellen und diese dann dem von *scipy.sparse* zur Verfügung gestellten Eigenwertlöser zu übergeben. Wir definieren dazu in unserer Implementierung eine Funktion, die dem Eigenwertlöser als linearer Operator übergeben wird. Es ist nämlich nicht notwendig dem Löser die explizite Matrix zu übergeben, sonder es reicht anzugeben, wie der Operator auf einem Vektor agiert. In dieser Funktion `matvec` zerlegen wir den Vektor $y \in \mathbb{C}^{nN}$ in gleichgroße Teile y_i, $i = 1, \dots, n$ und agieren auf diesen wie es am Ende von Abschnitt 5.2 beschrieben wird. Danach müssen die einzelnen Teile wieder zusammengeklebt werden, damit es sich tatsächlich um einen Operator $\mathbb{C}^{nN} \to \mathbb{C}^{nN}$ handelt.

Zuletzt wurden noch Laufzeitplots erstellt um zu überprüfen, ob wir durch unseren Löser tatsächlich einen Speed-up erzielen konnten.

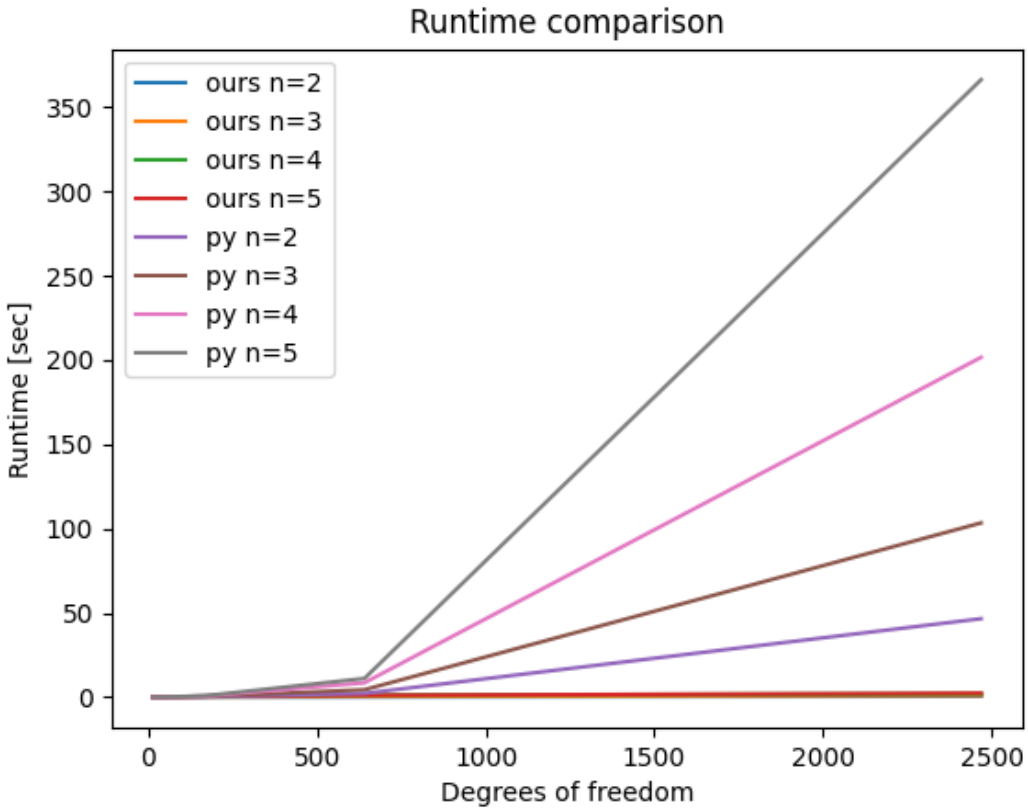

Abbildung 6.1: Lauzeit für verschiedene Polynomordnungen

Wir haben sowohl das Mesh verfeinert, wodurch die Anzahl der Freiheitsgrade bzw. die Dimension des Unterraumes wächst, als auch die Polynomordnung der EWPs erhöht. Wir haben den im vorherigen Abschnitt entwickelten Löser mit einer naiven Implementierung, in der die Matrix $A - \rho B$ explizit aufgestellt und invertiert wird, verglichen.

In Abbildung 6.1 sind die Laufzeiten zu verschiedenen Polynomordnungen geplottet. Wir sehen einen deutlichen Speed-up, insbesondere wenn man die Polynomordnung erhöht.

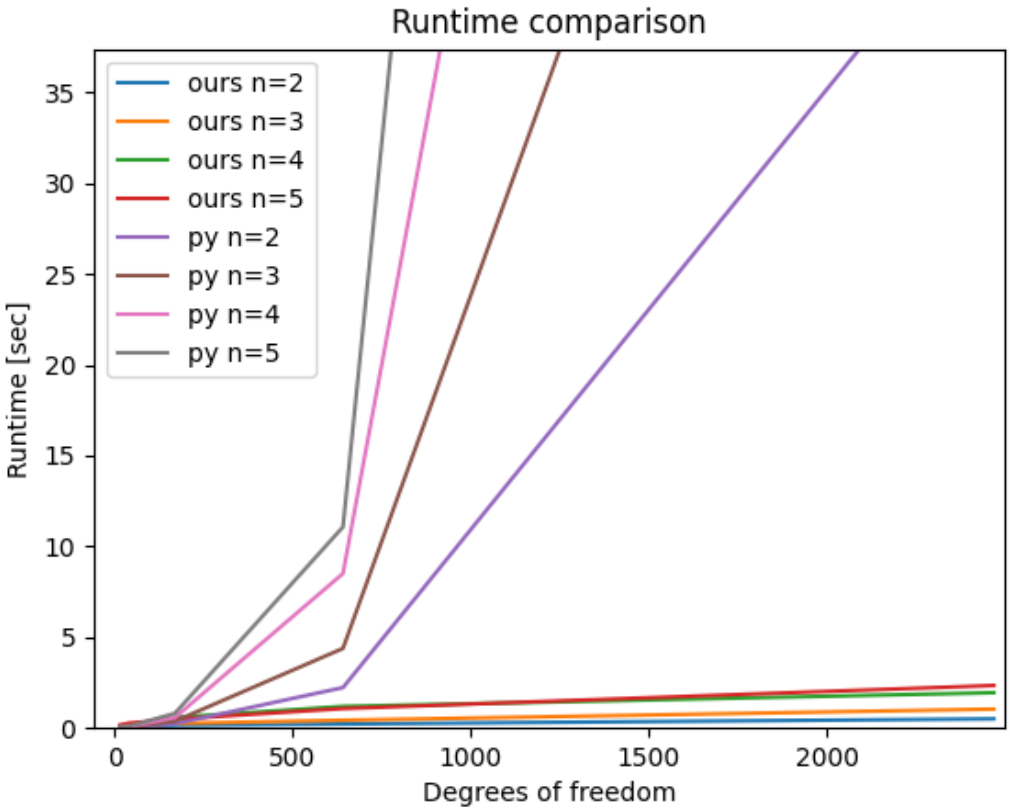

Abbildung 6.2: Laufzeit für verschiedene Polynomordnungen

In Abbildung 6.2 wurde die y-Achse eingeschränkt, um den Unterschied in den Laufzeiten noch etwas deutlicher zu sehen.

7 Anhang

7.1 Code

Algorithmus 1: Lösen des polynomiellen EWPs

Input : L Array, das die Bilinearformen enthält, ρ Shift-Parameter, $num \in \mathbb{N}$ Anzahl Eigenwerte

$n := L.length - 1$
$polynom = \sum_{i=0}^{n} \rho^i L[i]$
$y, tmp :=$ Netgen-Array von Hilfsvektoren, haben Struktur des Lösungsvektors
def `matvec(`v`):`
    `// `v`... Numpyvektor der Länge `nN
    for $i = 1, \ldots, n$ **do**
        Teile v in Vektoren der Länge N und speichere diese in y
    end
    invpoly = polynom.Inverse
    $tmp[1] = invpoly * (\sum_{n=1}^{N}((A_i - E_{i+1})y[i]) + A_n * y[n])$
    for $i = 2, \ldots, n$ **do**
        $tmp[i] = y[i-1] + \rho tmp[i-1]$
    end
    $res :=$ Numpy-Array der Länge nN
    Füge die einzelnen Vektoren in tmp zu einem zusammen und speichere in res
    return res
$A :=$ Linearer Operator der auf `matvec` basiert
$\mu, vecs :=$ Löse EWP $Ax = \mu x$
$\lambda := \rho + \frac{1}{\mu}$

Output: $\lambda, vecs[1, \ldots, N][\cdot]$ num erste Approximationen an EW und EV

Literaturverzeichnis

[FP21] Associate Prof. Dipl.-Ing. Dr.techn. Michael Feischl and Univ.Prof. Dipl.-Math. Dr.techn. Dirk Praetorius. Numerics of Partial Differential Equations: Stationary Problems, 2021. Hierbei handelt es sich um ein Vorlesungsskript, das im Rahmen der VO Numerik partieller Differentialgleichungen: stationäre Probleme an der Technischen Universität Wien den Studierenden zu Verfügung gestellt wurde. Diese Version wurde von Associate Prof. Dipl.-Math. Dr.rer.nat. Lothar Nannen modifiziert. Version: 19. Jänner 2021.

[Jü20] Univ.Prof. Dr.rer.nat. Ansgar Jüngel. Partielle Differentialgleichung - Vorlesungsmanuskript, 2020. Hierbei handelt es sich um ein Vorlesungsskript, das im Rahmen der VU Partielle Differntialgleichungen an der Technischen Universität Wien den Studierenden zur Verfügung gestellt wurde. Version: November 2020
Erhältlich unter: https://www.asc.tuwien.ac.at/~juengel/ → Teaching → Lecture Notes.

[Nan18] Associate Prof. Dipl.-Math. Dr.rer.nat.Lothar Nannen. Eigenwertprobleme, 2018. Hierbei handelt es sich um ein Vorlesungsskript, das im Rahmen dieses Seminars den Verfasser*innen dieser Arbeit zur Verfügung gestellt wurde. Version: 16. September 2020.