WEB-Approximation elliptischer Eigenwertprobleme

Von der Fakultät Mathematik und Physik der Universität Stuttgart zur Erlangung der Würde eines Doktors der Naturwissenschaften (Dr. rer. nat.) genehmigte Abhandlung

vorgelegt von

Martina Pfeil

aus Waiblingen

Hauptberichter:	Prof. Dr. Klaus Höllig
Mitberichter:	Prof. Dr. Ulrich Reif
Tag der Einreichung:	12. Juni 2007
Tag der mündlichen Prüfung:	19. Juli 2007

IMNG, Universität Stuttgart

2007

Inhaltsverzeichnis

Symbolverzeichnis

Δ	Laplace-Operator
λ_ℓ	ℓ-ter exakter Eigenwert
λ_ℓ^h	ℓ-ter approximierter Eigenwert
$(\,\cdot\,,\,\cdot\,)$	Skalarprodukt
$\mathbb{N}$, $\mathbb{Z}$, $\mathbb{R}$	natürliche, ganze und reelle Zahlen
∇	Gradient
$\Omega \subset \mathbb{R}^n$	beschränktes und zusammenhängendes Gebiet
$\overline{\Omega}$	Abschluß des Gebietes Ω
$\partial\Omega$	Rand des Gebietes Ω, Seite 11
$\preceq$, $\asymp$, $\succeq$	Ungleichungen bis auf Konstanten
A, A_h	Massenmatrix, Seite 20
B, B_h	Steifigkeitsmatrix, Seite 20
b^n	uniformer B-Spline vom Grad n, Seite 24
$b^n_{k,h}$	Kardinal-Spline vom Grad n bei Gitterweite h, Seite 25
$C^\ell(\Omega)$	Menge der ℓ-fach stetig differenzierbaren Funktionen
$C_0^\infty(\Omega)$	Menge aller Funktionen aus C^∞ mit kompaktem Träger in Ω
$e_{i,j}$	Erweiterungskoeffizient, Seite 30

h	Gitterweite, Seite 25
$H^n(\Omega)$	Sobolev-Raum der Ordnung n über Ω, Seite 15
$H_0^n(\Omega)$	Vervollständigung von $C_0^\infty(\Omega)$ bezüglich der Sobolev-Norm der Ordnung n, Seite 15
I	Einheitsmatrix
$L_2(\Omega)$	quadratisch integrierbare Funktionen, Seite 14
P	WEB-Standard-Projektor
$w\mathbb{B}_h^n$	Raum der gewichteten Spline-Funktionen mit Grad n und Gitterweite h, Seite 28
w	Gewichtsfunktion, Seite 28
$w^e\mathbb{B}_h^n$	Raum der gewichteten und erweiterten Spline-Funktionen mit Grad n und Gitterweite h, Seite 30

Das Eigenwertproblem in der Praxis

Die Lösung des verallgemeinerten Eigenwertproblems

$$Mu = \lambda Lu$$

mit großen Matrizen M und L ist für viele Fachbereiche relevant. Im Folgenden werden ein paar Beispiele aufgeführt, bei denen es eine Rolle spielt.

Im mathematischen Bereich benötigt man die Eigenwerte bei der Stabilitätsanalyse von Differentialgleichungen erster Ordnung.

In der Physik findet man dieses Eigenwertproblem in der Schwingungsgleichung, der Wellengleichung oder der Transportgleichung wieder, siehe [BW83].

Als Beispiel wird die homogene Wellengleichung im Zweidimensionalen genauer betrachtet.
Man untersucht die Differentialgleichung

$$\begin{aligned} \frac{1}{c^2} u_{tt} &= u_{xx} + u_{yy} \quad \text{in } \Omega \\ u &= 0 \quad \text{auf } \partial\Omega \end{aligned}$$

auf einem beschränkten Gebiet Ω und erhält mit dem Produktansatz

$$u(x, y, t) = f(x, y)g(t)$$

nach Umformungen die Gleichung

$$\frac{1}{c^2} \frac{g_{tt}}{g} = \frac{f_{xx} + f_{yy}}{f} .$$

Diese besitzt nur für eine Konstante $-\lambda \in \mathbb{R}_-$ eine Lösung. Dadurch bekommt man die beiden Differentialgleichungen

$$g_{tt} = -\lambda c^2 g\,, \tag{1}$$
$$f_{xx} + f_{yy} = -\lambda f\,. \tag{2}$$

Die zwei entstandenen Differentialgleichungen beschreiben ein Eigenwertproblem, welches in manchen Fällen nur numerisch gelöst werden kann.

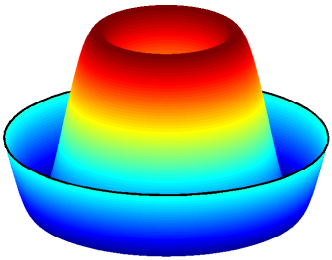
Man kann die Einheitskreisscheibe als beschränktes Gebiet nehmen. In der Praxis kann dies zum Beispiel die wellenförmige Bewegung eines Trommelfells mit Radius eins sein. In diesem Fall hat die Gleichung (2) die Lösungen

$$f_{k,n}(r,\varphi) = J_{|k|}(\mu_{|k|,n} r) e^{\mathrm{i}k\varphi}, \quad k \in \mathbb{Z},\, n \in \mathbb{N}\,,$$

mit $\mu_{|k|,n} > 0$, den Nullstellen der Bessel-Funktionen $J_{|k|}$. Die entsprechenden Eigenwerte sind $\lambda_{k,n} = \mu^2_{|k|,n}$. Für die Einheitskreisscheibe hat die Gleichung (1) die Lösungen

$$g(t) = a\cos(c\sqrt{\lambda}t) + b\sin(c\sqrt{\lambda}t)$$

mit passenden Konstanten a und b für die jeweiligen Anfangsbedingungen. Man erhält z. B. für $a = 1$ beim Zeitpunkt $t = 0$ folgende Lösung:



Dabei wurde für dieses Bild $k = 3$ und $n = 2$ gewählt, also eine Eigenfunktion genommen, die der zweiten Nullstelle der Bessel-Funktion J_3 entspricht.

Ein Anwendungsgebiet aus der Quantenmechanik ist die zeitunabhängige Schrödingergleichung

$$-\frac{\hbar}{2m}\Delta u(\vec{r}) + V(\vec{r})u(\vec{r}) = Eu(\vec{r})$$

mit der Nebenbedingung

$$\int_{\mathbb{R}^3} |u(\vec{r})|\, dx\, dy\, dz = 1\,.$$

Diese beschreibt ein skalares, nichtrelativistisches Teilchen mit der Masse m, welches sich stationär mit der Energie E in ein Potentialfeld $V(\vec{r})$ bewegt. $|u(\vec{r})|^2$ gibt die Aufenthaltswahrscheinlichkeitsdichte des Teilchens an der Stelle $\vec{r}$ an.

Die Schwingungsgleichung tritt auch bei Gebäuden oder Tragwerken auf. Dabei spielen vor allem die Eigenfrequenzen eine wichtige Rolle. Beim Brückenbau, vgl. [Bac04], sind Schwingungsprobleme ein wichtiger Gesichtspunkt. Dabei werden besonders die Fußgängerbrücken immer schwingungsanfälliger, da man heutzutage schlankere Konstruktionen und größere Spannweiten als bei älteren Brücken hat. Als spektakuläres Beispiel kann die Tacoma-Narrows-Brücke im amerikanischen Bundesstaat Washington genannt werden, die damals drittlängste Hängebrücke der Welt. Diese stürzte 1940 nur wenige Monate nach ihrer Fertigstellung ein. Solche kostspieligen Misserfolge will man in Zukunft natürlich vermeiden. Dies gilt auch für ähnlich konstruierte Bauwerke, wie weitgespannte Treppen, Schiffslandestege, Überlandleitungen und hohe Schornsteine.

Schwingungsphänomene spielen auch im Luftverkehr ([HÖuP98]) eine immer wichtigere Rolle. Hierbei werden aufgrund von Kapazitätsengpässen übergroße Transportflugzeuge benötigt. Durch die Größenverhältnisse treten sehr elastische, schwach gedämpfte Strukturen auf, deshalb wird der aeroelastische Gesichtspunkt für die Entwicklung immer relevanter. Bei diesen großen Flugzeugen besteht die Gefahr, dass die niedrigen Eigenfrequenzen der Strukturschwingung und das klassische Flatterverhalten den Flugkomfort und die Flugsicherheit negativ beeinflussen. Aus Kostengründen müssen die wesentlichen aeroelastischen Einflüsse schon bei Beginn der Planung eine Rolle spielen, bei der ein Teil die Eigenwertberechnung ist.

Aufbau der Arbeit

Im ersten Kapitel wird die Finite-Elemente-Methode eingeführt. Dabei werden die Grundlagen erklärt, die zur Lösung des Eigenwertproblems von Interesse sind. Das Eigenwertproblem wird mit speziellen Elementen, den WEB-Splines, approximiert. Die Elemente und deren Eigenschaften werden in Kapitel 2 erklärt. Es ist relevant, ob die Approximationsordnung der Eigenwerte und Eigenfunktionen dieselbe wie bei den gewöhnlichen Finite-Elementen ist. Der Beweis steht in Kapitel 3. Im zweiten Teil dieses Kapitels erkennt man die richtige Approximationsordnung in einigen Abbildungen. Die Eigenwerte wurden mit dem Lanczos-Algorithmus berechnet. Dieser wird in Kapitel 4 ausführlich eingeführt und die Verbesserungen und Probleme des Algorithmus aufgezeigt. Für die im Lanczos-Algorithmus notwendige Invertierung wird der Multigrid-Löser angewendet. Hier gibt es unterschiedliche Methoden, den Startwert zu berechnen. Auch die Abfolge beim Gitterwechsel kann auf verschiedenen Wegen dargestellt werden. Die Methoden werden in Kapitel 5 behandelt. Im nächsten Kapitel wird auf die Implementation des Eigenwertproblems eingegangen. Auch hier wurden mehrere Gebiete durchgerechnet. Das Kapitel 7 beschäftigt sich mit der Wellenbewegung von Wasser. Dabei steht die Schnelligkeit der Berechnung im Vordergrund, deshalb wurden nur lineare Splines genommen. Im Abschnitt 7.6 werden Wasserwellen für ein Gebiet berechnet, welches den größten Teil der Ostsee widerspiegeln soll.
Im Anhang A befinden sich die Formeln, die für den Beweis aus Kapitel 3 benötigt werden.

Danksagung

An dieser Stelle möchte ich allen danken, die mir mein Studium und meine Promotion ermöglichten.
Mein ganz besonderer Dank gilt Prof. Dr. Klaus Höllig für die Überlassung und Unterstützung bei der Ausfertigung dieser Arbeit.
Ebenfalls danke ich Prof. Dr. Ulrich Reif für die Bereitschaft, die vorliegende Arbeit zu begutachten.
Allen Mitarbeitern des IMNG danke ich für die gute Zusammenarbeit und die hilfreichen Ratschläge.
Außerdem danke ich meiner Familie, die mich während des Studiums und der Promotion jederzeit unterstützte.

Kapitel 1

Einführung

Im Kapitel 1 wird auf die allgemeine Theorie der Finite-Elemente eingegangen. Die Definitionen und Aussagen orientieren sich dabei an dem Buch von [Bra03]. Der erste Abschnitt ist eine kurze Einführung in partielle Differentialgleichungen, danach wird die Finite-Elemente-Methode vorgestellt.

1.1 Typeneinteilung

Auf einem offenen, zusammenhängenden und beschränkten Gebiet $\Omega \subset \mathbb{R}^n$, $n \in \mathbb{N}$ mit Gebietsrand $\partial\Omega = \overline{\Omega}\backslash\Omega$ kann man eine Differentialgleichung zweiter Ordnung

$$-\sum_{i,k=1}^{n} a_{ik}(x)\frac{\partial^2 u}{\partial x_i \partial x_k} + \sum_{i=1}^{n} b_i(x)\frac{\partial u}{\partial x_i} + c(x)u = f(x) \text{ für } x \in \Omega \quad (1.1)$$

mit $f : \Omega \mapsto \mathbb{R}$ in verschiedene Typen einteilen. Falls die Funktionen a_{ik}, b_i und c unabhängig von x sind, hat man eine lineare Differentialgleichung mit konstanten Koeffizienten. Für zweimal stetig differenzierbare Funktionen u gilt

$$\frac{\partial^2 u}{\partial x_i \partial x_k} = \frac{\partial^2 u}{\partial x_k \partial x_i},$$

daher ist in diesem Fall die Matrix

$$A(x) = (a_{ik}(x))$$

symmetrisch. Man geht im Folgenden davon aus, dass dies der Fall ist.

Definition 1.1: *Typeneinteilung*

1. *Die Gleichung (1.1) heißt elliptisch im Punkt x, falls $A(x)$ invertierbar ist und alle Eigenwerte dasselbe Vorzeichen haben.*
2. *Die Gleichung (1.1) heißt hyperbolisch im Punkt x, falls $A(x)$ invertierbar ist und genau ein Eigenwert ein anderes Vorzeichen hat als alle anderen.*
3. *Die Gleichung (1.1) heißt parabolisch im Punkt x, falls der Rang von $A(x)$ $n-1$ ist und alle Eigenwerte ungleich null dasselbe Vorzeichen haben.*

Einfache Beispiele für den jeweiligen Typ sind:

1. elliptisch: $-\Delta u = f$ (Poisson-Gleichung)
2. hyperbolisch: $u_{tt} - c^2 \Delta u = f$ (Wellengleichung)
3. parabolisch: $u_t - c^2 \Delta u = f$ (Wärmeleitungsgleichung)

Damit eine Differentialgleichung überhaupt eindeutig lösbar sein kann, müssen Anfangsbedingungen für $t = t_0$ bzw. Randbedingungen für $\partial\Omega$ für die gesuchte Funktion u vorgegeben sein. Dabei unterscheidet man drei Arten von Randbedingungen

1. Dirichlet-Randbedingung: $u = k$ für $x \in \Gamma_D$,
2. Neumann-Randbedingung: $\frac{\partial u}{\partial n} = g$ für $x \in \Gamma_N$,
3. Robin-Randbedingung: $u + a\frac{\partial u}{\partial n} = h$ für $x \in \Gamma_R$,

mit $\partial\Omega = \Gamma_D + \Gamma_N + \Gamma_R$, $\Gamma_D \cap \Gamma_N = \emptyset$, $\Gamma_D \cap \Gamma_R = \emptyset$ und $\Gamma_N \cap \Gamma_R = \emptyset$. k, g, h und a sind Funktionen, die von x abhängen.

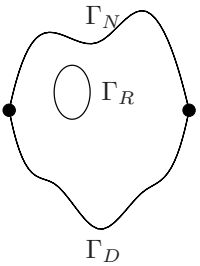


Abbildung 1.1: Gebietsrand mit verschiedenen Randbedingungen

Man kann, wie in Abbildung 1.1 gezeigt, auch mehrere Randbedingungen auf unterschiedlichen Abschnitten von $\partial\Omega$ haben.
Um überhaupt eine sinnvolle Lösung der Differentialgleichung zu bekommen, muss das Differentialgleichungsproblem sachgemäß gestellt sein.

Definition 1.2:
Ein Problem heißt sachgemäß gestellt, wenn eine Lösung existiert, diese eindeutig ist und stetig von den vorgegebenen Daten abhängt. Andernfalls heißt das Problem schlecht gestellt.

Erfüllt eine Funktion die gegebene Differentialgleichung (1.1) und nimmt auf dem Rand die vorgegebenen Randbedingungen an, so nennt man dies eine klassische Lösung, wenn sie bei Dirichlet-Randbedingungen in $C^2(\Omega) \cap C^0\left(\overline{\Omega}\right)$ oder bei Neumann-Randbedingungen in $C^2(\Omega) \cap C^1\left(\overline{\Omega}\right)$ enthalten ist.

1.2 Finite-Elemente-Methode

Die Methode der Finite-Elemente ist ein numerisches Verfahren zum Lösen partieller Differentialgleichungen, die z. B. aus dem Maschinenbau, dem Bauwesen oder der Luftfahrttechnik kommen. Dabei entsteht ein mathematisches, numerisch effizientes Modell, z. B. aus einem physikalischen Problem. Dies kann manchmal mit Hilfe bestimmter Eigenschaften, wie der Geometrie oder bestimmten Randbedingungen, vereinfacht werden. Für viele dieser Probleme kann eine klassische Lösung der partiellen Differentialgleichung kaum gefunden werden, deshalb greift man auf ein numerisches Näherungsverfahren, die Finite-Elemente-Methode, kurz FEM, zurück.
Man kann kein exaktes Datum zur Entstehung der FEM angeben. Die Arbeit von Schellbach [Sch51] zur Lösung eines Minimalflächenproblems im Jahr 1851 ist der FEM schon sehr ähnlich. Courant beschrieb 1943 [Cou43] in seiner Arbeit „Variational methods for the solution of problems of equilibrium and vibrations“ ein Verfahren mit Ansatzfunktionen mit lokalen Trägern. Dies fand jedoch keine praktische Bedeutung, da dabei ein Gleichungssystem mit vielen Unbekannten entsteht, welches ohne elektronische Hilfsmittel nicht lösbar ist. Nachdem die entsprechenden Rechner zur Verfügung standen, wurde die Finite-Elemente-Methode in der Strukturmechanik von M. J. Turner, Ray W. Clough, W. C. Martin und L. J. Topp [TCMT56] angewendet. Sie berechneten Flugzeugtragflügel bei Boeing in Seattle, USA. In den sechziger Jahren trugen Arbeiten von Agyris [Arg57] und Zienkiewicz [Zie67] zur rasanten Entwicklung

der FEM bei.
Hier wird das Eigenwertproblem

$$\begin{aligned} -\Delta u &= \lambda u \quad \text{in } \Omega \\ u &= 0 \quad \text{in } \partial\Omega \end{aligned} \tag{1.2}$$

genauer untersucht.
Für die weitere Finite-Elemente-Berechnung werden die Sobolev-Räume benötigt. Die Sobolev-Räume werden auf dem Funktionenraum $L_2(\Omega)$ definiert. Im $L_2(\Omega)$-Raum liegen Funktionen, deren Quadrat über das Gebiet Lebesgue-integrierbar sind, also

$$v(x) \in L_2(\Omega) \quad \Longleftrightarrow \quad \int_\Omega v(x)^2 \, dx < \infty$$

ist. Damit kann mit Funktionen $u(x), v(x) \in L_2(\Omega)$ über ein Gebiet Ω ein Skalarprodukt mit zugehöriger Norm definiert werden:

$$(u(x), v(x))_0 = (u(x), v(x))_{L_2(\Omega)} = \int_\Omega u(x)v(x) \, dx \tag{1.3}$$

$$\|u(x)\|_0 = \|u(x)\|_{L_2(\Omega)} = \sqrt{(u(x), u(x))_{L_2(\Omega)}} \tag{1.4}$$

Der $L_2(\Omega)$-Raum ist mit der Norm (1.4) vollständig und bildet mit dem Skalarprodukt (1.3) einen Hilbert-Raum. Außerdem benötigt man noch die Definition der schwachen Ableitung.

Definition 1.3: *Schwache Ableitung*
$u \in L_2(\Omega)$ *besitzt in* $L_2(\Omega)$ *die schwache Ableitung* $v = \partial^\alpha u$, *falls* $v \in L_2(\Omega)$ *und*

$$(\phi, v)_0 = (-1)^\alpha (\partial^\alpha \phi, u)_0 \quad \forall \phi \in C_0^\infty(\Omega)$$

ist.

Es liegen die Funktionen in C_0^∞, die beliebig oft differenzierbar sind und nur auf einer kompakten Teilmenge von Ω Werte ungleich Null annehmen.
Ist eine Funktion im klassischen Sinn differenzierbar, dann existiert die schwache Ableitung und beide stimmen überein. Für die weitere Finite-Elemente Einführung sind die Sobolev-Räume ein wichtiger Bestandteil. Daher werden diese nun eingeführt.

Lemma 1.1: *Sobolev-Räume*
Der Sobolev-Raum $H^k(\Omega)$ auf einem Gebiet $\Omega \in \mathbb{R}^n$ besteht aus Funktionen, für die

$$\|u\|_{k,\Omega}^2 = \sum_{|\alpha| \le k} \int_\Omega (\partial^\alpha u)^2 < \infty$$

gilt. Dabei ist $\alpha = (\alpha_1, \dots \alpha_n)$ mit $|\alpha| = \sum_{i=1}^{n} \alpha_i$ und $\partial^\alpha u = \frac{\partial^{|\alpha|} u}{\partial x_1^{\alpha_1} \dots \partial x_n^{\alpha_n}}$.
$\|\cdot\|_{k,\Omega}$ ist die Sobolev-Norm und das Skalarprodukt ist mit Funktionen $u, v \in H^k(\Omega)$ mit

$$(u,v)_{k,\Omega} = \sum_{|\alpha| \le k} \int_\Omega \partial^\alpha u \cdot \partial^\alpha v$$

definiert.
Damit erhält man auch die Sobolev-Halbnorm mit $u \in H^k(\Omega)$

$$|u|_{k,\Omega}^2 = \sum_{|\alpha| = k} \int_\Omega (\partial^\alpha u)^2 \ .$$

Sind Nullrandbedingungen wie in der Gleichung 1.2 vorgegeben, dann verwendet man in diesem Fall H_0^k-Räume. Der Raum H_0^k ist folgendermaßen definiert:

$$H_0^k = \left\{ v \in H^k \ ; \text{ es gibt } v_n \in C_0^\infty \text{ mit } \|v - v_n\|_k \to 0 \text{ für } n \to \infty \right\}$$

Die schwache Formulierung des Eigenwertproblems

$$-\Delta u = \lambda u \,,$$

mit der bei den Finite-Elementen gerechnet wird, bekommt man mit einigen Umformungen. Als erstes multipliziert man mit einer Testfunktion v, integriert über das Gebiet Ω

$$-\int_\Omega v \Delta u = \lambda \int_\Omega vu$$

und erhält dann mit dem ersten Integralsatz nach Green

$$\int_\Omega \nabla v \nabla u - \int_{\partial\Omega} \frac{\partial u}{\partial n} v = \lambda \int_\Omega vu \,.$$

Hat man nun eine der beiden folgenden Randbedingungen

$$u = 0 \text{ auf } \partial\Omega \quad \text{(Dirichlet-Randbedingung)} \tag{1.5}$$

$$\frac{\partial u}{\partial n} = 0 \text{ auf } \partial\Omega \quad \text{(Neumann-Randbedingung)}\,, \tag{1.6}$$

dann wird $\int\limits_{\partial\Omega} \frac{\partial u}{\partial n} v = 0$ und braucht nicht berücksichtigt und daher auch nicht berechnet zu werden. Mit den Randbedingungen (1.5) und (1.6) bekommt man die Variationsformulierung

$$a\,(u,v) = \lambda b\,(u,v) \qquad \begin{array}{l} \forall v \in H_0^1\,(\Omega) \text{ bei Dirichlet-RB} \\ \forall v \in H^1\,(\Omega) \text{ bei Neumann-RB} \end{array} \tag{1.7}$$

mit

$$a\,(u,v) = \int\limits_{\Omega} \nabla v \nabla u$$

$$b\,(u,v) = \int\limits_{\Omega} vu\,.$$

Ist u eine Lösung von (1.7), so spricht man von einer schwachen Lösung der Differentialgleichung mit den vorgegebenen Randbedingungen. Dabei ist

$$a: \quad H \times H \mapsto \mathbb{R} : (u,v) \mapsto a(u,v) \qquad \forall u,\, v \in H$$

in einem Hilbert-Raum H eine Bilinearform, die bestimmte Eigenschaften haben kann.

Definition 1.4: *Für einen Hilbert-Raum H ist eine Bilinearform $a : H \times H \mapsto \mathbb{R}$ stetig, wenn ein $c_1 > 0$ existiert, für das gilt*

$$|a(u,v)| \leq c_1 \|u\| \cdot \|v\| \qquad \forall u,\, v \in H\,.$$

Hat man eine symmetrische stetige Bilinearform, so heißt sie elliptisch, wenn ein $c_2 > 0$ existiert mit

$$a(v,v) \geq c_2 \|u\|^2 \qquad \forall v \in H\,.$$

Die Elliptizität lässt sich im $H_0^1(\Omega)$-Raum, also bei Dirichlet-Nullrandbedingungen, mit der Poincaré-Friedrichs-Ungleichung beweisen. Allerdings ist dies im $H^1(\Omega)$-Raum, den man bei Neumann-Randbedingungen hat, nicht der Fall. Die Elliptizität ist für die Eindeutigkeit einer Lösung relevant. Die Beschränktheit bekommt man mit Hilfe der Cauchy-Schwarz-Ungleichung.

Satz 1.1: *Poincaré-Friedrichs-Ungleichung*
Es existiert eine Konstante c mit

$$\|v\|_{k-1} \leq c\,|v|_k \qquad \forall v \in H_0^k(\Omega)\ .$$

Demzufolge sind auf dem $H_0^k(\Omega)$ die Norm $\|\cdot\|_k$ und die Halbnorm $|\cdot|_k$ äquivalent.

Mit Hilfe des Satzes von Lax-Milgram erhält man dann die eindeutige Lösbarkeit des Variationsproblems.

Satz 1.2: *Lax-Milgram*
Ist a eine elliptische und beschränkte Bilinearform und λ ein beschränktes lineares Funktional auf einem Hilbertraum H, dann hat das Variationsproblem für jeden abgeschlossenen Unterraum V von H

$$a(u,v) = \lambda(v)\,, \qquad v \in V\,,$$

eine eindeutige Lösung $u \in V$.

1.3 Triangulation und Basisfunktionen

Für den weiteren Verlauf benötigt man einen endlich dimensionalen Teilraum. Das Gebiet Ω wird in endlich viele Teilgebiete

$$\Omega = \bigcup_{\ell=1}^{m} \Omega_\ell$$

unterteilt, z. B. Dreiecke oder Parallelogramme im Zweidimensionalen, Tetraeder oder Quader im Dreidimensionalen. Diese Teilgebiete nennt man Elemente und die Zerlegung Triangulation. Dabei muss die Zerlegung nicht regelmäßig sein, aber es müssen bestimmte Eigenschaften erfüllt sein.

Definition 1.5:
Eine Triangulation $\mathcal{T} = \{\Omega_1\,,\ \Omega_1\,,\ \ldots \Omega_m\}$ *in Dreieck- oder Viereckelemente heißt zulässig, wenn:*

1. $\overline{\Omega} = \bigcup_{\ell=1}^{m} \Omega_\ell$

2. *Falls* $\Omega_i \cap \Omega_j$ *ein Punkt ist, so muss dies ein Eckpunkt von* Ω_i *und* Ω_j *sein.*

3. *Falls* $\Omega_i \cap \Omega_j$ *mehr als ein Punkt ist, so muss dies eine Kante von* Ω_i *und* Ω_j *sein.*

Dabei kann es Probleme geben, den Gebietsrand, wie man in Abbildung 1.2 sieht, genau darzustellen. Die Vernetzung ist oft teuer und aufwendig, damit die Approximationsgüte nicht verloren geht. Diese Probleme hat man bei den WEB-Splines nicht, da sich der Gebietsrand mit einer Gewichtsfunktion exakt darstellen lässt und man regelmäßige Gitter wählt.

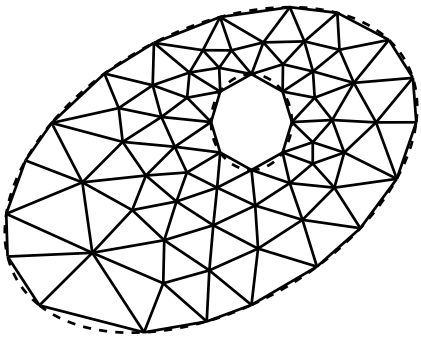


Abbildung 1.2: Triangulierung mit Dreiecken

Damit es rechenbar ist, wird aus dem unendlich-dimensionalen Ansatzraum ein endlich-dimensionaler Teilraum ausgewählt. Man erhält die Lösung u^h, welche die exakte Lösung u approximiert, mit

$$u^h = \sum_{i=1}^{n} c_i \varphi_i\,.$$

Dabei muss im Fall (1.5) φ_i für alle $i = 1, \ldots, n$ auf dem Rand Null sein. Die Basisfunktionen φ_i sollten nur auf wenigen Teilgebieten $\Omega_\ell \neq 0$ sein, da dies die spätere Integration vereinfacht. Für den Ansatzraum können WEB-Splines, die in Abschnitt 2.4 eingeführt werden, oder auch andere Basisfunktionen genommen werden. Bei den WEB-Splines gilt dann $\Psi = w^e \mathbb{B}_h^n$. Es werden meist stückweise stetige Polynome vom Grad t genommen, im Zweidimensionalen z. B.

$$\mathcal{P}_t = \left\{ u(x,y) = \sum_{\substack{i+k \leq t \\ i\,,\, k \geq 0}} c_{i,k} x^i y^k \quad \text{mit} \quad c_{i,k} \in \mathbb{R} \right\} .$$

Ein einfaches Beispiel sind die Lagrange-Finite-Elemente auf Dreiecken. Bei den linearen Lagrange-Finite-Elementen ist jede Basisfunktion auf jedem Dreieck linear. Sie sind über die Knoten definiert und jede Basisfunktion ist an einem Knoten eins und auf allen anderen Knoten Null. Da die Lagrange-Finite-Elemente über Funktionswerte an den Knoten definiert sind, werden sie auch nodale Basisfunktionen genannt. Bei anderen Basisfunktionen können auch Werte von Ableitungen an den Kanten oder Ecken eine Rolle spielen. Nimmt man das Normaldreieck, dann sind die drei linearen Lagrange-Basisfunktionen auf dem Dreieck

$$\varphi_1 = 1 - x - y\,, \quad \varphi_2 = x\,, \quad \varphi_3 = y\,.$$

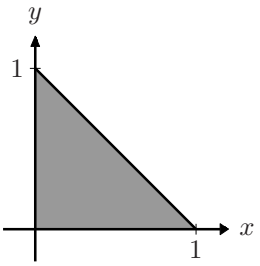


Abbildung 1.3: Normaldreieck

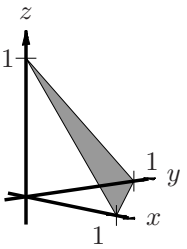


Abbildung 1.4: Lagrange-Basisfunktion $\varphi_1 = 1 - x - y$

Mit den Randbedingungen (1.5) oder (1.6) bekommt man mit dem endlichen Ansatzraum und den Testfunktionen $\varphi_i\,,\ i = 1, \ldots, n$ die Variationsformulierung

$$\sum_{i=1}^{n} c_i \int_\Omega \nabla\varphi_i \nabla\varphi_k = \lambda \sum_{i=1}^{n} c_i \int_\Omega \varphi_i \varphi_k \quad k = 1, \ldots, n\,.$$

Damit erhält man ein verallgemeinertes Eigenwertproblem in endlicher Dimension

$$Ax = \lambda Bx$$

mit den Matrizen $A, B \in \mathbb{R}^{n\times n}$ und den Einträgen

$$a_{i,j} = \int_\Omega \nabla\varphi_i \nabla\varphi_j \,, \quad x_i = c_i \text{ und } b_{i,j} = \int_\Omega \varphi_i\varphi_j \,.$$

A wird dabei Massenmatrix und B Steifigkeitsmatrix genannt. Die Integrale werden durch Approximationsformeln, z. B. Gauss-Quadraturformeln, numerisch berechnet. In den Abbildungen 1.5 und 1.6 wurden 3×3 Gauss-Punkte in jede Zelle gelegt, bei einem Schnittpunkt mit dem Rand muss man die Zelle unterteilen.

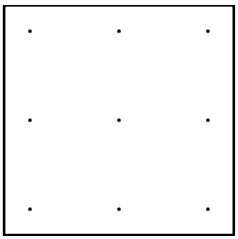


Abbildung 1.5: Gauss-Punkte in Zelle, die nicht vom Rand geschnitten wird

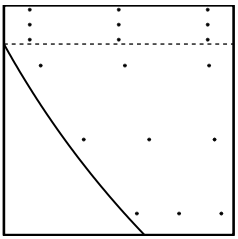


Abbildung 1.6: Gauss-Punkte in Zelle, die vom Rand geschnitten wird

Andere Beispiele für Finite-Elemente auf Dreiecken und Vierecken sind in der Abbildung 1.7 dargestellt.
Da anfangs vorausgesetzt wurde, dass die Basisfunktionen fast überall Null sind, entstehen dünn besetzte Matrizen. Die berechneten Gleichungssysteme werden in den folgenden Kapiteln weiter untersucht.
Die Finite-Elemente-Approximation u^h ist die beste Approximation im Unterraum $u^h \in \Psi$. Damit erhält man die Galerkin-Orthogonalität

$$a\left(u - u^h, v^h\right) = 0 \qquad \forall v^h \in \Psi \,,$$

die im Abschnitt 3.1 öfters benötigt wird.

● Festlegung des Funktionswertes

◉ Festlegung des Funktionswertes und der 1. und 2. Ableitung

⊥ Festlegung der Normalenableitung

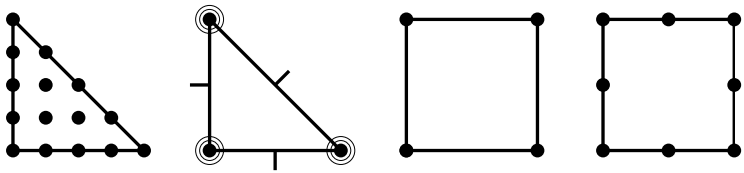


Abbildung 1.7: Beispiel für gebräuchliche Finite-Elemente

Die Approximationsordnung bei den gewöhnlichen Finite-Elementen, also z. B. bei den Lagrange-Elementen, ist für die Eigenwerte bei Grad k

$$\lambda_\ell^h - \lambda_\ell \leq ch^{2k}\lambda_\ell^{k+1}\,,$$

d. h. je größer ein Eigenwert ist, desto schlechter lässt er sich approximieren. Es gilt immer, dass der approximierte Eigenwert λ_ℓ^h größer als der exakte Eigenwert λ_ℓ ist. Für die Eigenfunktionen gilt bei einem einfachen Eigenwert die Fehlerabschätzung

$$\left\|u_\ell - u_\ell^h\right\|_0 \leq ch^{k+1}\lambda_\ell^{(k+1)/2}\,.$$

Bei mehrfachen Eigenwerten können die unterschiedlichen Eigenfunktionen in einem Eigenraum orthogonal gewählt werden, damit die Abschätzung immer noch gilt. Im Kapitel 3.1 wird gezeigt, dass die Eigenwerte mit der WEB-Spline-Basis dieselbe Approximationsordnung besitzen.

Kapitel 2

B-Splines

Bei der in den folgenden Kapiteln bearbeiteten Finite-Elemente-Methode basieren die Basisfunktionen auf Splines, also stückweise polynomialen Funktionen. Im Hinblick auf den Multigrid-Löser werden nur uniforme Knoten betrachtet, da dabei wesentliche Vorteile bestehen.

Splines spielen eine wichtige Rolle in der Approximation und geometrischen Modellierung. Die Arbeit von Schönberg [Sch46] zeigt, dass Splines gute Approximationseigenschaften besitzen. Nach den Ergebnissen von Boor [dB72] wurden Splines für viele Anwendungen interessant. Seine Algorithmen [dB77] bilden heute noch oft die Basis für viele angewendete Spline-Algorithmen. Ein weiterer wichtiger Punkt sind die Arbeiten von Bézier [B66́], [B67́] und [B72́]. Diese zeigen, dass Bernstein-Polynome gute Eigenschaften zur Beschreibung geometrischer Kurven und Flächen besitzen. Ähnliche Ergebnisse wurden durch Casteljau [dC59] erhalten, allerdings erreichten diese nie denselben Bekanntheitsgrad. Die Ergebnisse von Bézier und Casteljau verallgemeinerte man bald für Splines. Dabei spielte das Einfügen weiterer Knoten, nach den Arbeiten von Böhm [Böh80] und Cohen et al. [CLR80], eine wesentliche Rolle. Infolgedessen wurden B-Splines immer öfters in der numerischen Approximation und der geometrischen Modellierung verwendet.

Die Lösung des Eigenwertproblems mit Hilfe von B-Splines wurde schon in der Arbeit von Mößner [Mö ß06] untersucht.

2.1 B-Splines

Zur Aufstellung der Finite-Elemente-Matrizen müssen nun geeignete lokale Basisfunktionen definiert werden. Dabei gibt es für die B-Splines, eine Untermenge der Splines, die folgenden zwei Definitionsmöglichkeiten.

Definition 2.1: *B-Splines*
Ein uniformer B-Spline b^n vom Grad n ist durch die Rekursion

$$b^n(x) = \int_{x-1}^{x} b^{n-1}$$

definiert, beginnend mit der charakteristischen Funktion b^0 des Einheitsintervalls $[0, 1)$. Äquivalent dazu ist

$$\frac{d}{dx} b^n(x) = b^{n-1}(x) - b^{n-1}(x-1)$$

mit $b^n(0) = 0$.

Das entsprechende Integral der charakteristischen Funktion b^0 ergibt die stückweise lineare Hut-Funktion b^1

$$b^1(x) = \begin{cases} x & \text{für } 0 \le x \le 1 \\ 2 - x & \text{für } 1 \le x \le 2 \\ 0 & \text{sonst.} \end{cases}$$

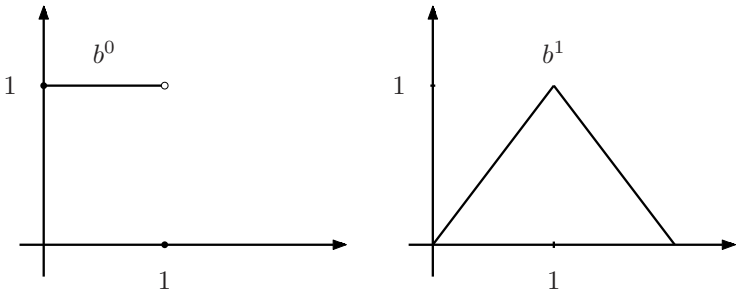


Abbildung 2.1: B-Splines vom Grad Null und Eins

Der B-Spline hat dabei grundlegende Eigenschaften:

- Positiver und lokaler Träger: $b^n > 0$ auf $(0, n+1)$, $b^n = 0$ sonst.

- Glattheit: b^n ist $(n-1)$-mal stetig differenzierbar mit Unstetigkeitsstellen der n-ten Ableitung an den Knoten $0, 1, \ldots, n+1$.
- Stückweise polynomiale Struktur: b^n ist ein Polynom vom Grad n auf jedem Intervall $[k, k+1]$, $k = 0, 1, \ldots, n$.

Für die Berechnung der Finite-Elemente-Matrizen ist die Definition 2.1 ungeeignet, da bei der Integration zur Aufstellung der benötigten Matrizen, z. B. mit Gauss-Integration, nur mit Werten an fest vorgegebenen Punkten gearbeitet wird. Durch die Formel (2.1) wird die Auswertung der B-Splines an diesen Punkten vereinfacht.

Satz 2.1: *Rekursionsformel*
Der B-Spline b^n ist eine gewichtete Kombination von B-Splines vom Grad $n-1$

$$b^n(x) = \frac{x}{n} b^{n-1}(x) + \frac{n+1-x}{n} b^{n-1}(x-1)\,. \tag{2.1}$$

Bei der Konstruktion von Finite-Elemente-Basen benötigt man transformierte B-Splines. Diese werden durch Transformation der uniformen Standard B-Splines b^n auf das Gitter

$$h\mathbb{Z} : \ldots, -2h, -h, 0, h, 2h, \ldots$$

mit der Gitterweite h definiert. Die resultierenden B-Splines $b^n_{k,h}$ spannen einen Raum von glatten Splines mit uniformen Knoten auf.

Definition 2.2: *Kardinal-Splines*
Für $h > 0$ und $k \in \mathbb{Z}$ sind

$$b^n_{k,h} = b^n\left(\frac{x}{h} - k\right)$$

die B-Splines auf dem Gitter $h\mathbb{Z}$. Ihre Linearkombination

$$\sum_{k\in\mathbb{Z}} c_k b^n_{k,h}$$

wird Kardinal-Spline vom Grad $\leq n$ mit Gitterweite h genannt.

Mit der Transformation $x \mapsto \frac{x}{h} - k$ verschiebt sich der Träger von b^n nach $[k, k+n+1]h$. Außerdem sind auf jedem Gitterintervall $Q = [\ell, \ell+1]h$ genau $n+1$ Splines ungleich Null.

2.2 Subdivision und Skalarprodukt von B-Splines

Beim Multigrid-Löser verwendet man eine Folge von Gittergrößen um eine gewisse Genauigkeit und Schnelligkeit mit minimalem Rechenaufwand zu erreichen. Außerdem ist das Verfeinern der Gitter an schwierigen Stellen gut geeignet, um auch kleinere Teilprobleme zu lösen.

Lemma 2.1: *Subdivision*
Die B-Splines $b^n_{k,h}$ können als Linearkombination von B-Splines mit der Gitterweite $\frac{h}{2}$ ausgedrückt werden

$$b^n_{k,h} = 2^{-n} \sum_{\ell=0}^{n+1} \binom{n+1}{\ell} b^n_{2k+\ell,\frac{h}{2}} \,. \tag{2.2}$$

Als Beispiel zeigt die Abbildung 2.2 den B-Spline $b^2_{k,h}$, der durch B-Splines mit halber Gitterweite $h/2$ erzeugt wird. Dabei gilt

$$b^2_{kh} = \frac{1}{4}\left(b^2_{2k,h/2} + 3b^2_{2k+1,h/2} + 3b^2_{2k+2,h/2} + b^2_{2k+3,h/2}\right) .$$

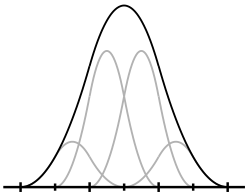


Abbildung 2.2: Subdivision von b^2_{kh}

Bei der Aufstellung der Finite-Elemente-Matrizen kommt das Skalarprodukt zweier B-Splines bzw. deren Gradienten vor. Dabei ist das Skalarprodukt $s^n_{k-\ell}$ zweier B-Splines $b^n_{k,h}$ und $b^n_{\ell,h}$

$$s^n_{k-l} = hb^{2n+1}(n+1+k-\ell)$$

und das Skalarprodukt $d^n_{k-\ell}$ deren Ableitungen

$$d^n_{k-\ell} = h^{-2}\left(2s^{n-1}_{k-\ell} - s^{n-1}_{k-\ell-1} - s^{n-1}_{k-\ell+1}\right) .$$

n	s_0^n	s_1^n	$\dots$			d_0^n	d_1^n	$\dots$		
1	$\frac{2}{3}$	$\frac{1}{6}$				2	-1			
2	$\frac{11}{20}$	$\frac{13}{60}$	$\frac{1}{120}$			1	$-\frac{1}{3}$	$-\frac{1}{6}$		
3	$\frac{151}{315}$	$\frac{397}{1680}$	$\frac{1}{42}$	$\frac{1}{5040}$		$\frac{2}{3}$	$-\frac{1}{8}$	$-\frac{1}{5}$	$-\frac{1}{120}$	
4	$\frac{15619}{36288}$	$\frac{44117}{181440}$	$\frac{913}{22680}$	$\frac{251}{181440}$	$\frac{1}{362880}$	$\frac{35}{72}$	$-\frac{11}{360}$	$-\frac{17}{90}$	$-\frac{59}{2520}$	$-\frac{1}{5040}$

Tabelle 2.1: Skalarprodukte und deren Ableitungen von B-Splines

Einige Skalarprodukte der B-Splines und deren Ableitungen sind in der Tabelle 2.1 für Gitterweite $h = 1$ angegeben, wobei aus der Symmetrie folgt, dass $s_i^n = s_{-i}^n$ und $d_i^n = d_{-i}^n$ ist.

Skalarprodukte von B-Splines mit höheren Ableitungen können mit Hilfe der Ableitungsformel

$$\left(\frac{d}{dx}\right)^{\alpha} b_{k,h}^n(x) = h^{-\alpha} \sum_{\nu=0}^{\alpha} (-1)^{\nu} \binom{\alpha}{\nu} b_{k+\nu,h}^{n-\alpha}$$

berechnet werden.

2.3 Finite-Elemente-Basen

Die einfachste Möglichkeit, univariate B-Splines auf mehrere Variable zu verallgemeinern, ist, Produkte von uniformen B-Splines zu bilden.

Definition 2.3: *Tensorprodukt-B-Splines*
Der m-variate B-Spline $b_{k,h}^n$ vom Grad n_ν in der ν-ten Variablen an der Stelle $(k_1, \dots, k_m)^T h$ und der Gitterweite h ist definiert als

$$b_{k,h}^n(x) = \prod_{\nu=1}^{m} b_{k_\nu,h}^{n_\nu}(x_\nu)$$

mit der Konvention, dass $n_1 = \cdots = n_m$ gilt, falls es nicht anders angegeben wird.

Dabei gelten für die multivariaten B-Splines dieselben Eigenschaften wie für die uniformen B-Splines. Die Aufstellung der Finite-Elemente-Matrizen vereinfacht der nachfolgende Satz.

Satz 2.2: *Partielle Ableitung*
Partielle Ableitungen 1. Ordnung der multivariaten B-Splines $b_{k,h}^{(n_1,\ldots,n_m)}$ *sind Differenzen von B-Splines niedrigeren Grades*

$$\partial^\alpha b_{k,h}^n = h^{-1}\left(b_{k,h}^{n-\alpha} - b_{k+\alpha,h}^{n-\alpha}\right) .$$

Wobei $\alpha = (1,0,\ldots,0)\ ,\ (0,1,0,\ldots,0)\ ,\ \ldots\ ,\ (0,\ldots,0,1)$ *die Einheitsvektoren sind.*

Bei der Konstruktion eines Finite-Elemente-Raumes aus stückweise polynomialen Funktionen wird hier der Splineraum $\mathbb{B}_h^n(\Omega)$ genommen.

Definition 2.4: *Splines*
Der Raum der Splines $\mathbb{B}_h^n(\Omega)$ *auf einem beschränkten Gebiet* $\Omega \subset \mathbb{R}^m$ *besteht aus den Linearkombinationen*

$$\sum_{k\in K} c_k b_{k,h}^n$$

mit den relevanten B-Splines, d. h. die Menge der relevanten Indizes K *enthält alle* $b_{k,h}^n$ *mit* $b_{k,h}^n \neq 0$ *für ein* $x \in \Omega$.

2.4 WEB-Splines

Da bisher der Rand $\partial\Omega$ eines Gebietes Ω nicht berücksichtigt wurde, werden gewichtete Spline-Räume

$$w\mathbb{B}_h^n(\Omega) = \mathrm{span}_{k\in K}\, wb_k^n$$

eingeführt. Dabei müssen gewisse Voraussetzungen an die Gewichtsfunktion w gestellt werden.

Definition 2.5: *Gewichtsfunktion*
Eine Gewichtsfunktion w *von der Ordnung* $n \in \mathbb{N}_0$ *ist auf* $\overline{\Omega}$ *stetig und erfüllt*

$$w(x) \asymp \mathrm{dist}(x,\Gamma)^n\,, \quad x \in \Omega\,,$$

für eine Teilmenge Γ *von* $\partial\Omega$. *Dabei wird angenommen, dass* Γ *ein positives* $(m-1)$*-dimensionales Maß hat und genügend regulär ist, damit die Steigung der Abstandsfunktion begrenzt ist. Falls* w *glatt ist und auf*

ganz $\partial\Omega$ linear ($n = 1$) verschwindet, nennt man w eine Standardgewichtsfunktion.

Dabei ist die Positivität der Gewichtsfunktion innerhalb des Gebietes wesentlich. Falls $w(x) = 0$ für einige $x \in \Omega$ gilt, verschwinden alle Funktionen in dem gewichteten Spline-Raum $w\mathbb{B}$ bei x, und die Approximationsgüte kann nicht mehr erreicht werden.
Für manche Gebiete, wie z. B. für eine Ellipse, ist die Angabe einer Gewichtsfunktion einfach

$$w(x) = 1 - \frac{x^2}{a^2} - \frac{y^2}{b^2}\,.$$

Die Räume $\mathbb{B}$ und $w\mathbb{B}$ liefern die optimale Approximationsgüte, sind aber in Bezug auf die Gitterweite h nicht stabil. Die Instabilität entsteht durch B-Splines mit sehr kleinem Träger im Gebiet Ω und verursacht numerische Probleme für $h \to 0$. Die Stabilitätsprobleme werden dadurch behoben, dass man die B-Splines mit sehr kleinen Träger zu einer bestimmten stabilen Untermenge der Basis $\mathbb{B}$ addiert. Aus diesem Grund findet eine Zellklassifikation statt.

Definition 2.6: *Innere und äußere B-Splines*
Die Gitterzellen $Q = \ell h + [0,1]^m h$ sind in innere, Rand- oder äußere Zellen aufgeteilt, je nachdem, ob $Q \subseteq \overline{\Omega}$ das Innere von Q $\partial\Omega$ schneidet oder ob $Q \cap \Omega = \emptyset$ gilt. Unter den relevanten B-Splines $b_k\,,\ k \in K\,,$ unterscheidet man zwischen inneren B-Splines

$$b_i\,, \quad i \in I\,, \tag{2.3}$$

die mindestens eine innere Zelle Q_i innerhalb ihres Trägers haben, und den äußeren B-Splines

$$b_j\,, \quad j \in J = K \backslash I\,, \tag{2.4}$$

für die $\operatorname{supp} b_j$ *aus Randzellen und äußeren Zellen besteht.*

Die Hinzufügung der äußeren B-Splines an die inneren B-Splines muss allerdings so geschehen, dass die Approximationsgüte des WEB-Spline-Raumes vom Grad n erhalten bleibt. Dabei ist entscheidend, dass alle Polynome mit Koordinatengrad $\leq n$ Elemente des resultierenden Raumes bleiben. Daher erhält man die folgende Definition.

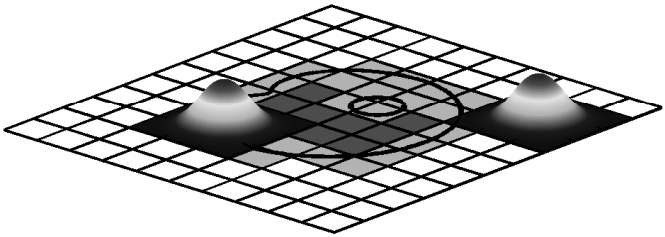


Abbildung 2.3: Innere und äußere B-Splines auf einem Gebiet

Definition 2.7: *WEB-Splines (Gewichtete erweiterte B-Splines)*
Für einen äußeren Index $j \in J$ ist $I(j) = \{\ell + 0, \ldots, n^m\} \subset I$ ein m-dimensionaler Würfel mit inneren Indizes, die j am nächsten sind. Dabei wird angenommen, dass h klein genug ist, so dass solch ein Würfel existiert. Außerdem werden durch

$$e_{i,j} = \prod_{\nu=1}^{m} \prod_{\substack{\mu=0 \\ \ell_\nu + \mu \neq i_\nu}}^{n} \frac{j_\nu - \ell_\nu - \mu}{i_\nu - \ell_\nu - \mu} \tag{2.5}$$

die Werte der mit $J(i)$ in Verbindung stehenden Lagrange-Polynome und durch $J(i)$ die Menge aller j mit $i \in I(j)$ bezeichnet. Die B-Splines mit

$$B_i = \frac{w}{w(x_i)} \left[b_i + \sum_{j \in J(i)} e_{i,j} b_j \right], \quad i \in I\,, \tag{2.6}$$

bilden eine Basis für den WEB-Raum $w^e \mathbb{B}_h^n(\Omega)$. Dabei ist x_i die Mitte einer inneren Gitterzelle $Q_i \subseteq \overline{\Omega} \cup \operatorname{supp} b_i$.

In der Abbildung 2.4 wurde dem äußeren Index j ein mögliches Quadrat mit neun Indizes zugeordnet.
Die -3 berechnete sich dabei mit $j - \ell = (-1, 0)$ und $i - \ell = (1, 0)$ folgendermaßen

$$e_{i,j} = \begin{pmatrix} \frac{-1-0}{1-0} & \frac{-1-2}{1-2} \end{pmatrix} \begin{pmatrix} \frac{0-1}{0-1} & \frac{0-2}{0-2} \end{pmatrix} = -3\,.$$

Dabei übernehmen die WEB-Splines alle wesentlichen Eigenschaften von den B-Splines, außer der Positivität. Für die Finite-Elemente-Approximation sind die folgenden Eigenschaften besonders relevant:

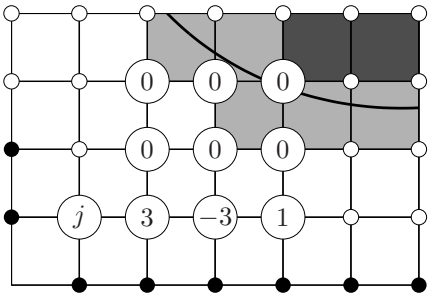


Abbildung 2.4: WEB-Splines bei biquadratischen Ansatzfunktionen

- Erweiterungskoeffizienten: Es gilt $|e_{i,j}| \preceq 1$ und $e_{i,j} = 0$ für $\|i - j\| \succeq 1$. Außerdem müssen für kleines h nur $\preceq h^{m-1}$ B-Splines verändert werden, für die Mehrheit der inneren Indizes i gilt $B_i = \frac{w}{w(x_i)} b_i$.
- Lokale Träger: Der Durchmesser von $\operatorname{supp} B_i$ ist $\preceq h$ und in jedem Würfel $Q \cup \overline{\Omega}$ sind nur $\preceq 1$ WEB-Splines ungleich Null.
- Stabilität: Die WEB-Splines sind linear unabhängig und

$$\left\| \sum_i c_i B_i \right\|_0 \asymp h^{\frac{m}{2}} \|C\| .$$

  Insbesondere gilt $\|B_i\|_0 \asymp h^{\frac{m}{2}}$.
- Approximationsordnung: Der WEB-Raum $w^e \mathbb{B}_h$ beinhaltet gewichtete Polynome vom Koordinatengrad $\leq n$. Außerdem gilt

$$\inf_{u_h \in w^e \mathbb{B}_h} \|u - u_h\|_0 \preceq h^{n+1} ,$$

  falls w und $\frac{u}{w}$ glatt ist.

In allen Abschätzungen hängen die Konstanten vom Grad n, vom Gebiet Ω und der Gewichtsfunktion w ab.

Kapitel 3

Eigenwertproblem

Betrachtet man das Eigenwertproblem theoretisch, hängt die Konvergenzordnung vom Splinegrad des WEB-Raumes $w^e\mathbb{B}_h^n$ ab. Das Ziel dieses Kapitels ist, eine Fehlerschranke für die Eigenwerte und Eigenfunktionen zu finden. Dabei orientiert sich der Beweis am Buch von [SF88]. Im zweiten Abschnitt werden spezielle Beispiele auf unterschiedlichen Gebieten durchgerechnet.

3.1 Theorie

In diesem Abschnitt soll die Approximationsordnung der Eigenwerte von elliptischen Eigenwertproblemen zweiter Ordnung

$$Lu = \lambda u$$

mit

$$Lu = -\sum_{i,j=1}^{n} \partial_j \left(a_{i,j}(x)\partial_i\, u\right) + \sum_{i=1}^{n} b_i(x)\partial_i\, u + c(x)u \tag{3.1}$$

untersucht werden. Dabei ist E_ℓ der Raum, der durch die richtigen Eigenfunktionen $u_1, \ldots, u_\ell$ mit den Eigenvektoren $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_\ell$ aufgespannt wird. Pu ist die WEB-Standard-Projektion. Für die Untersuchung werden mehrere Lemmata benötigt. Dabei sind die meisten Beweise fast analog zu [SF88] und werden deshalb hier nicht ausgeführt.

Lemma 3.1: *e_ℓ ist die Menge der Einheitsfunktionen in E_ℓ und*

$$\sigma_\ell^h = \max_{u \in e_\ell} |2(u, u - Pu) - (u - Pu, u - Pu)| \ .$$

Unter der Bedingung, dass $\sigma_\ell^h < 1$ ist, sind die Eigenwerte durch

$$\lambda_\ell^h \leq \frac{\lambda_\ell}{1 - \sigma_\ell^h}$$

nach oben beschränkt.

Um σ_ℓ^h abzuschätzen, benötigt man die folgende Identität.

Lemma 3.2: *Falls $u = \sum_{i=1}^{\ell} c_i u_i \in e_\ell$ ist, dann gilt*

$$(u, u - Pu) = \sum c_i \lambda_i^{-1} a(u_i - Pu_i, u - Pu)\,.$$

Für die weiteren Schritte wird noch nachfolgender Satz aus [BO91] benötigt.

Satz 3.1: *Für $k \geq 2$ soll für den Rand und für die Funktionen aus Gleichung (3.1) gelten:*

- $\partial\Omega \in C^k$
- $a_{i,j},\, b_i \in C^{k-1}\left(\overline{\Omega}\right)$
- $c \in C^{k-2}\left(\overline{\Omega}\right)$

Damit erhält man, dass alle Eigenfunktionen des Differentialoperators in $H^k(\Omega)$ liegen und dass

$$\|u_j\|_k \leq c_1 \lambda_j^{k/2} \|u_j\|_0 \qquad j = 1, 2, \ldots$$

gilt.

Damit ergibt sich die endgültige Abschätzung für die Eigenwerte.

Satz 3.2: *Für die Basisfunktionen $w^e \mathbb{B}_h^n$ und eine Standardgewichtsfunktion w gibt es eine Konstante δ, so dass die approximierten Eigenwerte für ein kleines h durch*

$$\lambda_\ell \leq \lambda_\ell^h \leq \lambda_\ell + 2\delta h^{2n} \lambda_\ell^{n+1}$$

beschränkt sind. Dabei müssen für die Funktionen $a_{i,j}$, b_i, c mit $0 \leq i, j \leq m$ aus Gleichung (3.1) und den Rand die Voraussetzungen des Satzes 3.1 gelten.

Beweis: Der Beweis läuft fast analog zu [SF88]. Allerdings wendet man auf den Term

$$\begin{aligned}
2\,|(u, u - Pu)| \leq & 2KC^2 h^{2n} \left\| \sum_{i=1}^{\ell} c_i \lambda_i^{-1} u_i \right\|_{n+1} \|u\|_{n+1} \\
\leq & C' h^{2n} \left\| \sum_{i=1}^{\ell} c_i \lambda_i^{(n+1)/2-1} u_i \right\|_0 \left\| \sum_{i=1}^{\ell} c_i \lambda_i^{(n+1)/2} u_i \right\|_0 \\
\leq & C' h^{2n} \lambda_\ell^n
\end{aligned}$$

die Jackson-Ungleichung aus Anhang A.5 an. Der vorletzte Schritt benötigt die Ungleichung $\|u_j\|_k \leq c\|\lambda^{k/2} u_j\|_0$. Für $k = 1$ ist es die Elliptizität $\|u_j\|_1^2 \leq c^2\, a(u_j, u_j)$. Für alle anderen $k \in \mathbb{N}$ gilt die Ungleichung mit dem Satz 3.1.
Der andere Term von σ_ℓ^h ist höherer Ordnung von h, da man mit der Jackson-Ungleichung mit $\ell = 0$

$$(u - Pu, u - Pu) \leq C^2\, h^{2(n+1)} \|u\|_{n+1}^2$$

erhält.

□

Dadurch bekommt man die Fehlerabschätzung

$$\lambda_\ell^h - \lambda_\ell \leq C h^{2n} \lambda_\ell^{n+1}$$

für den ℓ-ten Eigenwert bei Gitterweite h.
Die approximierten Eigenwerte haben also dieselbe Approximationsordnung wie bei den gewöhnlichen Finiten Elementen.
Im nächsten Schritt wird die Approximationsordnung der Eigenfunktionen untersucht. Für die richtige Konvergenzrate der Eigenfunktionen benötigt man wieder einige Vorbereitungsschritte. Als erstes wird die folgende Gleichung bewiesen.

Lemma 3.3: *Sind* $(u_\ell, u_\ell) = \left(u_\ell^h, u_\ell^h\right) = 1$ *normiert, dann gilt*

$$a\left(u_\ell - u_\ell^h, u_\ell - u_\ell^h\right) = \lambda_\ell \left\|u_\ell - u_\ell^h\right\|_0^2 + \lambda_\ell^h - \lambda_\ell\,. \tag{3.2}$$

Somit muss man noch eine Fehlerabschätzung von $\left\|u_\ell - u_\ell^h\right\|_0^2$ finden und kann die alte Fehlerabschätzung von $\lambda_\ell^h - \lambda_\ell$ für den Rest verwenden. Aber auch hierfür ist Vorarbeit notwendig.

Lemma 3.4: *Für alle j und ℓ gilt*

$$\left(\lambda_j^h - \lambda_\ell\right)\left(Pu_\ell, u_j^h\right) = \lambda_\ell\left(u_\ell - Pu_\ell, u_j^h\right) . \tag{3.3}$$

Die Eigenfunktionen $u_1^h, \dots u_n^h$ sollen eine Orthonormalbasis für S^h bilden, deshalb gilt

$$Pu_\ell = \sum_{j=1}^{n}\left(Pu_\ell, u_j^h\right) u_j^h . \tag{3.4}$$

Sind keine mehrfachen Eigenwerte vorhanden, dann findet man für kleines h ein δ mit

$$\frac{\lambda_\ell}{\left|\lambda_j^h - \lambda_\ell\right|} \le \delta \qquad \forall j \neq \ell .$$

Mit $\alpha = \left(Pu_\ell, u_\ell^h\right)$ kann folgende Abschätzung berechnet werden

$$\begin{aligned}
\left\|Pu_\ell - \alpha u_\ell^h\right\|_0^2 &= \sum_{j\neq\ell}\left(Pu_\ell, u_j^h\right)^2 \\
&= \sum_{j\neq\ell}\left(\frac{\lambda_\ell}{\lambda_j^h - \lambda_\ell}\right)^2\left(u_\ell - Pu_\ell, u_j^h\right)^2 \\
&\le \delta^2 \sum_{j\neq\ell}\left(u_\ell - Pu_\ell, u_j^h\right)^2 \\
&\le \delta^2 \left\|u_\ell - Pu_\ell\right\|_0^2 .
\end{aligned} \tag{3.5}$$

Der letzte Schritt in der Abschätzung ergibt sich mit $a_j = \left(u_\ell - Pu_\ell, u_j^h\right)$

$$\begin{aligned}
0 &\le \left\| u_\ell - Pu_\ell - \sum_{j=1}^{n} a_j u_j^h \right\|_0^2 \\
&= (u_\ell - Pu_\ell, u_\ell - Pu_\ell) - 2\sum_{j=1}^{n} a_j\left(u_\ell - Pu_\ell, u_j^h\right) + \sum_{j,k=1}^{n} a_j a_k\left(u_j^h, u_k^h\right) \\
&= \left\|u_\ell - Pu_\ell\right\|_0^2 - 2\sum_{j=1}^{n} a_j\left(u_\ell - Pu_\ell, u_j^h\right) + \sum_{j=1}^{n} a_j^2 \\
&= \left\|u_\ell - Pu_\ell\right\|_0^2 - \sum_{j=1}^{n}\left(u_\ell - Pu_\ell, u_j^h\right)^2 + \sum_{j=1}^{n}\left(\left(u_\ell - Pu_\ell, u_j^h\right) - a_j\right)^2 .
\end{aligned}$$

Setzt man nun a_j ein, dann wird der letzte Term Null, und zusammengefasst ist es

$$\begin{aligned} 0 &\leq \|u_\ell - Pu_\ell\|_0^2 - \sum_{j=1}^{n} \left(u_\ell - Pu_\ell, u_j^h\right)^2 \\ \Longrightarrow \quad & \sum_{j=1}^{n} \left(u_\ell - Pu_\ell, u_j^h\right)^2 \leq \|u_\ell - Pu_\ell\|_0^2 \,. \end{aligned}$$

Mit $\alpha = \left(Pu_\ell, u_\ell^h\right)$ und der Dreiecksungleichung aus A.1 ist

$$\left\|u_\ell - \alpha u_\ell^h\right\|_0 \leq \|u_\ell - Pu_\ell\|_0 + \left\|Pu_\ell - \alpha u_\ell^h\right\|_0 \leq (1+\delta)\,\|u_\ell - Pu_\ell\|_0 \,.$$

Mit der Jackson-Ungleichung erhält man eine Abschätzung für den Fehler $\|u_\ell - Pu_\ell\|_0$ und damit

$$\begin{aligned} \left\|u_\ell - \alpha u_\ell^h\right\|_0 &\leq c_1\,(1+\delta)\,h^k\,\|u_\ell\|_k \\ &\leq C h^k \lambda_\ell^{k/2} \,. \end{aligned} \tag{3.6}$$

Satz 3.3: *Für den Raum $w^e\mathbb{B}_h^n$ und einen einfachen Eigenwert gilt*

$$\left\|u_\ell - u_\ell^h\right\| \leq c_1 h^{n+1} \lambda_\ell^{(n+1)/2} \,, \tag{3.7}$$

$$a\left(u_\ell - u_\ell^h, u_\ell - u_\ell^h\right) \leq c_2 h^{2n} \lambda_\ell^{n+1} \,. \tag{3.8}$$

Mit der Ungleichung (3.6) bekommt man die Fehlerabschätzung für die Ungleichung (3.7) mit $k = n+1$ und mit Gleichung (3.2) erhält man eine Abschätzung für die Gleichung (3.8).

Für mehrfache Eigenwerte $\lambda_\ell = \lambda_{\ell+1} = \cdots = \lambda_{\ell+M}$ findet man für kleines h eine Konstante δ, die diese Eigenwerte von den anderen Eigenwerten trennt. Die Konstante α ist in diesem Fall eine Matrix der Größe $(M+1)\times(M+1)$:

$$\alpha_{m,i} = \left(Pu_{\ell+m}, u_{\ell+i}^h\right) \qquad \text{für } 0 \leq i,\, m \leq M \,.$$

Wie in (3.5) gilt mit der Matrix α für $0 \le m \le M$ die Ungleichung

$$\begin{aligned}\left\| Pu_{\ell+m} - \sum_{i=0}^{M} \alpha_{m,i} u^h_{\ell+i} \right\|_0^2 &= \sum_{j \notin \{\ell,\dots,\ell+M\}} \left(Pu_{\ell+m}, u^h_j\right)^2 \\ &= \sum \left(\frac{\lambda_\ell}{\lambda^h_j - \lambda_\ell} \right)^2 \left(u_{\ell+m} - Pu_{\ell+m}, u^h_j\right)^2 \\ &\le \delta^2 \sum \left(u_{\ell+m} - Pu_{\ell+m}, u^h_j\right)^2 \\ &\le \delta^2 \left\| u_{\ell+m} - Pu_{\ell+m} \right\|_0^2 .\end{aligned}$$

Weitere Umformungen ergeben

$$\begin{aligned}\left\| u_{\ell+m} - \sum_{i=0}^{M} \alpha_{m,i} u^h_{\ell+i} \right\|_0 &\le \left\| u_{\ell+m} - Pu_{\ell+m} \right\|_0 + \left\| Pu_{\ell+m} - \sum_{i=0}^{M} \alpha_{m,i} u^h_{\ell+i} \right\|_0 \\ &\le (1+\delta) \left\| u_{\ell+m} - Pu_{\ell+m} \right\|_0 .\end{aligned}$$

Nun wird wieder die Jackson-Ungleichung angewendet

$$\begin{aligned}\left\| u_{\ell+m} - \sum_{i=0}^{M} \alpha_{m,i} u^h_{\ell+i} \right\|_0 &\le c_1 (1+\delta) h^k \left\| u_{\ell+m} \right\|_k \\ &\le c_2 h^k \lambda_\ell^{k/2} .\end{aligned}$$

Im Folgenden wird die Matrix α invertiert und die Gleichung mit α^{-1} multipliziert. Dadurch erhält man einen Vektor $U_{\ell+m} = \sum_{i=0}^{M} \gamma_i u_{\ell+i}$, der eine Linearkombination aus den Eigenvektoren zu dem mehrfachen Eigenwert λ_ℓ ist. Man setzt für $k = n+1$ in die Gleichung ein und bekommt

$$\left\| u^h_{\ell+m} - U_{\ell+m} \right\|_0 \le c h^{n+1} \lambda^{(n+1)/2} .$$

Dies ist dieselbe Konvergenzordung wie bei den einfachen Eigenwerten. Damit hat man bei den Eigenwerten und Eigenvektoren bzw. Eigenfunktionen dieselbe Fehlerordnung wie bei den gewöhnlichen Finiten Elementen.

3.2 Anwendungsbeispiele

Um die Konvergenz der Eigenwerte praktisch nachzuweisen, verwendet man einige Beispielgebiete im Zweidimensionalen. Für einfache Gebiete,

wie die Einheitskreisscheibe und ein beliebiges Rechteck, kennt man die Eigenfunktionen und die exakten Eigenwerte. Es werden die Eigenwerte folgender Differentialgleichung

$$\begin{aligned} -\Delta u &= \lambda u \quad \text{in } \Omega \\ u &= 0 \quad \text{auf } \partial\Omega \end{aligned}$$

numerisch berechnet.

3.2.1 Einheitskreis

Als erstes wird der Einheitskreis untersucht, da dabei keine Ecken auftreten und die berechneten Eigenwerte mit den exakten Eigenwerten verglichen werden können. Auf dem Einheitskreis sind die Eigenwerte $\lambda_{k,n} = \mu^2_{|k|,n}$ mit den Nullstellen $\mu_{|k|,n}$ der Bessel-Funktionen $J_{|k|}$. Die ersten zehn Nullstellen sind beispielsweise:

$$2.40,\ 3.83,\ 3.83,\ 5.14,\ 5.14,\ 5.52,\ 6.38,\ 6.38,\ 7.02,\ 7.02$$

Für die Konvergenz werden B-Splines von der Ordnung eins bis vier untersucht. Dabei wird die Startgitterweite $h = 1/2$ fünfmal halbiert. Für die ersten zehn Eigenwerte erhält man die Fehler und die Fehlerordnung für verschiedene Splinegrade wie in Abbildung 3.1.

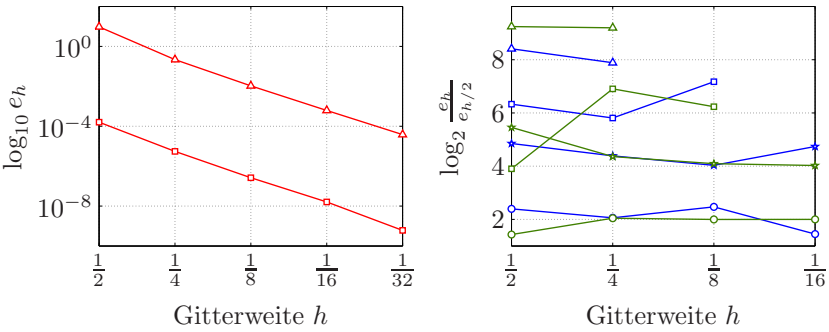
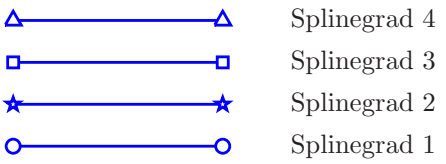


Abbildung 3.1: Fehlerordnung der Eigenwerte am Einheitskreis

Im ersten Bild erkennt man, dass die Fehlerkurven für den ersten und zehnten Eigenwert beim Splinegrad zwei auf der logarithmischen Skala eine Gerade bilden. Außerdem liegt die Gerade für den zehnten Eigenwert

über der Geraden für den ersten Eigenwert, da in der Fehlerabschätzung die Größe des Eigenwertes eine Rolle spielt. Die Bilder für andere Splinegrade sehen gleich aus, allerdings werden die beiden Kurven nach unten verschoben und die Steigung der Geraden wird größer.
Das zweite Bild zeigt die optimale Approximationsordnung $2n$ wieder für den ersten und zehnten Eigenwert, also dieselbe wie bei den gewöhnlichen FEM.



Bei den höheren Splinegraden können nicht alle Gitterweiten untersucht werden, da die Approximation schnell sehr genau wird und die Rechenungenauigkeit einen zu großen Einfluss hat.

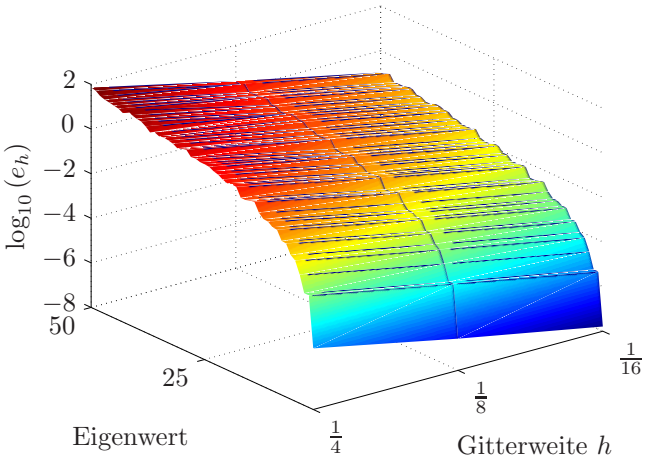


Abbildung 3.2: Fehler der ersten 50 Eigenwerte

Die Fehlerabschätzung

$$\lambda_\ell^h - \lambda_\ell \leq Ch^{2n}\lambda_\ell^{n+1}$$

für die Eigenwerte hängt beim ℓ-ten Eigenwert nicht nur von der Gitterweite h, sondern auch vom Betrag des Eigenwertes ab, d. h. der Fehler nimmt zu, falls der Betrag des Eigenwertes sich vergrößert. Dies wird auch in der Abbildung 3.2 ersichtlich. Dabei wurden die Fehler $e_{\ell,h} = \lambda_\ell^h - \lambda_\ell$ der kleinsten 50 Eigenwerte bei den Gitterweiten $h = \frac{1}{4}, \frac{1}{8}$ und $\frac{1}{16}$ beim Splinegrad zwei berechnet. Hierbei wächst der Fehler bei höheren Eigenwerten sehr stark an.

3.2.2 Beispiel mit glattem Rand und Loch

Hierbei handelt es sich um ein allgemeines Beispiel ohne Ecken, allerdings sind die exakten Eigenwerte nicht bekannt. Durch das Loch soll die Symmetrie des Gebietes gestört werden. Das Gebiet ist in Abbildung 3.3 dargestellt.

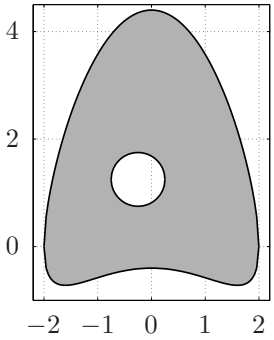


Abbildung 3.3: Gebiet mit glattem Rand

Da die Eigenwerte nicht bekannt sind, berechnet man die Fehler der Eigenwerte, indem man sie immer mit den Eigenwerten auf dem Gitter mit halber Gitterweite vergleicht. Man geht davon aus, dass die Eigenwerte auf dem Gitter mit halber Gitterweite die exakten Eigenwerte sind. Die Fehlerkurven für die Eigenwerte sind in Abbildung 3.4 dargestellt. Auch hier erkennt man im ersten Bild, dass die Fehlerkurven für den ersten und zehnten Eigenwert eine Gerade bilden und dass die Gerade für den zehnten Eigenwert oberhalb der anderen Geraden liegt, der Fehler also immer größer ist. Dies entspricht wieder dem theoretischen Beweis.
Im zweiten Bild ist die experimentelle Approximationsordnung zu sehen, allerdings nicht so schön wie im ersten Beispiel.

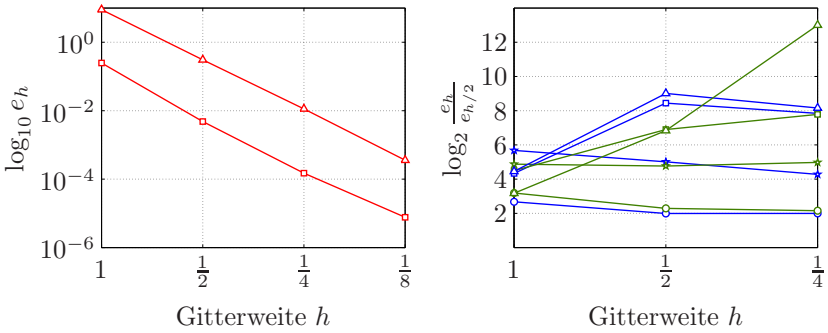


Abbildung 3.4: Fehlerordnung der Eigenwerte für glatten Rand

3.2.3 Beliebiges Beispiel mit Ecken

Als Nächstes wird ein beliebiges Gebiet mit Ecken (Abbildung 3.5) genommen, bei dem man die Eigenwerte wie im vorherigen Beispiel ebenfalls nicht kennt. In diesem Beispiel wurde die Bedingung der Glattheit der Gewichtsfunktion aufgrund der Ecken nicht eingehalten. Hier wurde für die Gewichtsfunktion w keine Standardgewichtsfunktion verwendet, und im Beweis für die Approximationsordnung kann die Jackson-Ungleichung nicht angewendet werden, die allerdings für Satz 3.2 benötigt wird.

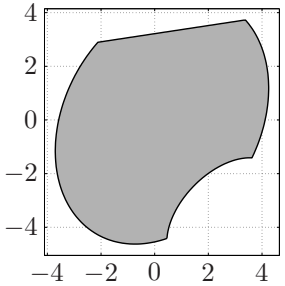
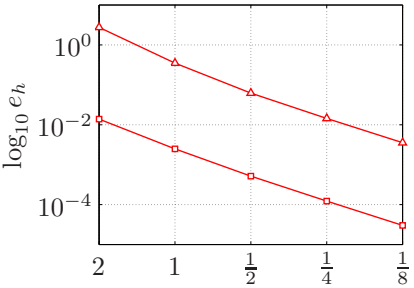
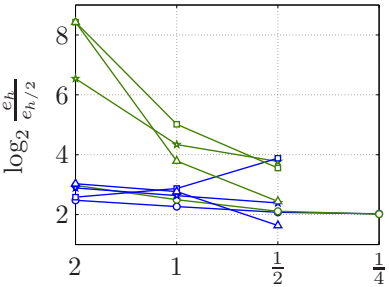


Abbildung 3.5: Ein beliebiges Gebiet mit Ecken

Aus diesem Grund erhält man nicht die bewiesene Approximationsordnung und man sieht, dass bei höheren Ansatzgraden die Bedingung einer

Standardgewichtsfunktion vorausgesetzt werden sollte. Im linken Bild wurde der erste und zehnte Eigenwert beim Splinegrad eins abgebildet. Für diesen Ansatzgrad erhält man die optimale Konvergenzordnung zwei, für alle anderen Ansatzgrade nicht.





Kapitel 4

Lanczos-Verfahren

Nach der Assemblierung der Finite-Elemente-Matrizen A_h und $B_h \in \mathbb{R}^{n \times n}$ für das verallgemeinerte Eigenwertproblem

$$A_h u = \lambda B_h u$$

werden im folgenden Schritt die Eigenwerte oder die Eigenvektoren und Eigenwerte berechnet. Dabei existieren mehrere Varianten, um die Eigenwerte dieses Problems zu bestimmen, zum Beispiel

- Bestimmung der Nullstellen des charakteristischen Polynoms,
- die Mises-Iteration,
- die Wielandt-Iteration und
- der QR-Algorithmus.

Die Nachteile der verschiedenen Verfahren sind unter anderem der hohe Rechen- und Zeitaufwand, dass nur einer oder alle Eigenwerte berechnet werden und die numerische Stabilität. Daher wurde in diesem Fall der Lanczos-Algorithmus implementiert. Hierbei werden die gesuchten Eigenwerte λ_i in einer Folge von Teilräumen $V_1 \subset V_2 \subset \ldots \subset \mathbb{R}^n$ approximiert.
Dieses Verfahren wurde von Cornelius Lanczos [Lan50] entwickelt. Cornelius Lanczos wurde am 2. Februar 1893 in Szekesfehervar, Ungarn, geboren. Er studierte ab 1911 an der Budapester Universität, die er 1921 mit dem Doktortitel verließ. 1921 bis 1930 forschte er in Deutschland, erst als Assistent an der Freiburger Universität, dann in Frankfurt am

Main, wo er auch seine Habilitation abschloss. Danach war er Stipendiat der Notgemeinschaft der deutschen Wissenschaft bei Albert Einstein. Er verließ Deutschland 1931 und forschte von 1931 bis 1951 in den USA. Er war dort unter anderem Professor der mathematischen Physik an der Purdue Universität in Lafayette, war hauptamtlicher Mitarbeiter des National Bureau of Standards und hatte den Posten eines Senior Research Engineer bei der Boeing Airplane Company in Seattle. In seinen letzten Jahren, von 1952 bis 1974, war er Professor am Institute for Advanced Studies in Dublin, Irland. Am 24. Juni 1974 starb er in Budapest.

4.1 Voraussetzungen und Algorithmus

Für die Bestimmung der Eigenwerte in einem Unterraum V_k von $\mathbb{R}^n$ wird eine Orthogonalbasis von V_k gebildet. Die Berechnung wird mit Hilfe der folgenden Drei-Term-Rekursion konstruiert:

Satz 4.1: *Sei $V_1 \subset V_2 \subset \ldots \subset \mathbb{R}^n$ eine aufsteigende Kette von Unterräumen der Dimension $\dim V_k = k$ in einem Vektorraum $\mathbb{R}^n$ und $A : \mathbb{R}^n \mapsto \mathbb{R}^n$ eine selbstadjungierte lineare Abbildung bezüglich eines Skalarprodukts $(\,\cdot\,,\,\cdot\,)$ auf $\mathbb{R}^n$, d. h.*

$$(Au, v) = (u, Av) \quad \forall\, u, v \in \mathbb{R}^n \;,$$

so dass

$$A\,(V_k) \subset V_{k+1} \quad \textit{und} \quad A\,(V_k) \not\subset V_k$$

ist. Dann gibt es zu jedem $v_1 \in V_1$ genau eine Erweiterung zu einem Orthonormalsystem $\{v_k\}$ mit $v_k \in V_k$ für alle k und

$$1 = (Av_{k-1}\,,\, v_k) \quad \forall\, k \geq 2 \;.$$

Die Familie $\{v_k\}$ berechnet sich mit der Drei-Term-Rekursion und der anschließenden Normierung

$$\begin{aligned} v_k &= (A + \alpha_k)\, v_{k-1} + \beta_k v_{k-2} \quad \textit{für} \quad k = 2, 3, \ldots \\ v_k &= \frac{v_k}{\|v_k\|} \end{aligned}$$

mit $v_0 := 0$ und

$$\alpha_k := -\,(Av_{k-1}\,,\, v_{k-1}) \quad , \quad \beta_k := -\,(v_{k-1}\,,\, v_{k-1}) \;.$$

Beweis: (mit Induktion)
Induktionsanfang $k = 1$: v_1 bildet ein Orthonormalsystem.
Induktionsschluss: Sei $v_1, \ldots, v_{k-1}$ eine bereits konstruierte Orthonormalbasis. v_k kann dabei so gewählt werden, dass v_k orthogonal zu v_i mit $i \in \{1\,, \ldots\,, k-1\}$ steht und $v_k = Av_{k-1} + \sum_{j=1}^{k-1} \alpha_j v_j$ gilt. Dann weiß man, dass $(v_k - Av_{k-1})$ in V_{k-1} liegt, da

$$(v_k\,,\, v_k) = (Av_{k-1}\,,\, v_k) \quad \Longrightarrow \quad (v_k - Av_{k-1}\,,\, v_k) = 0\,.$$

Damit erhält man

$$v_k - Av_{k-1} = \sum_{j=1}^{k-1} c_j v_j \quad \text{mit} \quad c_j = (v_k - Av_{k-1}\,,\, v_j)\ .$$

Da v_k orthogonal zu $v_1, \ldots, v_{k-1}$ stehen soll und A eine selbstadjungierte Abbildung ist, gilt

$$c_j = \underbrace{(v_k\,,\, v_j)}_{=0} - (Av_{k-1}\,,\, v_j) = -\,(v_{k-1}\,,\, Av_j)\ .$$

Wegen $Av_j \in V_{j+1}$ bekommt man, dass $c_1, \ldots, c_{k-3} = 0$ sein muss und

$$\begin{aligned} c_{k-1} &= -\,(Av_{k-1}\,,\, v_{k-1})\ , \\ c_{k-2} &= -\,(v_{k-1}\,,\, Av_{k-2}) = -\,(v_{k-1}\,,\, v_{k-1})\ . \end{aligned}$$

Mit

$$v_k = (A + c_{k-1})\, v_{k-1} + c_{k-2} v_{k-2} = (A + \alpha_k)\, v_{k-1} + \beta_k v_{k-2}$$

erhält man das nächste v_k orhogonal zu v_i mit $i < k$ und induktiv die Behauptung. Die Orthonormalität ergibt sich mit Hilfe der Normierung. □

Zur Berechnung der Eigenwerte werden beim Lanczos-Algorithmus die Krylov-Unterräume V_k

$$V_k(x) = \operatorname{span}\left\{x, Ax, \ldots, A^{k-1}x\right\}$$

mit $x \neq 0$ genommen.
Damit bekommt man Algorithmus 4.1 für das Eigenwertproblem

$$Au = \lambda u\,.$$

Dabei wird als Startvektor ein beliebiger, zufällig generierter Vektor $v_1 \neq 0$ genommen. Die Konvergenz bezüglich eines Eigenwertes λ_i und dessen Eigenvektors v_i ist relativ langsam, falls dieser fast senkrecht auf v_1 steht, also $v_1^t v_i \approx 0$ ist. Der Eigenwert kann nicht approximiert werden, wenn sogar $v_1^t v_i = 0$ ist. Mit der zufälligen Wahl von v_1 tritt dieser Fall eigentlich nie ein.

Algorithmus 4.1 Lanczos-Algorithmus

wähle v_1 mit $\|v_1\| = 1$ beliebig,
setze $\beta_1 = 0$ und $v_0 = 0$
for $i = 1 : m$ **do**
  $w_i = Av_i$
  $\alpha_i = (w_i, v_i)$
  $w_i = w_i - \alpha_i v_i - \beta_i v_{i-1}$
  $\beta_{i+1} = \|w_i\|$
  $v_{i+1} = \frac{w_i}{\beta_{i+1}}$
end for

Mit Algorithmus 4.1 erhält man eine Orthonormalbasis $\{v_k\}$ und eine spaltenorthonormale Matrix

$$X_k = [v_1 , \dots , v_k] ,$$

für die

$$X_k^t A X_k = \begin{pmatrix} \alpha_1 & \beta_2 & & & \\ \beta_2 & \alpha_2 & \beta_3 & & \\ & \ddots & \ddots & \ddots & \\ & & \beta_{k-1} & \alpha_{k-1} & \beta_k \\ & & & \beta_k & \alpha_k \end{pmatrix} = T_k$$

und

$$X_k^t X_k = I$$

gilt. T_k wird als Lanczos-Matrix bezeichnet, und deren Eigenwerte approximieren die Eigenwerte von A. Die Diagonale ergibt sich mit der Definition von α_i im Algorithmus 4.1. Dabei ist $\alpha_i = (w_i, v_i) = (Av_i, v_i) = v_i^t A v_i$. Die Nebendiagonale resultiert ebenfalls aus Algorithmus 4.1. Man nimmt die Zeile $v_{i+1} = \frac{w_i}{\beta_{i+1}}$, multipliziert mit β_{i+1} und ersetzt w_i. Somit bekommt man

$$v_{i+1}\beta_{i+1} = Av_i - \alpha_i v_i - \beta_i v_{i-1} .$$

Nun wird mit v_{i+1}^t multipliziert und berücksichtigt, dass $v_{i+1}^t v_{i+1} = 1$ und $v_{i+1}^t v_j = 0$ mit $j \neq i+1$ gilt.

$$\beta_{i+1} = v_{i+1}^t A v_i$$

Die restlichen Einträge sind Null, da $Av_j \in \text{span}\{v_1, \ldots, v_{j+1}\}$ liegt und daher das Skalarprodukt aufgrund der Orthogonalität mit einem Vektor $v_{j+\ell}$ mit $\ell \geq 2$ immer Null ergibt. Dadurch ist die Berechnung der Eigenwerte von T_k wesentlich einfacher, da die Matrix T_k tridiagonal und kleiner als A ist. Seien nun $\lambda_1 \leq \ldots \leq \lambda_k$ die Eigenwerte von T_k und $s_1, \ldots, s_k$ die zugehörenden Eigenvektoren, dann nennt man λ_i , $i = 1, \ldots k$ Ritz-Werte und $u_i = X_k s_i$, $i = 1, \ldots k$ Ritz-Vektoren . Erhält man im k-ten Schritt des Lanczos-Algorithmus $\beta_{k+1} = 0$, so ist der Unterraum V_k invariant unter A und es gilt

$$AX_K = X_k T_k \,.$$

Die approximierten Eigenwerte und Eigenvektoren sind in diesem Fall exakt. Will man weitere Eigenwerte bestimmen, so startet man den Lanczos-Algorithmus mit einem v_1^* mit

$$v_i^t v_1^* = 0 \,, \quad i = 1, \ldots, k$$

neu.

Eine wichtige Frage ist, welche Eigenwerte eigentlich vom Lanczos-Algorithmus als erstes approximiert werden. Dazu wurde im Buch von Trefethen und Bau [TB97] die Faustregel eingeführt, dass die Lanczos-Methode dazu tendiert, Eigenwerte großer symmetrischer Matrizen zu finden, die in Gebieten liegen, die eine geringere Dichte haben als die Gleichgewichtsverteilung auf $[-1, 1]$ mit der Dichte

$$f(\lambda) = \frac{1}{\pi\sqrt{1-\lambda^2}} \,.$$

In der Veröffentlichung von Kuijlaars [Kui00] gibt es eine quantitative Version der Faustregel. Dabei werden die Gebiete beschrieben, welche die Eigenwerte beinhalten, die gut durch die Ritz-Werte approximiert werden. Diese Gebiete hängen von der Verteilung der Eigenwerte und von dem Verhältnis Größe der Matrix zu Anzahl der Iterationen ab.

Man erhält mit

$$\begin{aligned}
\sigma &= \text{Verteilung der Eigenwerte} \\
t &= \frac{\text{Anzahl der Lanczos-Vektoren}}{\text{Zeilen- bzw. Spaltenanzahl der Matrix}} \\
\Lambda(t;\sigma) &= \text{Bereich, in dem die Eigenwerte nach wenigen Schritten} \\
&\quad \text{gut approximiert werden} \\
r(t) &= \text{Funktion auf } (0,1]
\end{aligned}$$

die folgenden Sätze. Dabei wird die Verteilung der Zufallsveränderlichen X durch die Verteilungsfunktion

$$W(x) = P(X \leq x) \quad \text{für} \quad -\infty < x < \infty$$

beschrieben. Sie gibt an, mit welcher Wahrscheinlichkeit die Zufallsgröße X einen Wert zwischen $-\infty$ und x annimmt. Man spricht von einer stetigen Verteilungsfunktion, wenn man die Wahrscheinlichkeit, dass die Zufallsgröße X in $[a,b]$ liegt, durch folgendes Integral darstellen kann:

$$P(a \leq x \leq b) = \int_a^b w(t)\,dt$$

Hierbei wird w die Wahrscheinlichkeitsdichte genannt.

Satz 4.2: *Für σ muss gelten, dass der Träger in $[-1,1]$ liegt und die Dichte w gerade in diesem Intervall, also $w(-\lambda) = w(\lambda)$, ist. Nimmt man an, dass $\sqrt{1-\lambda^2}\,w(\lambda)$ abnimmt für $\lambda \in [0,1]$, dann existiert für jedes $t \in (0,1)$ ein $r(t) \in (0,1]$, so dass*

$$\Lambda(t;\sigma) = \mathbb{R} \backslash [-r(t), r(t)]$$

ist.

Nach Satz 4.2 werden in diesem Fall also die Randeigenwerte gut approximiert. Ein Beispiel dafür ist, wenn alle Eigenwerte im Intervall $[-1,1]$ gleichverteilt sind, also die Eigenwerte wie in Abbildung 4.1 verteilt sind. Bei einer $N \times N$-Matrix wären in diesem Fall die Eigenwerte

$$\lambda_{k,N} = -1 + 2\frac{k-1}{N-1}, \quad k = 1, \ldots, N$$

und die Funktion

$$r(t) = \sqrt{1-t^2}\,.$$

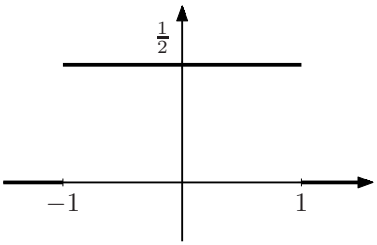


Abbildung 4.1: Gleichverteilung der Eigenwerte

Ein weitere Gruppe von Beispielen wird nach dem folgenden Satz gegeben.

Eine Bedingung, um innere Eigenwerte gut zu approximieren, ist wiederum nach [Kui00].

Satz 4.3: *Es muss wieder gelten, dass der Träger von σ auf $[-1,1]$ ist und die Dichte w in diesem Intervall gerade ist. Nimmt man an, dass $\sqrt{1-\lambda^2}\, w(\lambda)$ zunimmt für $\lambda \in [0,1]$, dann gilt*

$$\Lambda(t;\sigma) = (-\infty,-1) \cup (1,\infty)$$

oder

$$\Lambda(t;\sigma) = (-\infty,-1) \cup (-r(t),r(t)) \cup (1,\infty)$$

für ein $r(t) \in (0,1]$.

Satz 4.3 sagt also aus, wann die einzelnen Eigenwerte außerhalb des Trägers oder die inneren Eigenwerte gut approximiert werden. Ein Beispiel hierfür ist die ultrasphärische Verteilung

$$f(\lambda) = C_\alpha \left(1-\lambda^2\right)^\alpha$$

mit $\alpha > -1$ und $C_\alpha = \frac{\Gamma(\alpha+3/2)}{\sqrt{\pi}\Gamma(\alpha+1)}$. Diese ist in Abbildung 4.2 mit $\alpha = -0.5$ dargestellt. Dabei treten für $\alpha > -\frac{1}{2}$ die Bedingungen aus Satz 4.2 ein, und die Randeigenwerte werden zuerst approximiert. Für $-1 < \alpha < -\frac{1}{2}$ sind die Bedingungen aus Satz 4.3 erfüllt, und die inneren Eigenwerte werden in den ersten Lanczos-Schritten gefunden.

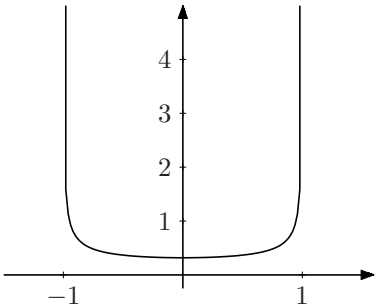


Abbildung 4.2: Ultrasphärische Verteilung der Eigenwerte

4.2 Shift-and-Invert Lanczosverfahren

In der Praxis kommt es öfters vor, dass nur Eigenwerte in einem bestimmten Frequenzbereich von Interesse sind, da z. B. dort gefährliche Situationen eintreten können. Für diesen Fall muss man eine abgeänderte Form des Lanczos-Algorithmus anwenden. Man greift auf die Grundidee der Wielandt-Iteration [Höl98] zurück. Will man die Eigenwerte in der Umgebung von $\sigma > 0$ berechnen, dann wird das verallgemeinerte Eigenwertproblem

$$Au = \lambda Bu \quad \text{in} \quad (A - \sigma B)^{-1} Bu = \frac{1}{\lambda - \sigma} u$$

mit den neuen Eigenwerten $\nu_i = \frac{1}{\lambda_i - \sigma}$ umgewandelt. Dabei ist bei den neuen Eigenwerten ν_i der Betrag am größten, falls λ_i in der Nähe von σ liegt.
Diese werden nun mit der abgeänderten Form des Lanczos-Algorithmus, vgl. [ER80], berechnet.
Dabei gilt in diesem Fall, dass

$$X_k^t B X_k = I$$

ist, also die berechneten v_i eine B-orthonormale Basis bilden.
Die betragsgrößten Eigenwerte ν_i werden mit dem Lanczos-Algorithmus schnell approximiert. Damit kann die Berechnung in verschiedene Frequenzbereiche unterteilt werden. Will man die Eigenwerte im Intervall $[\lambda_1, \lambda_\ell]$ berechnen, kann man dieses in m Segmente unterteilen, für die jeweils

$$m_i = [\lambda_i, \lambda_{i+1}] \ ,$$

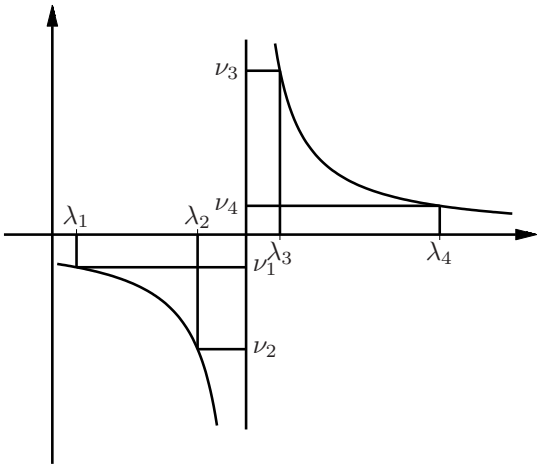
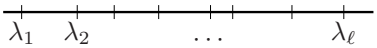


Abbildung 4.3: Eigenwertverteilung mit Spektral-Transformation

mit $i = 1, \cdots, m$ und $\lambda_{m+1} = \lambda_\ell$, gilt (vgl. [Kom03]).



Diese können parallel bearbeitet werden. Dabei ist immer dasselbe mathematische Problem zu lösen, allerdings in verschiedenen Frequenzbereichen. Es häufen sich die Eigenwerte, die weit von σ liegen, bei der Null, und die Eigenwerte, die nahe bei σ liegen, werden getrennt. Dadurch werden auch mit Hilfe von Satz 4.2 und 4.3 die Eigenwerte nahe σ approximiert, da diese eine geringe Dichte aufweisen. Für die Berechnung des Fehlers setzt man $C = (A - \sigma B)^{-1} B$. Dafür wird zuerst folgende Vorüberlegung benötigt. Bei k Lanczos-Schritten mit $k < n$ erhält man in der ersten Spalte von $X_k T_k$ und CX_k

$$\beta_2 v_2 + \alpha_1 v_1$$

und in der i-ten Spalte mit $1 < i < k$

$$\beta_i v_{i-1} + \alpha_1 v_i + \beta_{i+1} v_{i+1} \,.$$

Algorithmus 4.2 Shift-and-Invert Lanczos-Algorithmus

wähle v_1 mit $\|v_1\|_B = 1$ beliebig,
setze $\beta_1 = 0$ und $v_0 = 0$
for $i = 1 : k$ **do**
  $w_i = \underbrace{(A - \sigma B)^{-1} B}_{C} v_i$
  $\alpha_i = (w_i, v_i)_B$
  $w_i = w_i - \alpha_i v_i - \beta_i v_{i-1}$
  $\beta_{i+1} = \|w_i\|_B$
  $v_{i+1} = \frac{w_i}{\beta_{i+1}}$
end for

In der k-ten Spalte bekommt man unterschiedliche Ergebnisse. Für $X_k T_k$

$$\beta_k v_{k-1} + \alpha_k v_k$$

und für CX_k

$$\beta_k v_{k-1} + \alpha_k v_k + \beta_{k+1} v_{k+1} .$$

Also gilt

$$CX_k = X_k T_k + \beta_{k+1} v_{k+1} e_k^T$$

mit dem k-ten Einheitsvektor e_k. s_i sind die Eigenvektoren von T_k. Der Fehler ϵ ist

$$\begin{aligned} \|\epsilon\|_B &= \left\| C\tilde{u}_i - \tilde{\lambda}_i \tilde{u}_i \right\|_B = \left\| \left(CX_k - \tilde{\lambda}_i X_k \right) s_i \right\|_B \\ &= \|(CX_k - X_k T_k)\, s_i\|_B = \left\| \left(\beta_{k+1} v_{k+1} e_k^T \right) s_i \right\|_B \\ &= \beta_{k+1} \, |s_{ik}| \end{aligned}$$

mit $s_{ik} = e_k^T s_i$, da $\|v_{k+1}\|_B = 1$ ist.
Seien nun $\tilde{u}_i$, $\tilde{\lambda}_i$ mit $i = 1, \ldots, k$ die mit dem Lanczos-Verfahren approximierten Eigenvektoren bzw. Eigenwerte und u_i , $\frac{1}{\lambda_i - \sigma}$ die richtigen Eigenvektoren bzw. Eigenwerte. Damit ergibt sich mit der Matrix C und

$$\tilde{u}_i = \sum_k \alpha_k u_k \qquad \|\tilde{u}_i\|_B = 1 \quad \text{und} \quad \sum_k \alpha_k^2 = 1$$

der erste Schritt der Fehlerabschätzung für die Eigenwerte.

$$\begin{aligned}\left\|\left(C-\tilde{\lambda}_i I\right)\tilde{u}_i\right\|_B^2 &= \left\|\left(C-\tilde{\lambda}_i I\right)\sum_k \alpha_k u_k\right\|_B^2 = \left\|\sum_k \left(C-\tilde{\lambda}_i I\right)\alpha_k u_k\right\|_B^2 \\ &= \left\|\sum_k \alpha_k \left(\frac{1}{\lambda_k-\sigma}-\tilde{\lambda}_i\right)u_k\right\|_B^2 \\ &= \sum_k \alpha_k^2\left(\frac{1}{\lambda_k-\sigma}-\tilde{\lambda}_i\right)^2 \geq \sum_k \alpha_k^2\left(\frac{1}{\lambda_i-\sigma}-\tilde{\lambda}_i\right)^2 \\ &= \left|\frac{1}{\lambda_i-\sigma}-\tilde{\lambda}_i\right|^2\end{aligned}$$

Die Ungleichung ist erfüllt, da $\frac{1}{\lambda_i-\sigma}$ am nächsten zu $\tilde{\lambda}_i$ liegt. Insgesamt erhält man also

$$\beta_{k+1}\left|s_{ik}\right| \geq \left|\frac{1}{\lambda_i-\sigma}-\tilde{\lambda}_i\right| .$$

Man will allerdings wissen, wie gut λ_i von $\tilde{\lambda}_i^{-1}+\sigma$ geschätzt wird. Dafür werden folgende mathematischen Umformungen gemacht:

$$\begin{aligned}\left|\lambda_i-\left(\frac{1}{\tilde{\lambda}_i}+\sigma\right)\right| &= \left|\left(\lambda_i-\sigma\right)\tilde{\lambda}_i^{-1}\left(\frac{1}{\lambda_i-\sigma}-\tilde{\lambda}_i\right)\right| \\ &= \left|\lambda_i-\sigma\right|\left|\tilde{\lambda}_i^{-1}\right|\left|\frac{1}{\lambda_i-\sigma}-\tilde{\lambda}_i\right| \\ &\leq \left|\frac{1}{\tilde{\lambda}_i}\right|\left|\lambda_i-\sigma\right|\beta_{k+1}\left|s_{ik}\right|\end{aligned}$$

Durch folgende Überlegungen erhält man die endgültige Abschätzung. Sei ν_m mit $m=1,\ldots,n$ und $\nu_1 \leq \nu_2 \leq \ldots \leq \nu_n$ alle Eigenwerte des mit der Spektraltransformation erhaltenen Eigenwertproblems und $\nu_{m,j} = \frac{1}{\lambda_i-\sigma}$ so angeordnet, dass $\nu_{m,1} \leq \nu_{m,2} \leq \ldots \leq \nu_{m,k}$ gilt. Für alle $j=1,\ldots,k$ existiert ein $t \in \{1,\ldots,n\}$, so dass $\nu_{t,j}=\nu_t$ ist. Dabei approximieren die $\tilde{\lambda}_j$ mit $j \in \{1,\ldots,k\}$ die Eigenwerte, die nahe bei σ liegen. Diese Eigenwerte werden mit der Spektraltransformation am betragsgrößten, liegen also danach am rechten und linken Rand. Es soll nun

$$\left|\nu_{m,j}\right| \geq \left|\tilde{\lambda}_j\right| \quad \text{mit } j \in \{1,\ldots,k\}$$

gezeigt werden. Im ersten Fall soll $\tilde{\lambda}_i$ eine Approximation für den linken Teilbereich sein, also kleiner als σ und deswegen kleiner als Null. Mit dem Minimum-Maximumprinzip von Courant-Fischer (vgl. A.4) ergibt sich folgende Abschätzung

$$\nu_{m,\ell} = \nu_m \leq \min_{S_m \subset X_k} \max_{v^h \in S_m} R\left(v^h\right) = \tilde{\lambda}_\ell \quad \Longrightarrow \quad |\nu_{m,\ell}| \geq \left|\tilde{\lambda}_\ell\right|$$

mit S_m, einem m-dimensionalen Unterraum der Lanczos-Vektoren. Im zweiten Fall ist es nun eine Approximation für den rechten Teil, somit größer σ und Null. Wiederum mit dem Minimum-Maximumprinzip von Courant-Fischer berechnet sich

$$\nu_{m,\ell} = \nu_m \geq \max_{S_{n-m+1} \subset X_k} \min_{v^h \in S_{n-m+1}} R\left(v^h\right) = \tilde{\lambda}_\ell \quad \Longrightarrow \quad |\nu_{m,\ell}| \geq \left|\tilde{\lambda}_\ell\right|$$

mit S_{n-m+1}, einem $n-m+1$-dimensionalen Unterraum der Lanczos-Vektoren. Dieser Unterraum existiert, da hier die größten Eigenwerte approximiert werden und daher m nahe bei der Dimension n liegt. Also ergibt sich insgesamt

$$\begin{aligned} \left|\frac{1}{\lambda_i - \sigma}\right| \geq \left|\tilde{\lambda}_i\right| &\Longrightarrow |\lambda_i - \sigma| \leq \left|\frac{1}{\tilde{\lambda}_i}\right| \\ \left|\lambda_i - \left(\frac{1}{\tilde{\lambda}_i} + \sigma\right)\right| \leq \left|\frac{1}{\tilde{\lambda}_i}\right| |\lambda_i - \sigma|\, \beta_{k+1} |s_{ik}| &\leq \frac{1}{\tilde{\lambda}_i^2} \beta_{k+1} |s_{ik}| \,. \end{aligned} \tag{4.1}$$

Damit kann der Fehler für alle $i \in \{1, \ldots, k\}$ einfach, ohne dass man die geschätzten Eigenvektoren berechnet und ohne Matrix-Vektor-Multiplikation, abgeschätzt werden (vgl. [NOPEJ87]).
Die Informationen für die gesuchten Eigenwerte stecken im zufällig gewählten Startvektor, deshalb benötigt man manchmal viele Iterationsschritte, bis die gesuchten Eigenwerte konvergieren. Durch die hohe Iterationsanzahl entsteht ein hoher Speicheraufwand. Durch einen Neustart während des Algorithmus kann dieser Speicheraufwand beschränkt werden. Beim impliziten Neustart ([Sor95]) kombiniert man einen k-Schritt Lanczos-Algorithmus mit dem QR-Algorithmus mit Shift. Man will k Eigenwerte bestimmen und startet mit einer Faktorisierung

$$CX_m = X_m T_m + f_m e_m^t \quad \text{mit} \quad f_m = \beta_{i+1} v_{i+1} \,,$$

wobei $m = k + p$ ist.
Eine Möglichkeit zur Auswahl der Shifts ist die „Exact Shift Strategy" , dabei teilt man die Eigenwerte von der Matrix T_k in gewünschte und ungewünschte Eigenwerte auf, und p ungewünschte Eigenwerte werden als

Algorithmus 4.3 Implicitly restarted Lanczos

Input: (C, X_m, T_m, f_m)
Suche p Shifts $\mu_1, \ldots, \mu_p$
$q^T = e_m^t$
for $i = 1 : p$ **do**
  $[Q_j, R_j] = qr\,(T_m - \mu_i I)$
  $T_m = Q_j^t T_m Q_j$
  $X_m = X_m Q_j$
  $q = q^t Q_j$
end for
$f_k = v_{k+1}\tilde{\beta}_k + f_m \sigma_k$ mit $\sigma_k = q(k)$ und $\tilde{\beta}_k = T_m(k, k+1)$
$X_k = X_m(1 : n, 1 : k)$
$T_k = T_k(1 : k, 1 : k)$
Berechne p zusätzliche Schritte des Lanczos-Algorithmus,
damit man ein neues X_m, T_m und f_m erhält und
wiederhole den Algorithmus.

Shifts gewählt. Man dämpft dabei die Komponenten der ungewünschten Eigenwerte im Startvektor. Bei diesem Verfahren spielen mehrfache Eigenwerte auch keine Rolle und werden entsprechend ihrer Vielfachheit gefunden.

4.3 Eigenwertberechnung

Bei der ersten Implementation des Lanczos-Algorithmus entdeckte man ein numerisches Problem. Dabei ging nach ein paar Lanczos-Schritten die Orthogonalität der Vektoren v_i verloren. Für Abbildung 4.3 wurde der Lanczos-Algorithmus 100-mal durchlaufen und $v_1^t B v_k$ mit $k = 2, \ldots, 100$ berechnet. Dies müßte bei exakter Berechnung aufgrund der B-Orthogonalität immer Null ergeben.
Dies führte zu „falschen“ Eigenwerten. Dabei gibt es zwei Typen von falschen Eigenwerten. Der erste Typ sind Kopien von „richtigen“ Eigenwerten und der zweite Typ sind völlig falsche Zahlenwerte. Die falschen Zahlenwerte verschwinden meistens beim Vergrößern der Lanczos-Matrix, um dann an einer anderen Stelle wieder aufzutauchen. Für dieses Problem werden nun zwei Lösungen vorgestellt.

Man kann die „falschen“ Eigenwerte „herausfiltern“. Dabei gilt für die Eigenwerte, die „falsch“ sind, dass die Eigenwerte von T_m nahe bei Ei-

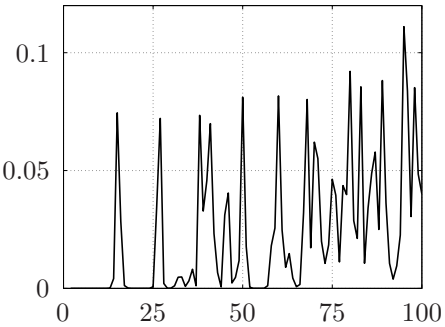


Abbildung 4.4: Fehlende Orthogonalität zu v_1

genwerten von $\tilde{T}_m$ liegen. Dabei ist

$$\tilde{T}_m = \begin{cases} \tilde{T}_m(i,i) = \alpha_{i+1} & \text{für } i = 1:m-1 \\ \tilde{T}_m(i,i+1) = \beta_{i+2} & \text{für } i = 1:m-2 \\ \tilde{T}_m(i+1,i) = \beta_{i+2} & \text{für } i = 1:m-2 \end{cases} .$$

$\tilde{T}_m$, also T_m ohne die erste Zeile und erste Spalte. Damit erhält man den Algorithmus 4.4.

Algorithmus 4.4 „Herausfiltern“ der „falschen“ Eigenwerte

berechne die Eigenwerte λ_i und $\tilde{\lambda}_i$ von T_m bzw. $\tilde{T}_m$
for $i = 1:m$ **do**
  if falls λ_i einfach ist **then**
    if ein $k \in [1, m-1]$ existiert mit $|\lambda_i - \tilde{\lambda}_k| < \max(1, \lambda_i) \cdot \text{Tol}$ **then**
      λ_i ist „falscher“ Eigenwert
    end if
  end if
end for

Hierbei filtert man zwar die Eigenwerte heraus, die völlig falsche Zahlenwerte annehmen, allerdings kann die Vielfachheit der Eigenwerte aufgrund möglicher Kopien nicht herausgefunden werden. Dieses Problem wird in den nächsten Absätzen behoben.

Eine andere Möglichkeit besteht darin, die Lanczos-Vektoren während des Algorithmus zu orthogonalisieren. Dabei gibt es drei Möglichkeiten:

- vollständige Orthogonalisierung
- selektive Orthogonalisierung
- partielle Orthogonalisierung

Bei der vollständigen Orthogonalisierung werden die neu berechneten Lanczos-Vektoren gegen alle vorher berechneten Lanczos-Vektoren orthogonalisiert. Dies kann z. B. mit Hilfe vom klassischen Gram-Schmidt-Verfahren erreicht werden.

$$v'_{j+1} = v_{j+1} - \sum_{k=1}^{j} \left(v_k^t B v_{j+1}\right) v_k$$

Die vollständige Orthogonalisierung ist ein sicheres aber auch teures Verfahren. Dadurch ist jeder neu berechnete Vektor orthogonal zu den bisher berechneten Lanczos-Vektoren, und die benötigte Orthogonalität ist wieder hergestellt. Der Verlust der Orthogonalität aufgrund der Rechnerungenauigkeit spielt also keine relevante Rolle mehr.

Bei der vollständigen Orthogonalisierung ist der Aufwand groß, da in jedem Schritt der neu berechnete Lanczos-Vektor mit allen vorher berechneten Lanczos-Vektoren orthogonalisiert wird. Bei der selektiven Orthogonalisierung wird zwar auch in jedem Iterationsschritt orthogonalisiert, allerdings nur mit bestimmten ausgewählten Vektoren. Man stoppt hierbei an gewählten Stellen. Bei einem Stopp wird das T_k diagonalisiert und die Fehlerschranken der noch nicht berechneten Ritz-Vektoren werden berechnet. Die Ritz-Vektoren, die hinreichend genau sind, werden bestimmt, orthogonalisiert und abgespeichert. Dabei spielt es keine Rolle, ob die Ritz-Werte zu den berechneten Ritz-Vektoren im gesuchten Frequenzbereich liegen. Bis zum nächsten Stopp werden alle neu berechneten Lanczos-Vektoren nur gegen die vorher gespeicherten Ritz-Vektoren orthogonalisiert. Dadurch wird der Aufwand in der Berechnung erheblich reduziert und die Rechenzeit vermindert. Für den Zeitpunkt des Stopps gibt es mehrere Möglichkeiten. Man kann ein fest vorgegebenes m wählen und nach allen m Schritten eine Pause machen. Diese Methode führt aber in der Praxis zu nicht zufriedenstellenden Ergebnissen und wird daher nicht angewandt. Es ist besser, variable Stopps einzuführen. Dabei kann die fehlende Orthogonalität der berechneten Vektoren auf unterschiedliche Art abgeschätzt werden (vgl. [PS79]) und bei zu großer Abweichung stoppt man. Eine andere Möglichkeit der Orthogonalisierung wird im nächsten Absatz eingeführt.

Bei der partiellen Orthogonalisierung (vgl. [Sim84]) wird nicht gegen die Ritz-Vektoren orthogonalisiert, sondern gegen die berechneten Lanczos-Vektoren. Es soll $w_{j,k} = w_{k,j} = v_j^t B v_k$ sein.

Algorithmus 4.5 Partielle Orthogonalisierung

- Lanczos-Schritt $v_{j+1} = Cv_j - \alpha_j v_j - \beta_j v_{j-1}$
- Berechne neue $w_{j+1,k}$ für $k = 1, \ldots, j$
- Wähle Menge $L(j) = \{k : 1 \leq k \leq j\}$ und berechne

$$v_{j+1} = v_{j+1} - \sum_{k \in L(j)} \left(v_k^t B v_{j+1}\right) v_k$$

Dabei orthogonalisiert man einen neu berechneten Lanczos-Vektor v_{j+1} nur, falls ein k existiert, für das $\left|v_{j+1}^t B v_k\right| > \sqrt{\epsilon}$ gilt. ϵ ist die Maschinengenauigkeit, und man erreicht damit die Semiorthogonalität der Lanczos-Vektoren. Durch die Semiorthogonalität bleiben die wesentlichen Eigenschaften des Lanczos-Algorithmus erhalten. Muss man v_{j+1} orthogonalisieren, dann muss dies auch für den nächsten Vektor v_{j+2} durchgeführt werden. Dies ist notwendig, da $\left|v_j^t B v_k\right|$ schon nahe bei $\sqrt{\epsilon}$ gelegen ist und v_j für die Berechnung von v_{j+2} benötigt wird. Allerdings reicht es nicht aus, wenn man für die Menge $L(j)$ die Vektoren v_k mit $k = 1, \ldots, j$ verwendet, für die $\left|v_{j+1}^t B v_k\right| > \sqrt{\epsilon}$ gilt, sondern man muss auch einige benachbarte Vektoren der entsprechenden Vektoren hinzufügen, da diese v_k beeinflusst haben.

Kapitel 5

Multigrid-Löser

Im Algorithmus 4.2 muss im ersten Schritt

$$w_i = (A - \sigma B)^{-1} B v_i$$

die Matrix $A - \sigma B$ invertiert werden, bzw. das lineare Gleichungssystem

$$\underbrace{(A - \sigma B)}_{D} w_i = \underbrace{B v_i}_{b} \quad \Longrightarrow \quad D w_i = b$$

muss nach w_i aufgelöst werden. Dabei gibt es mehrere Ansatzmethoden. Eine Methode sind die direkten Löser, wie z. B. die Gaußsche Eliminationsmethode, die QR-Zerlegung oder die Cholesky-Zerlegung. Darüber hinaus gibt es die linearen Iterationsverfahren, zu denen die Richardson-Iteration, die Jacobi-Iteration oder die Gauss-Seidel-Iteration gehören, und die nichtlinearen Iterationsverfahren, wie z. B. das konjugierte Gradientenverfahren. Da bei diesen Verfahren allerdings der Aufwand zu hoch ist, sind diese bei großen Matrizen nicht mehr rentabel.
Im ersten Abschnitt des Kapitels werden die wichtigsten Glätter eingeführt und deren Funktion erklärt. In den folgenden Abschnitten wird dann direkt auf den Multigrid-Löser eingegangen.

5.1 Glätter

Klassische Iterationsverfahren zum Lösen eines linearen Gleichungssystems $Ax = b$ basieren in der Regel auf einer Fixpunktgleichung

$$x_{k+1} = F(x_k) \ .$$

Möchte man die richtige Lösung finden, so muss F genau einen Fixpunkt x^* besitzen, der gleichzeitig Lösung der ursprünglichen Gleichung ist. F bekommt man durch Umformungen

$$\begin{aligned} Ax = b \quad &\Longleftrightarrow \quad G^{-1}(b - Ax) = 0 \\ &\Longleftrightarrow \quad F(x) = \left(I - G^{-1}A\right) x + G^{-1}b = x \end{aligned}$$

mit einer invertierbaren Matrix G. Für die Wahl von G zerlegt man im Allgemeinen die Matrix $A = L+D+R$. Dabei ist D eine Diagonalmatrix mit den Einträgen auf der Diagonalen von A, L ist eine linke untere und R eine rechte obere Dreiecksmatrix, wobei bei L und R die Diagonale auf Null gesetzt wird:

$$L = \begin{pmatrix} 0 & \dots & \dots & 0 \\ a_{2,1} & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ a_{n,1} & \dots & a_{n,n-1} & 0 \end{pmatrix}, \quad R = \begin{pmatrix} 0 & a_{1,2} & \dots & a_{1,n} \\ \vdots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & a_{n-1,n} \\ 0 & \dots & \dots & 0 \end{pmatrix}.$$

Ein lineares Iterationsverfahren

$$x_{k+1} = \underbrace{\left(I - G^{-1}A\right)}_{Q} x_k + G^{-1}b$$

konvergiert, falls der Spektralradius der Iterationsmatrix Q

$$\varrho = \max\left\{|\lambda| : Qu = \lambda u, u \neq 0\right\}$$

kleiner als eins ist. Dabei ist die Konvergenz langsam, wenn ϱ nahe bei eins liegt.
Setzt man $G = I$, so erhält man die Richardson-Iteration

$$x_{k+1} = (I - A)\, x_k + b\,.$$

Diese konvergiert bei symmetrisch positiv definiten Matrizen, falls der maximale Eigenwert von A kleiner als zwei ist. Ein etwas komplizierteres Iterationsverfahren, das Jacobi-Verfahren, bekommt man für $G = D$

$$x_{k+1} = \left(I - D^{-1}A\right) x_k + D^{-1}b\,.$$

Dieses Verfahren konvergiert bei jeder strikt diagonaldominanten Matrix, d.h. für

$$|a_{i,i}| > \sum_{i \neq j} |a_{i,j}|\ .$$

Um das Gauss-Seidel-Verfahren zu erhalten, muss man $G = D+L$ setzen, also

$$\begin{aligned} x_{k+1} &= \left(I - (D+L)^{-1}A\right) x_k + (D+L)^{-1}b \\ &= -(D+L)^{-1}Rx_k + (D+L)^{-1}b\,. \end{aligned}$$

Um die verhältnismäßig komplizierte Invertierung $(D+L)^{-1}$ zu umgehen, kann die Gauss-Seidel-Iteration auch zeilenweise berechnet werden

$$x_{k+1} = -D^{-1}\left(Lx_{k+1} + Rx_k\right) + D^{-1}b\,.$$

Die Konvergenz ist für symmetrisch positiv definite Matrizen gesichert. Die Konvergenz aller bisherigen Iterationsverfahren kann mit Relaxation beschleunigt werden. Dabei führt man einen Relaxationsparameter ω ein und ersetzt x_{k+1} mit

$$x_{k+1} = x_k + \omega\left(x_{k+1} - x_k\right)\,.$$

Damit erhält man die folgenden Iterationsverfahren:

$x_{k+1} = (I - \omega A)\, x_k + \omega b$	Richardson
$x_{k+1} = \left(I - \omega D^{-1}A\right) x_k + \omega D^{-1}b$	Jacobi
$x_{k+1} = x_k - \omega D^{-1}\left(Lx_{k+1} + Rx_k + Dx_k\right) + \omega D^{-1}b$	SOR

Der SOR-Algorithmus konvergiert bei symmetrisch positiv definiten Matrizen genau dann, wenn $0 < \omega < 2$ ist. Die optimale Wahl liegt bei $\omega > 1$. Nutzt man die Symmetrie der Matrix aus, so erhält man den SSOR-Löser, dabei wird allerdings die Konvergenz der Iteration meistens nicht beschleunigt. Will man eine Gleichung relativ genau lösen, so wird eine zu große Anzahl von Iterationen benötigt. Damit sind diese Verfahren nicht rentabel. Allerdings dämpfen diese Verfahren vor allem die hochfrequentigen Anteile im Fehler. Diese Eigenschaft der Verfahren wird Glättungseigenschaft genannt. Hat der Fehler nur noch niederfrequentige Anteile, können diese auch gut auf einem gröberen Gitter dargestellt werden. Das neue Gleichungssystem kann nun auf dem gröberen Gitter mit weniger Unbekannten gelöst werden.
Die Schaubilder in Abbildung 5.2 stellen die relaxierte Richardson-Iteration mit 75 Unbekannten dar. Es wurden lineare B-Splines über dem Intervall $[0,1]$ für das Beispielproblem

$$\begin{aligned} -\Delta u &= 1 \quad x \in (0,1) \\ u(0) &= 0 \text{ und } u(1) = 0 \end{aligned}$$

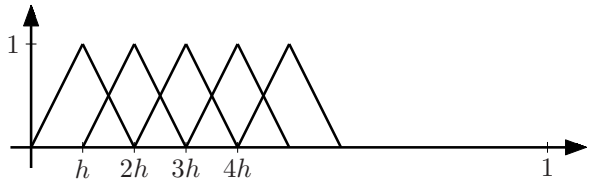


Abbildung 5.1: Basisfunktionen

genommen. Da jeweils ein Knoten genau bei Null und eins lag, war eine Gewichtsfunktion nicht notwendig.
Mit diesen Ansatzfunktionen erhielt man die folgende Galerkin-Matrix A mit $a_{i,j} = \int \nabla b_i \cdot \nabla b_j$ und den Lastvektor b mit $b_i = \int f \cdot b_i$.

$$A = \frac{1}{h} \begin{pmatrix} 2 & -1 & & & \\ -1 & \ddots & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & -1 \\ & & & -1 & 2 \end{pmatrix}, \quad b = h \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}$$

Für ω wurde $\|A\|_\infty^{-1}$ genommen und u_0 wurde zufällig gewählt.

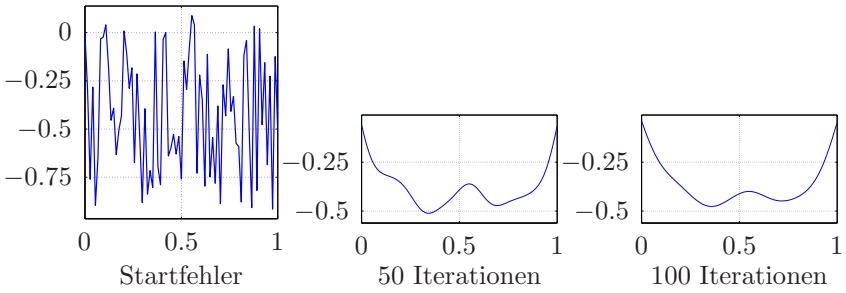


Abbildung 5.2: Glättungseigenschaft der Richardson-Iteration

Dabei wurde in Abbildung 5.2 immer der Fehler $u - u_h^\ell$ nach ℓ Iterationen abgebildet und man erkennt deutlich die Glättungseigenschaft der Richardson-Iteration.

Hat man eine nicht positiv definite Matrix im Sparse-Format beim zu

lösenden linearen Gleichungssystem $Ax = b$, konvergieren viele Löser nicht mehr. Daher wird noch auf einen anderen Glätter eingegangen. Dies ist die Klasse der Glätter, die auf unvollständiger LU-Zerlegung (ILU) basiert. Bei der LU-Zerlegung $A = LU$ löst man erst das Gleichungssystem $Ly = b$, danach $Ux = y$, um den Lösungsvektor x zu bestimmen. Bei der ILU-Zerlegung erhält man noch eine „Restmatrix" R mit $A = \widetilde{L}\widetilde{U} - R$, wobei die beiden Matrizen $\widetilde{L}$ und $\widetilde{U}$ auch Sparse-Format aufweisen müssen. Im einfachsten Fall kann man davon ausgehen, dass die beiden Matrizen an den Stellen Nulleinträge haben, an denen A Nulleinträge hatte. Man könnte die Einträge wie bei einer LU-Zerlegung berechnen, und danach die Einträge von $\widetilde{L}$ und $\widetilde{U}$ an den Stellen durch Null ersetzen, bei denen auch bei A Nullen stehen. Dadurch muss mit einer „Restmatrix" R ausgeglichen werden.

Die ILU-Zerlegung hat im Allgemeinen schlechte Konvergenzeigenschaften für das Lösen von linearen Gleichungssystemen. Doch die Glättungseigenschaften sind in weiten Bereichen so gut, dass die Zerlegung als Glätter angewendet werden kann (vgl. [TOS01]). Bei den verschiedenen ILU-Zerlegungen wird eine gewisse Besetzungsstruktur $\boldsymbol{P}$ für die Matrizen $\widetilde{L}$ und $\widetilde{U}$ vorgegeben. Nur an diesen fest vorgegebenen Stellen dürfen Einträge ungleich Null stehen. Ein möglicher Algorithmus ist Algorithmus 5.1 aus [Mei05].

Eine unvollständige LU-Zerlegung wird ILU(0) (vgl. [Saa03]) genannt, falls man für die Besetzungsstruktur genau die Nicht-Nulleinträge der Matrix A nimmt. Dabei hat diese Faktorisierung eine schlechte Konvergenzrate. Genauere unvollständige LU-Zerlegungen sind effizienter und rentabler. Dies ist bei der ILU(p)-Zerlegung der Fall. Dabei wird die Nullmenge für ILU(1) gebildet, indem man die ILU(0)-Zerlegung von der Matrix $A = L_0U_0 - R_0$ berechnet und die Nullmenge von L_0U_0 nimmt. Bei ILU(2) wird die Nullmenge mit Hilfe des Produktes der Matrizen L_1U_1 bestimmt, die aus der ILU(1)-Zerlegung entstehen.

Bei den Basisfunktionen aus dem linken Bild von Abbildung 5.3 gibt jeder Punkt die linke untere Ecke eines Splines vom Grad eins an. Ein Träger einer Basisfunktion wurde grau hinterlegt. Damit ergibt sich die Belegungsstruktur der Matrix, die im rechten Bild dargestellt ist.

In Abbildung 5.4 erkennt man die Belegungsstrukturen verschiedener ILU(p)-Faktorisierungen und sieht die Zunahme der Nicht-Nulleinträge in den Matrizen L_p und U_p bei wachsendem p.

Algorithmus 5.1 ILU-Zerlegung

for $i = 1:1:n$ **do**
  for $k = 1:1:n$ **do**
    if $(k,i) \in \boldsymbol{P}$ **then**

$$\ell_{k,i} = a_{k,i} - \sum_{\substack{m=1 \\ (k,m) \in \boldsymbol{P} \\ (m,i) \in \boldsymbol{P}}}^{i-1} \ell_{k,m} u_{m,i}$$

    end if
  end for
  for $k = i+1:1:n$ **do**
    if $(i,k) \in \boldsymbol{P}$ **then**

$$u_{i,k} = \frac{1}{\ell_{i,i}} \left(a_{i,k} - \sum_{\substack{m=1 \\ (i,m) \in \boldsymbol{P} \\ (m,k) \in \boldsymbol{P}}}^{i-1} \ell_{i,m} u_{m,k} \right)$$

    end if
  end for
end for

5.2 Idee und Algorithmus

Alle eingeführten Glätter haben die Eigenschaft, dass sie die hochfrequentigen Fehleranteile stark dämpfen und die niederfrequentigen Fehleranteile kaum verbessern. Falls man nun die Glätter mit einem Verfahren kombiniert, welches die niederfrequenten Fehleranteile minimiert, würde der Löser schnell konvergieren. Nach mehreren Glättungsschritten ist der Fehler genügend glatt und enthält hauptsächlich einen langwelligen Frequenzanteil, wie in Abbildung 5.5 dargestellt. Dabei wurde die lineare Approximation auf zwei verschiedenen Gitterweiten eingezeichnet und man erkennt, dass der Fehler auch auf dem Gitter mit doppelter Gitterweite gut approximiert wird. Dieser langwellige Frequenzanteil im Fehler verschwindet mit den linearen Iterationsverfahren nicht wesentlich. Um ihn einfacher zu berechnen, würde es ausreichen, wenn man das lineare Gleichungssystem

$$A_h u_{e,h} = e_h$$

auf der doppelten Gitterweite

$$A_{2h} u_{e,2h} = e_{2h}$$

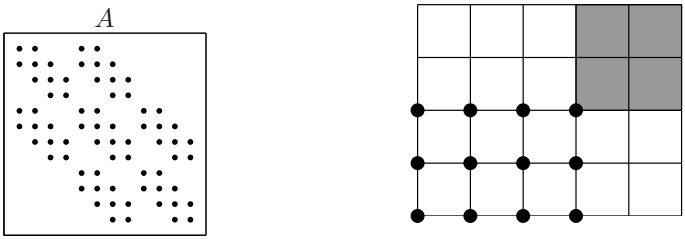


Abbildung 5.3: Basisfunktion und Matrix

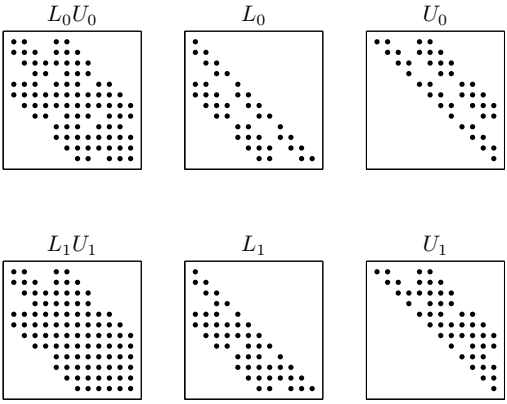


Abbildung 5.4: Beispiel einer Belegungsstruktur für ILU

löst. Dabei soll u_h die schon berechnete fehlerhafte Lösung und $e_h = A_h u_h - b_h$ sein. Man müsste die berechnete Lösung $u_{e,2h}$ auf das feinere Gitter projizieren $\tilde{u}_{e,2h}$ und würde eine verbesserte Lösung auf dem feineren Gitter $\tilde{u} = u_h - \tilde{u}_{e,2h}$ erhalten.

Dabei werden im Algorithmus 5.2 die Abkürzungen aus Tabelle 5.1 verwendet.

Da man ein regelmäßiges Gitter hat, ist hierbei die Gitterverfeinerung sehr einfach. Die Restriktion und Prolongation wird bei den B-Splines und bei den gewichteten B-Splines mit der Gleichung 2.2 auf Seite 26 berechnet. Bei multivariaten Spline-Räumen verändern sich die Faktoren entsprechend.

β	Parameter für den rekursiven Aufruf
lev	momentane Stufe ($lev = 1$ ist feinstes Gitter)
$u_{lev} = S^{\alpha}(u_{lev}, b_{lev})$	α-malige Anwendung des Glätters mit der bisher berechneten Lösung u_{lev} und rechter Seite b_{lev}
$u_{lev+1} = R(u_{lev})$	Restriktion, d.h. Einschränkung von u_{lev} auf ein gröberes Gitter
$u_{lev-1} = P(u_{lev})$	Prolongation, d.h. Projektion von u_{lev} auf ein feineres Gitter
lev_{max}	Stufenanzahl

Tabelle 5.1: Abkürzungen für statischen Multigrid-Algorithmus

Algorithmus 5.2 Statischer Multigrid-Algorithmus

$u_{lev} = \text{Multigrid}(u_{lev}, b_{lev}, lev)$
$u_{lev} = S^{\alpha}(u_{lev}, b_{lev})$
$r = R(A_{lev}u_{lev} - b_{lev})$
if $lev + 1 < lev_{max}$ **then**
  for $j = 1 : \beta$ **do**
    $v_{lev+1} = \text{Multigrid}(0, r, lev + 1)$
  end for
else if $lev + 1 = lev_{max}$ **then**
  $v_{lev+1} = A^{-1}_{lev+1} r$
end if
$u_{lev} = u_{lev} - P(u_{lev+1})$
$u_{lev} = S^{\alpha}(u_{lev}, b_{lev})$

Lemma 5.1: *Multivariate Subdivision*
Ein m-dimensionaler B-Spline b_{2h} vom Ansatzgrad n auf einem Gitter mit Gitterweite $2h$ kann als Linearkombination

$$b_{\ell,2h} = \sum_{k \in K} s_{k-2\ell} b_{k,h}\,, \quad s_{\alpha} = 2^{-nm} \prod_{\nu=1}^{m} \binom{n+1}{\alpha_{\nu}} \tag{5.1}$$

durch B-Splines b_h auf einem Gitter mit Gitterweite h dargestellt werden. Dabei setzt man $\binom{n+1}{a} = 0$ für $a > n + 1$ oder $a < 0$.

Bei den WEB-Splines sieht es nicht wesentlich komplizierter aus. Man muss noch die Erweiterungskoeffizienten berücksichtigen.

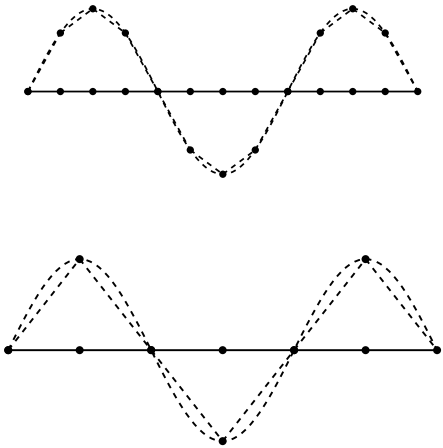


Abbildung 5.5: Lineare Approximation auf verschiedenen Gitterweiten

Für die in Abbildung 5.6 dargestellten B-Splines vom Grad zwei mit der Gitterweite h mit acht Basisfunktionen und Gitterweite $2h$ mit drei Basisfunktionen erhält man zum Beispiel mit der Gleichung 2.2 auf Seite 26 die transponierte Prolongationsmatrix

$$P^t = \begin{pmatrix} \frac{1}{4} & \frac{3}{4} & \frac{3}{4} & \frac{1}{4} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{4} & \frac{3}{4} & \frac{3}{4} & \frac{1}{4} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{4} & \frac{3}{4} & \frac{3}{4} & \frac{1}{4} \end{pmatrix} .$$

Damit erhält man einen Vektor $u_h \in \mathbb{B}_h^n$, indem der Vektor $u_{2h} \in \mathbb{B}_{2h}^n$ mit der Prolongationsmatrix multipliziert wird. Es gilt dabei $\mathbb{B}_{2h}^n \subset \mathbb{B}_h^n$.

$$u_h = P u_{2h} \,, \quad p_{k,\ell} = s_{k-2\ell} \text{ aus Gleichung (5.1)}$$

Für die Restriktion wird oft als Matrix die transponierte Prolongationsmatrix, also $R = P^t$ genommen. Führt man zusätzlich nur die Gewichtung ein, so ändert sich an den Matrizeneinträgen, im Gegensatz zu den WEB-Splines, nichts. Dabei erhält man die Basis der WEB-Splines, indem mit der Erweiterungsmatrix E_h die Basis der gewichteten B-Splines multipliziert wird. Die Erweiterungsmatrix E_h enthält die vorher bestimmten Koeffizienten $e_{i,j}$ mit Gleichung 2.5 auf Seite 30. Will man

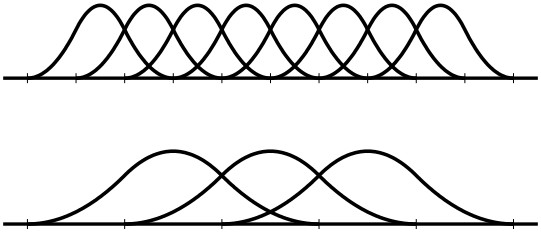


Abbildung 5.6: Beispiel für die Restriktion

nun die Prolongationsmatrix für die WEB-Splines bestimmen, wird als erstes die Erweiterung mit E_{2h}^t „rückgängig" gemacht, und danach multipliziert man mit der Prolongationsmatrix $P(I,:)$. Man verwendet nur die Zeilen der inneren $i \in I$ B-Splines (vgl. 2.3). Nun multipliziert man mit einer Diagonalmatrix D, bei der die $w(x_i)$ aus Gleichung (5.1) auf der Diagonalen stehen. Dabei werden wieder nur die $i \in I$ berücksichtigt, die zu inneren Splines gehören. Dies ergibt also ein Produkt aus drei Matrizen D, $P(I,:)$ und E_{2h} in der richtigen Reihenfolge.

$$\widetilde{P} = DP(I,:)E_{2h}^t$$

Die mathematische Formel für die Matrixeinträge von $\widetilde{P}$ ist

$$\widetilde{p}_{i,\ell} = \frac{w\left(x_{i,h}\right)}{w\left(x_{\ell,2h}\right)} \left(s_{i-2\ell} + \sum_{j \in J_{2h}(\ell)} e_{\ell,j,2h} s_{i-2j} \right) .$$

Als neue Restriktionsmatrix wird wieder $\widetilde{R} = \widetilde{P}^t$ genommen. Allerdings lässt sich aufgrund der Erweiterung die Basis bei doppelter Gitterweite $w^e \mathbb{B}_{2h}^n$ nicht exakt durch die Basis $w^e \mathbb{B}_h^n$ mit $\widetilde{P}$ darstellen, was bei den gewichteten B-Splines noch der Fall war.
Im Algorithmus 5.2 wird das Gleichungssystem in Zeile neun exakt gelöst, wenn man auf dem gröbsten Gitter angelangt ist. Da man immer nur für den bisherigen Fehler $e_{lev} = A_{lev} u_{lev} - b_{lev}$ in der Gleichung $e_{lev+1} = A_{lev+1} u_{lev+1}$ den Vektor u_{lev+1} auf dem gröberen Gitter berechnen will, wird für den Startvektor der Nullvektor genommen. Für den Parameter β erhält man unterschiedliche Formen des Stufenwechsels, z. B. für $\beta = 1$ den V-Zyklus und für $\beta = 2$ den W-Zyklus, vgl. Abbildung 5.8.
Der Speicher- und Rechenaufwand wächst mit der Anzahl der Unbekannten linear an, und die Konvergenzrate ist unabhängig von der Anzahl

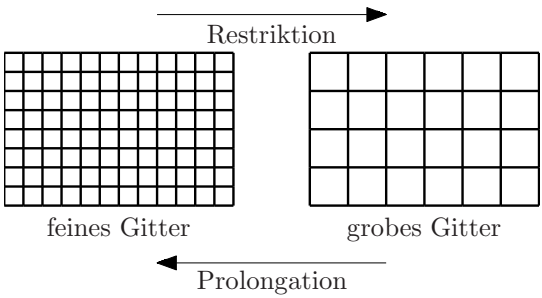


Abbildung 5.7: Gitterverfeinerung

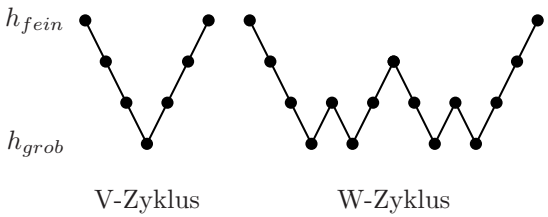


Abbildung 5.8: Verschiedene Multigrid-Zyklen

der Unbekannten, deshalb ist der Multigrid-Algorithmus ein optimales Lösungsverfahren. In Abbildung 5.9 sind zwei Gitter mit Gebiet eingezeichnet.

5.3 Verbesserung des Algorithmus

Zum ursprünglichen Multigrid-Algorithmus gibt es noch Änderungen des Algorithmus. Zwei Möglichkeiten werden in diesem Abschnitt näher erläutert.
Die erste mögliche Änderung von Algorithmus 5.2 ist der dynamische Multigrid-Zyklus. Dieser wird ausführlich in der Veröffentlichung [Bra77] beschrieben. Eine Änderung in dieser Veröffentlichung ist, dass die Anzahl der Glättungsschritte nicht fest vorgelegt ist. Es wird immer wieder untersucht, ob die Glättung noch effektiv ist oder die Fehlerreduktion zu gering ist. Danach wird entschieden, ob man weiter glättet oder ein Gitterwechsel stattfindet. Außerdem ist die Reihenfolge der Prolongation

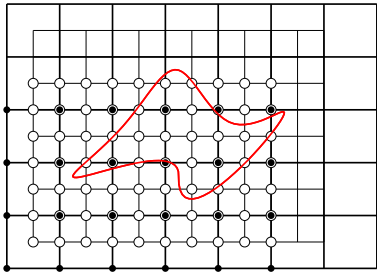


Abbildung 5.9: Gebiet mit zwei verschiedenen Gittern

und Restriktion nicht bestimmt, sondern kann variabel ablaufen. Eine mögliche Abfolge ist in Abbildung 5.10 dargestellt. Ist der Fehler klein, dann geht man auf ein feineres Gitter. Ist der Fehler nach dem Glätten noch zu groß, wird auf ein gröberes Gitter gewechselt.

Dabei hat man zwei neue Steuerparameter. ε_{lev} ist die relative Lösertoleranz, die auf jedem Gitter variieren kann. Damit man auf ein feineres Gitter gelangt, muss der relative Fehler $\frac{\text{err}}{\|b_{lev}\|}$ mit $\text{err} = \|A_{lev}u_{lev} - b_{lev}\|$ auf dem gröberen Gitter kleiner sein als die vorgegebene Toleranz. Der andere Steuerparameter ϱ gibt an, wie lange geglättet werden soll, wann ein Glättungsschritt noch sinnvoll ist und wann der iterative Löser keine wesentliche Verbesserung mehr bringt. Durch die erhöhte Flexibilität im Gitterwechsel kann der Löser effizienter werden. Wenn man im Gegensatz zum ursprünglichen Algorithmus öfters den Fehler err berechnet, wird hierfür mehr Rechenzeit benötigt. Dabei spielt auch die Wahl der beiden Steuerparameter eine nicht unwesentliche Rolle.

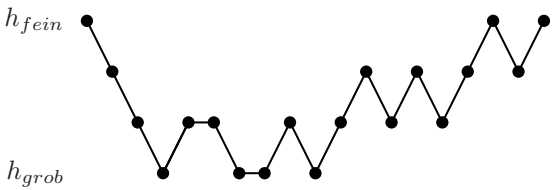


Abbildung 5.10: Dynamischer Multigrid

Algorithmus 5.3 Dynamischer Multigrid-Algorithmus

$lev = 1$
while $lev \neq 1$ und $\frac{\text{err}}{\|b_{lev}\|} > \varepsilon_{lev}$ **do**
  $\text{err} = \|A_{lev} u_{lev} - b_{lev}\|$
  $\text{err}_{old} = 2 \cdot \text{err} \cdot \varrho$
  while $\frac{\text{err}}{\|b_{lev}\|} > \varepsilon_{lev}$ und $\frac{\text{err}}{\text{err}_{old}} < \varrho$ und $lev \neq lev_{max}$ **do**
    $\text{err}_{old} = \text{err}$
    $u_{lev} = S^{\alpha}(u_{lev}, b_{lev})$
    $\text{err} = \|A_{lev} u_{lev} - b_{lev}\|$
  end while
  if $lev = lev_{max}$ **then**
    $u_{lev} = A_{lev}^{-1} b_{lev}$
  end if
  if if $\frac{\text{err}}{\|b_{lev}\|} > \varepsilon_{lev}$ **then**
    $lev = lev + 1$
    $b_{lev} = R(A_{lev-1} u_{lev-1} - b_{lev-1})$
    $u_{lev} = 0$
  else
    $lev = lev - 1$
    $u_{lev} = u_{lev} - P(u_{lev+1})$
    $u_{lev} = S^{\alpha}(u_{lev}, b_{lev})$
  end if
end while

Bisher wurde die Berechnung des Algorithmus auf dem feinsten Gitter gestartet. In diesem Fall war schon eine Approximation der Lösung auf diesem Gitter bekannt. Da dies nicht oft der Fall ist, kann die Lösung auf dem gröbsten Gitter exakt berechnet werden, und man kann mit Hilfe der Prolongation auf immer feinere Gitter gelangen. Das feinste Gitter kann natürlich wieder auf verschiedenen Wegen erreicht werden; die Schrittabfolge ist nicht fest vorgegeben.
Beim Full-Multigrid (FMG), vgl. Algorithmus 5.4, ist die Startwertberechnung eine Nested-Iteration, man überträgt die Lösung von einem gröberen Gitter auf ein feineres Gitter. Dadurch kann die Effizienz der normalen Multigrid-Iteration noch verbessert werden und der FMG-Algorithmus ist meistens am effizientesten.
Dabei wird auf den Algorithmus 5.2 zugegriffen. Der Wert $iter_k$ mit dem Stufenindex k gibt die Anzahl der Aufrufe des Multigrid-Verfahrens auf jeder Gitterstufe an, bevor man ein feineres Gitter einbezieht. In der Ab-

Algorithmus 5.4 Full-Multigrid

$u_{max_{lev}} = A^{-1}_{max_{lev}} b_{max_{lev}}$
for $k = max_{lev} - 1 : -1 : 1$ **do**
  $u_k = P\,(u_{k+1})$
  for $j = 1 : iter_k$ **do**
    $u_k = \text{Multigrid}\,(u_k, b_k, k)$
  end for
end for

bildung 5.11 wurde beim Multigrid-Verfahren der V-Zyklus gewählt. Als $iter_k$ wurde auf der Stufe $k = max_{lev} - 1$ zwei und für alle anderen Stufen k eins genommen. Nimmt man beim ursprünglichen Multigrid-Verfahren ein anderes $\beta \neq 1$, dann sieht der FMG-Algorithmus entsprechend aus.

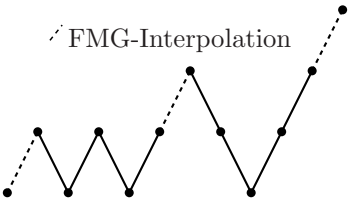


Abbildung 5.11: Mögliche Startwertberechnung

Kapitel 6

Implementation

In diesem Kapitel wird die Implementation des Eigenwertproblems

$$\begin{aligned} -\Delta u &= \lambda u \quad \text{in } \Omega \\ u &= 0 \quad \text{auf } \partial\Omega \end{aligned}$$

mit homogenen Dirichlet-Randbedingungen beschrieben. Dabei geht man den Weg über die Variationsformulierung, wie in Gleichung (1.7). Dieses Programm wurde mit beliebigem Ansatzgrad der WEB-Splines in MATLAB programmiert.

6.1 Programmaufbau

Das Eigenwertproblem wird mit dem Multigrid-Algorithmus gelöst. Hier arbeitet man auf mehreren Gitterebenen, wobei die feinste Gitterweite angegeben werden muss. Mit dieser Angabe werden als erstes auf dem feinsten Gitter die Zellen und deren Klassifikation (innere, Rand- oder äußere Zelle) bestimmt, und damit werden die gröberen Gitter berechnet. Im Folgenden gibt q die Anzahl der Gitterverfeinerungen an. Nach diesem Schritt kann die Massenmatrix und die Steifigkeitsmatrix auf dem feinsten Gitter aufgestellt werden. Da man Nullrandbedingungen vorgegeben hat, fällt das Randintegral weg. Dabei wird auf dem regelmäßigen Gitter über alle Zellen geschleift. Innerhalb dieser Schleife werden beim Ansatzgrad n die Skalarprodukte

$$a_{i,j} = \int_\Omega \nabla B_i \left(\nabla B_j\right)^t , \qquad b_{i,j} = \int_\Omega B_i B_j ,$$

für die in dieser Zelle relevanten $(n+1) \times (n+1)$ wB_h^n-Kombinationen berechnet. Man erhält insgesamt eine Vierfachschleife. Bei einem Gitter der Größe $m_1 \times m_2$ entstehen in dieser Schleife zwei vierdimensionale Matrizen $\widetilde{A}$ und $\widetilde{B}$ der Größe $m_1 \times m_2 \times (n+1) \times (n+1)$. Dabei werden die Integrale nur berechnet, falls es sich um eine innere Zelle oder eine Randzelle handelt, ansonsten sind sie Null. Die Integration erfolgt über Gauss-Integration, da es schwierig ist, die Integrale exakt zu bestimmen. Ein Integral wird durch eine endliche Summe approximiert

$$\int_a^b f(x)\,dx \approx \sum_{i=1}^{\ell} g_i f(x_i)\,.$$

Die Gewichte g_i für $i = 1, \ldots, \ell$ und Integrationspunkte x_i für $i = 1, \ldots, \ell$ stehen mit $a = -1$ und $b = 1$ in Tabelle 6.1. Dabei sind die Gauss-Formeln exakt für Polynome bis zum Grad $2\ell - 1$. Für andere Grenzen a und b müssen g_i und x_i transformiert werden:

$$\tilde{g}_i = \frac{b-a}{2} g_i\,, \quad \tilde{x}_i = a + \frac{b-a}{2}\,(x_i + 1)$$

ℓ	x_i	g_i
2	$\pm\frac{1}{3}\sqrt{3}$	1
3	0	$\frac{8}{9}$
	$\pm\frac{1}{5}\sqrt{15}$	$\frac{5}{9}$
4	$\pm\frac{1}{35}\sqrt{525 - 70\sqrt{30}}$	$\frac{1}{2} + \frac{1}{36}\sqrt{30}$
	$\pm\frac{1}{35}\sqrt{525 + 70\sqrt{30}}$	$\frac{1}{2} - \frac{1}{36}\sqrt{30}$

Tabelle 6.1: Gauss-Gewichte und Gauss-Punkte

Will man über einen n-dimensionalen Quader aus dem $\mathbb{R}^n$ integrieren, dann wird die Tensorprodukt-Gauss-Formel mit denselben Auswertungs-

punkten und den Gewichten

$$\int_{a_1}^{b_1} \cdots \int_{a_n}^{b_n} f(x_1, \ldots, x_n)\, dx_n \cdots dx_1 \approx \sum_{i_1=1}^{\ell_1} \cdots \sum_{i_n=1}^{\ell_n} \left(\prod_{k=1}^{n} \tilde{g}_{\ell, i_\ell} \right) f(\tilde{x}_{1,i_1}, \ldots, \tilde{x}_{n,i_n})$$

angewandt. Hat man eine Randzelle, müssen die Gauss-Punkte und Gauss-Gewichte entsprechend transformiert werden. Für jede Zelle werden zuerst die passenden Gauss-Knoten und Gauss-Gewichte bestimmt. An diesen Punkten werden nun die B-Splines und die Gewichtsfunktion und deren Gradienten ausgewertet. Die Aufteilung der Gauss-Punkte ist auch in den Abbildungen 1.5 und 1.6 zu erkennen. Damit kann die lokale Massenmatrix A_{loc} und Steifigkeitsmatrix B_{loc} auf jeder Zelle Q bestimmt werden. Man erhält folgende Matrixeinträge:

$$(a_{i,j}) \approx \sum_{i_1=1}^{\ell_1} \sum_{i_2=1}^{\ell_2} g_{1,i_1} g_{2,i_2} \alpha_1 \alpha_2$$

$$(b_{i,j}) \approx \sum_{i_1=1}^{\ell_1} \sum_{i_2=1}^{\ell_2} g_{1,i_1} g_{2,i_2} w^2(x_{i_1}, y_{i_2})\, b_i(x_{i_1}, y_{i_2})\, b_j(x_{i_1}, y_{i_2})$$

mit

$$\alpha_1 = \operatorname{grad} w(x_{i_1}, y_{i_2})\, b_i(x_{i_1}, y_{i_2}) + w(x_{i_1}, y_{i_2}) \operatorname{grad} b_i(x_{i_1}, y_{i_2})$$
$$\alpha_2 = \operatorname{grad} w(x_{i_1}, y_{i_2})\, b_j(x_{i_1}, y_{i_2}) + w(x_{i_1}, y_{i_2}) \operatorname{grad} b_j(x_{i_1}, y_{i_2})$$

Mit diesen Werten setzt man die globale Massenmatrix A und Steifigkeitsmatrix B zusammen und legt zusätzlich am Ende die Diagonaleinträge, die Null sind, auf eins fest. Für die weitere Berechnung werden die vierdimensionalen Matrizen in zweidimensionale Matrizen umgewandelt. Danach wird die Erweiterungsmatrix bestimmt, damit man als Finite-Elemente-Basis die $w^e B_h^n$ erhält. Dabei wird in einer Schleife über alle B-Splines für jeden inneren B-Spline die Indexmenge der äußeren B-Splines bestimmt, die zu diesem B-Spline addiert werden. Für jeden äußeren B-Spline wird die Menge der inneren B-Splines bestimmt, zu

denen dieser Spline addiert wird. Die Einträge der Erweiterungsmatrix E werden mittels Auswertung geeigneter Lagrange-Funktionen und der Gewichtsfunktion, wie in Gleichung (2.6) beschrieben, berechnet. Mit den Multiplikationen

$$A^e = E \cdot A \cdot E^t\,, \qquad B^e = E \cdot B \cdot E^t$$

entstehen die kleineren Matrizen bezüglich der WEB-Spline-Basis, wobei das hochgestellte e die Erweiterung bedeutet. In den Abbildungen 6.1 und 6.2 erkennt man, wie sich die Belegungsstruktur der Matrix beim Splinegrad drei verändert. Dabei ist in diesem Beispiel A eine 1008×1008-Matrix und die erweiterte Matrix A^e eine 698×698-Matrix.

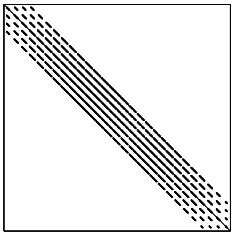


Abbildung 6.1: Matrix ohne Erweiterung

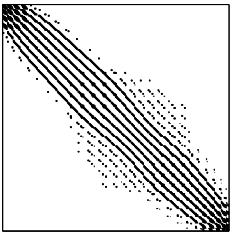


Abbildung 6.2: Matrix mit Erweiterung

Die entsprechende Erweiterungsmatrix ist in Abbildung 6.3 dargestellt. Man sieht auch die rechteckige Form der Erweiterungsmatrix, also dass der Raum der WB-Splines größer ist als der Raum der WEB-Splines.

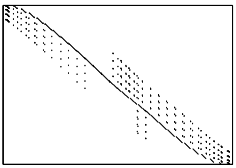


Abbildung 6.3: Struktur einer Erweiterungsmatrix

Die Erweiterungsmatrix wird ebenfalls auf jeder Gitterweite benötigt. Hat man nun die Steifigkeitsmatrix und die Massenmatrix auf dem feinsten Gitter aufgestellt, können die Projektionsmatrizen

$$P_i \quad \text{für } i = 1, \ldots, (q-1)$$

bestimmt werden. Diese werden unter anderem dafür benötigt, um die Massenmatrix und die Steifigkeitsmatrix auf den unterschiedlichen Gitterweiten darzustellen, ohne sie auf jedem Gitter mittels Integration zu bestimmen. Die Matrizen berechnen sich nur noch mit Multiplikation

$$\begin{aligned} A_i^e &= P_i \cdot A_{i+1}^e \cdot P_i^t \\ B_i^e &= P_i \cdot B_{i+1}^e \cdot P_i^t \end{aligned} \qquad \text{für } i = 1, \ldots, (q-1)\,.$$

Nun hat man für den Lanczos-Algorithmus alle notwendigen Matrizen und wendet diesen an. Als Startvektor für den Lanczos-Algorithmus wird ein zufällig gewählter normierter Vektor v_1 genommen. Dadurch wird fast ausgeschlossen, dass v_1 senkrecht auf einem gesuchten Eigenvektor steht und dieser nicht approximiert werden kann. Man wendet nun den Shift-and-Invert Lanczos-Algorithmus (vgl. Algorithmus 4.2) an und löst im ersten Schritt mittels Multigrid-Verfahren die Gleichung

$$w_i = \underbrace{(A_1^e - \sigma B_1^e)}_{C_1^e}{}^{-1} B_1^e v_i\,. \tag{6.1}$$

Zuerst wird der Startwert für w_i bestimmt. Dabei geht man von der gröbsten Gitterweite aus und kommt mittels Projektion auf immer feinere Gitter, bis man das feinste Gitter erreicht hat, wie in Algorithmus 5.4 beschrieben. Da bei der Wahl von $\sigma \neq 0$ die Matrix $C_1^e = A_1^e - \sigma B_i^e$ nicht mehr positiv definit sein muss, wird hier als Glätter die in MATLAB schon vorhandene ILU-Faktorisierung genommen, also für $C^e = \widetilde{L}\widetilde{U} - R$

$$x_{k+1} = x_k - \widetilde{U}^{-1}\widetilde{L}^{-1}\left(C^e x_k - b\right)\,.$$

Andere Glätter, wie z. B. die SSOR-Iteration, konvergieren in dem Fall, falls die Matrix nicht mehr positiv definit ist, nicht mehr. Danach wird der Lanczos-Algorithmus mit implizitem Neustart fortgesetzt (vgl. Algorithmus 4.3). Dabei wird für die Shifts die „Exact Shift Strategy" angewendet. Die Anzahl der Shifts hängt davon ab, wie gut die Eigenwerte schon approximiert wurden. Maximal nimmt man die halbe Anzahl der Lanczos-Schritte als Anzahl der Shifts und wählt die schlechtesten Ritz-Werte als Shifts. Die Fehler werden mit Formel (4.1) abgeschätzt. Da die Orthogonalität der Lanczos-Vektoren bald verloren geht, wurde hier die vollständige Orthogonalisierung gewählt. Ohne Orthogonalisierung bekommt man schon nach wenigen Schritten aufgrund der Rundungsfehler schlecht approximierte Eigenwerte.

6.2 Benutzeroberfläche

Man muss in der Benutzeroberfläche das Gebiet und die Gewichtsfunktion vorgeben. Die Randkurve des Gebietes muss als Bézierkurve gegeben sein. Dabei ist eine Bézierkurve der Ordnung m eine Kurve der Form

$$r(x) = \sum_{i=1}^{m} b_i^m(x) p_i$$

mit den Kontrollpunkten p_i und den Bernsteinpolynomen

$$b_{i+1}^{m+1}(x) = \binom{m}{i} (1-x)^{m-i} x^i \qquad \text{für } 0 \leq x \leq 1\,.$$

Die Abbildung 6.4 zeigt eine Bézierkurve der Ordnung fünf. Dabei geht die Kurve immer durch den ersten Punkt p_1 und den letzten Punkt p_m.

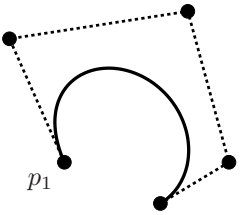


Abbildung 6.4: Bézierkurve der Ordnung fünf

Die Gewichtsfunktion muss in impliziter Form angegeben werden, wobei die Gewichtsfunktion am Rand Null und innerhalb des Gebietes größer Null sein muss. Außerdem ist die feinste Gitterweite h und der Splinegrad n der WEB-Splines $w^e B_h^n$ von Interesse. Wenn man schon abgespeicherte Matrizen hat und keine neuen berechnen will, so ist dies nicht nötig, da nur diese Matrizen geladen werden müssen. Ansonsten werden die Matrizen mit den Daten berechnet und, falls man möchte, abgespeichert. In der Benutzeroberfläche kann der Bereich $[\alpha, \beta]$, in dem die Eigenwerte approximiert werden sollen, und die maximale Anzahl max_{EW} der zu bestimmenden Eigenwerte vorgegeben werden. Dabei wird der Lanczos-Algorithmus abgebrochen, falls der Bereich überschritten wird oder die maximale Anzahl erreicht ist. Für σ wird für den ersten Lanczos-Schritt die linke vorgegebene Schranke α genommen.

Man gibt an, wieviel Lanczos-Schritte man berechnen will und bestimmt danach ein größeres σ. Damit muss man auch ein neues C_1^e aus Gleichung (6.1) berechnen. Mit dem alten σ erhält man eine bestimmte Anzahl Eigenwerte, die schon konvergiert sind. Die darauf folgenden bekommt man im neuen Lanczos-Schritt mit dem größeren σ.

Außerdem muss noch der Parameter für den rekursiven Aufruf aus Algorithmus 5.2 angegeben werden. Dadurch entscheidet sich, ob der V-Zyklus, der W-Zyklus oder ein anderer Zyklus durchlaufen wird. Die Anzahl der Glättungsschritte, die in jedem Schritt durchgeführt werden sollen, müssen auch vorgegeben werden.

Im Hauptmenü kann, wie in Abbildung 6.5, noch das Gebiet mit dem feinsten Gebiet ausgegeben werden, wobei die inneren, Rand- und äußeren Zellen farbig markiert sind.

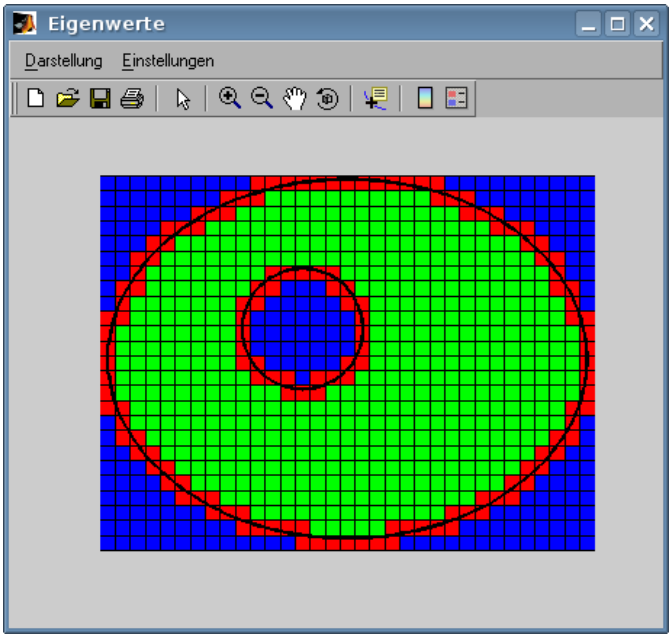


Abbildung 6.5: Darstellung des feinsten Gitters

Das Eingabefenster in MATLAB sieht man in Abbildung 6.6.

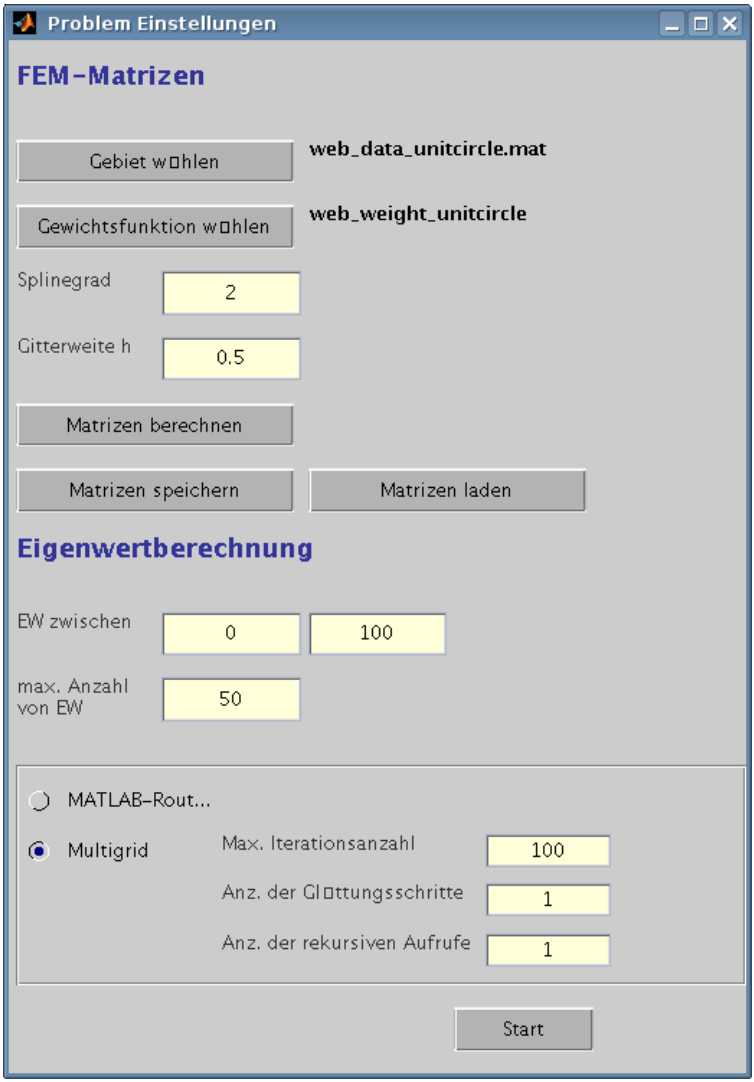


Abbildung 6.6: Darstellung der Benutzeroberfläche

Dabei ist das Gebiet ein mat-File und die Gewichtsfunktion ein m-File. Sind die Matrizen berechnet, kann gewählt werden, ob man den

MATLAB-Löser oder den Multigrid-Löser anwenden will. In MATLAB wird das Problem mit dem eigs-Befehl gelöst. Dies ist eine Subroutine des Programmpakets ARPACK und verwendet ebenfalls den Lanczos-Algorithmus mit implizitem Neustart, allerdings ohne Multigrid-Löser.

6.3 Beispielrechnungen

Hierbei wird als erstes wieder der Einheitskreis betrachtet, da der Vorteil darin besteht, dass man die richtigen Eigenwerte kennt. In diesem Fall geht es darum, möglichst viele Eigenwerte bei großen Matrizen zu bestimmen. Die Eigenwerte werden zwischen Null und 5000 bei einer Gitterweite 2^{-7} mit Splinegrad drei berechnet. Diese sind in Abbildung 6.7 dargestellt. Insgesamt sind es 1214 Eigenwerte. In diesem Beispiel hat man überwiegend doppelte Eigenwerte, da nur die Nullstellen der Bessel-Funktion nullter Ordnung einfach vorkommen. Alle anderen Eigenwerte kommen doppelt vor, da die Nullstellen der Bessel-Funktionen J_k mit denen für J_{-k}, für $k = 1, 2, \ldots$, übereinstimmen. Die Nullstellen im Quadrat sind die Eigenwerte in diesem Beispiel.

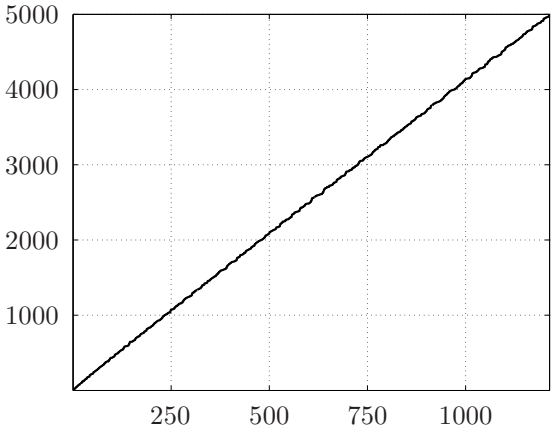


Abbildung 6.7: Eigenwerte am Einheitskreis

Dieses Beispiel zeigt, dass die Eigenwerte fast gleichverteilt sind. Daher werden hier die Randeigenwerte, also die mit dem größten Betrag gut approximiert. Beim Shift-and-Invert Lanczos-Algorithmus sind es die, welche nahe bei σ liegen.

Als zweites Gebiet nimmt man wieder das Gebiet aus Abbildung 3.3. Hier wurden die Eigenwerte zwischen Null und 2500 bestimmt. Die Gitterweite betrug 2^{-6} beim Ansatzgrad zwei. Abbildung 6.8 zeigt für dieses Gebiet die berechneten Eigenwerte. In diesem Beispiel sind die genauen Eigenwerte nicht bekannt. Man kann also nur die Eigenwerte auf verschiedenen Gitterweiten vergleichen, um Approximationsordnungen zu bestimmen, vgl. Unterabschnitt 3.2.2.

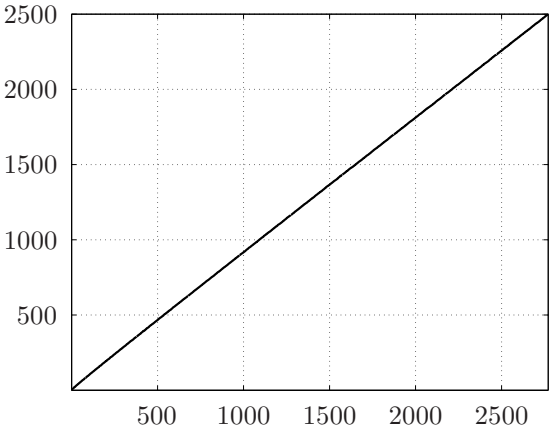


Abbildung 6.8: Eigenwerte bei glattem Rand

Auch in diesem Beispiel erkennt man, dass die Eigenwerte gleichverteilt sind, also immer die Eigenwerte gut approximiert werden, die nahe bei dem σ liegen. Im vorgegebenen Bereich lagen 2771 Eigenwerte, wobei alle Eigenwerte die Vielfachheit eins hatten.

Als letztes Beispiel wird das Quadrat mit Kantenlänge π betrachtet. Hier sind die Eigenwerte bekannt. Allerdings hat dieses Gebiet keine glatten Ränder, daher kann in diesem Fall keine Aussage über die Approximationsordnung gemacht werden. Man nimmt Splinegrad vier bei einer Gitterweite 2^{-6}. Das Gebiet hat zwar Ecken, aber die Lösung (6.2) hat in den Ecken keine Singularitäten. Als Gewichtsfunktion wurde

$$w(x, y) = x\,(\pi - x)\,y\,(\pi - y)$$

genommen, deren erste Ableitungen glatt sind. Hier kommen fast nur mehrfache Eigenwerte vor. Die Eigenwerte in diesem Quadrat sind a^2+b^2

mit beliebigen, natürlichen Zahlen a und b. Für die Lösung

$$u(x, y) = \sin(ax) \cdot \sin(bx) \tag{6.2}$$

gilt

$$-\Delta u = \left(a^2 + b^2\right) u = \lambda u \,.$$

Die Eigenwerte zwischen Null und 2000 sind in Abbildung 6.9 dargestellt. Auch hier ist wieder eine Gleichverteilung der Eigenwerte zu erkennen. In dem vorgegebenen Frequenzbereich wurden 1529 Eigenwerte gefunden, wobei die Vielfachheit der Eigenwerte bis zu acht (z. B. $1105 = 4^2+33^2 = 9^2 + 32^2 = 12^2 + 31^2 = 23^2 + 24^2$) anstieg.

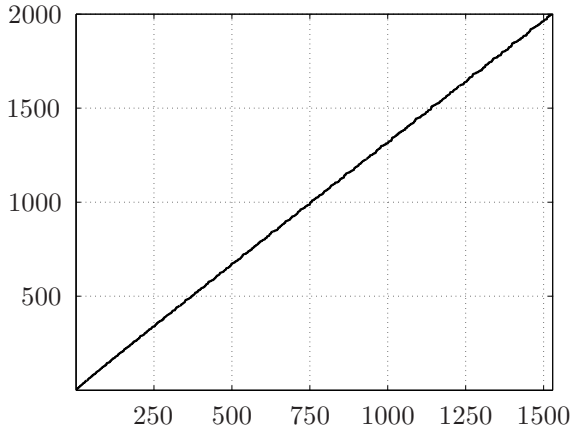


Abbildung 6.9: Eigenwerte auf dem Quadrat mit Kantenlänge π

In allen Beispielen wurde das Spektrum der zu berechnenden Eigenwerte in mehreren Intervallen einzeln betrachtet. Das zu untersuchende Spektrum $[\lambda_{\min}, \lambda_{\max}]$ wurde in n gleich große Intervalle $[\lambda_{\min}, \lambda_1]$, $[\lambda_1, \lambda_2]$, …, $[\lambda_{n-1}, \lambda_{\max}]$ aufgeteilt, und in diesen Abschnitten wurden die Eigenwerte berechnet. Diese wurden anschließend zusammengefasst. Damit ist die Parallelisierbarkeit des Verfahrens schon gegeben, da alle Teilintervalle auf verschiedenen Rechnern unabhängig voneinander untersucht werden können. Allerdings sollten in jedem Teilbereich etwa gleich viele Eigenwerte liegen, da sonst die Parallelisierung nicht sinnvoll ist. Die Aufteilung in unterschiedliche Frequenzbereiche ist dann sinnvoll, wenn die Matrizen und der zu untersuchende Frequenzbereich groß sind.

In den verschiedenen Gebieten wurden die Ansatzgrade zwei, drei und vier untersucht. Dass die ersten zwei Beispiele die richtige Approximationsordnung haben, wurde in Abschnitt 3.2 dargestellt. Darauf wurde in diesem Abschnitt kein Wert gelegt. Der Ansatzgrad eins wird im folgenden Kapitel noch untersucht. Dabei geht es um ein Eigenwertproblem mit Neumann-Nullrandbedingungen, bei dem es auf die Schnelligkeit der Berechnung ankommt.

Kapitel 7

Wasserwellen

Als ein Anwendungsbeispiel wurde die Wellenbewegung auf der Wasseroberfläche auf beschränkten zusammenhängenden Gebieten mit senkrechtem Rand untersucht. Diese Gebiete findet man z. B. bei Hafenbecken oder in Schwimmbädern. Dabei repräsentieren die berechneten Eigenwerte die Eigenfrequenzen und die Eigenvektoren die Wellenbewegung an der Oberfläche. Als Basisfunktionen wurden nicht die WEB-Splines aus Abschnitt 2.4, sondern Box-Splines genommen, da diese bei der Integration einfacher sind. Hierbei sollte auch nicht die Genauigkeit der wesentliche Faktor sein, da physikalische Modelle nicht exakt die Wirklichkeit wiederspiegeln, sondern es wurde der Zeitfaktor minimiert.

7.1 Einführung

Da in diesem Beispiel die Laufzeit der wichtigste Faktor war, wurde der Ansatzgrad der Splines fest auf eins gesetzt. Dadurch konnte die Integration stark vereinfacht werden.
In bestimmten Fällen können nach [ZT00] Oberflächenwellen mit der Gleichung

$$\nabla^{\mathrm{t}} \left(H \nabla u \right) + \frac{\omega^2}{g} u = 0 \tag{7.1}$$

mit der Gewichtskraft g dargestellt werden. Dabei gibt u die Auslenkung der Welle von der Wasseroberfläche an und sollte, damit die Genauigkeit der Lösung noch gewährleistet ist, wesentlich kleiner sein als die Wassertiefe H an einer beliebigen Stelle. Für konstantes $H = h$ erhält man

die Helmholtz-Gleichung

$$\nabla^{\mathrm{t}} \left(\nabla u\right) + k^2 u = 0 \tag{7.2}$$

mit den Eigenwerten $k = \frac{\omega}{\sqrt{gh}}$. Mit der Gleichung (7.1) werden nun die Eigenwerte auf verschiedenen Gebieten bestimmt. Dabei multipliziert man die Gleichung mit einer Testfunktion v und integriert auf beiden Seiten über das Gebiet Ω

$$\int\limits_{\Omega} v \left(\nabla^{\mathrm{t}} \left(H \nabla u\right)\right) + \int\limits_{\Omega} v \left(\frac{\omega^2}{g} u\right) = 0\,.$$

Auf das erste Integral wendet man nun den ersten Integralsatz nach Green an

$$\int\limits_{\Omega} \left(\nabla v\right)^{\mathrm{t}} \left(H \nabla u\right) + \int\limits_{\partial\Omega} v H \frac{\partial u}{\partial n} + \int\limits_{\Omega} v \left(\frac{\omega^2}{g} u\right) = 0 \tag{7.3}$$

mit $\frac{\partial u}{\partial n}$ der Normalenableitung am Rand $\partial\Omega$. Für einen vertikalen, ohne Störung reflektierenden Rand gilt $\frac{\partial u}{\partial n} = 0$ und damit ist in Gleichung (7.3) das zweite Integral Null und muss nicht berechnet werden. Dieser Fall wurde implementiert, und es konnten auf unterschiedlichen Gebieten die Eigenwerte bestimmt werden. Da in diesem Fall Neumann-Randbedingungen vorliegen, kann man den Ansatzgrad der Basis noch verringern, da die Gewichtsfunktion weggelassen werden kann. Die Gewichtsfunktion wird noch für die lineare Approximation des Gebietsrandes benötigt. Die lineare Approximation wird mit den vier Werten an den Ecken jeder Gitterzelle bestimmt. Man benötigt also nur die Gewichtsfunktion an den Gitterpunkten. Mit dem linearisierten Rand auf jeder Gitterzelle gibt es folgende Schnittfälle:

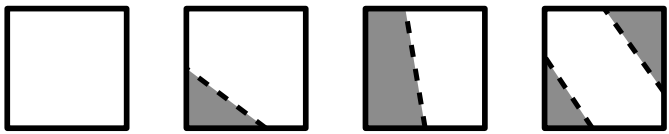


Abbildung 7.1: Mögliche Schnittfälle im zweidimensionalen Fall

Alle anderen Schnittfälle gehen durch Drehung in die in Abbildung 7.1 dargestellten Schnittfälle über. Dabei spiegelt die grau eingefärbete Fläche entweder den inneren oder den äußeren Teil wieder. Der letzte Fall

aus Abbildung 7.1 sollte möglichst nicht auftreten, ansonsten muss das Gitter feiner gewählt werden. Da der Ansatzgrad der Basisfunktionen linear ist, wird die Funktion H, die die Wassertiefe beschreibt, ebenfalls auf jedem Teildreieck linear approximiert

$$\widetilde{H} = a + bx + cy \quad \text{mit} \quad a\,,\ b\,,\ c\ \in \mathbb{R}\,.$$

Die Basisfunktionen können mit wenigen Integrationspunkten exakt integriert werden. Für die Massenmatrix gilt

$$a_{i,j} = \int_\Omega \underbrace{\nabla b_i}_{\text{const}}\ \widetilde{H}\ \underbrace{\nabla b_j}_{\text{const}} = \nabla b_i \nabla b_j \int_\Omega \widetilde{H}$$

und die Steifigkeitsmatrix ist

$$b_{i,j} = \int_\Omega b_i b_j\,.$$

Jeder Schnittfall kann als Dreieck oder als Differenz von zwei Dreiecken dargestellt werden.

7.2 Basisfunktionen und Löser

Die Basisfunktionen sind Box-Splines, vgl. [dBHR93],

$$b(x) = B_\Xi\,, \quad \Xi = \begin{pmatrix} 1 & 0 & -1 \\ 0 & 1 & 1 \end{pmatrix}.$$

Man erhält den Box-Spline, indem man die charakteristische Funktion χ über das Einheitsquadrat $[0,1]^2$ in Richtung

$$\xi = \begin{pmatrix} -1 \\ 1 \end{pmatrix}$$

mittelt

$$b(x) = \int_0^1 \chi\left(x - t\xi\right)\, dt \qquad x \in \mathbb{R}^2\,.$$

b ist eine stückweise lineare positive Funktion mit Träger $[0,1]^2 + [0,1]\xi$, wie in Abbildung 7.2 und Form aus Abbildung 7.3.

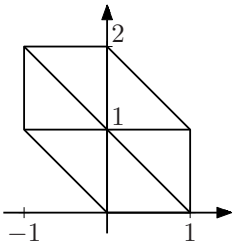


Abbildung 7.2: Träger eines Box-Splines

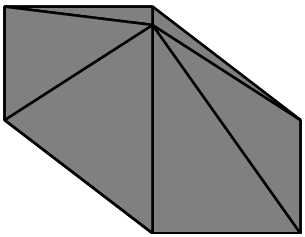


Abbildung 7.3: Form eines Box-Splines

Dieser Basisansatz ist hier sinnvoll, da auf jedem Dreieck der Box-Spline eine lineare Funktion ist, und die Integration sich daher wesentlich vereinfacht. Man verringert dadurch die Konvergenzordnung nicht, da die Gewichtsfunktion linear approximiert wird und dies die Konvergenzrate schon vorgibt.
Dieser Box-Spline wird entsprechend seiner Position verschoben und gemäß der Gitterweite skaliert

$$b_k^h = b\left(\cdot/(h-k)\right)\,, \quad h>0\,, \quad k=(k_1,k_2)\in\mathbb{Z}^2\,.$$

Das Gitter wird durch die Spaltenvektoren von Ξ aufgespannt. Dabei wird jede Gitterzelle in zwei Dreiecke aufgeteilt. Das Gitter ist zwar komplizierter als bei den Tensorprodukt-B-Splines, allerdings überwiegt der Vorteil, dass der Grad des Polynomraums nur linear und nicht quadratisch ist.
Mit den Box-Splines als Basisfunktionen kann man die Massenmatrix A und die Steifigkeitsmatrix B berechnen und man erhält das verallgemeinerte Eigenwertproblem

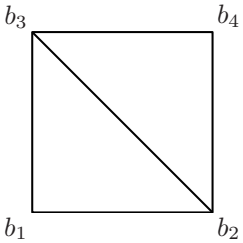
$$Au = \lambda Bu\,, \tag{7.4}$$

dessen Eigenwerte werden im Folgenden mit dem Lanczos-Algorithmus berechnet. Die Anzahl der Nicht-Null-Einträge in einer Zeile in den Matrizen A und B verringert sich (sieben bei den linearen Box-Splines gegenüber neun bei WEB-Splines mit Tensorprodukt-B-Splines mit Ansatzgrad eins). Als Glätter wurde der SSOR-Algorithmus genommen. Da in diesem Fall Neumann-Randbedingungen vorliegen, hat die Gleichung (7.4) einen Nulleigenwert mit zugehörigem Eigenvektor $(1,\,1,\,\ldots,\,1)^{\mathrm{t}}$, da dieser die Differentialgleichung und die Randbedingung erfüllt und zu jeder Lösung addiert werden kann. Es entspricht einer Verschiebung nach oben bzw. unten. Damit die Konvergenz des SSOR-Algorithmus gesichert ist, müssen alle Eigenwerte größer Null sein. Der Null-Eigenwert

ist nicht von Interesse, daher werden alle restlichen Eigenvektoren im Orthogonalraum zu $(1, 1, \ldots, 1)^{\mathrm{t}}$ gesucht. Die restlichen Eigenwerte sind größer Null, also ist die Konvergenz des Glätters gesichert. Die Lanczos-Vektoren werden orthogonal zum Vektor $v_1 = (1, 1, \ldots, 1)^{\mathrm{t}}$ bestimmt, und somit ist dazu der Krylov-Unterraum orthogonal. Da nur aus diesem Unterraum die Eigenvektoren approximiert werden, ist gesichert, dass alle Vektoren senkrecht zu v_1 sind und dass beim SSOR-Algorithmus alle Vektoren keinen Anteil von v_1 haben.

7.3 Integration

Die Integrale werden im Voraus als Polynome in Abhängigkeit von den Funktionswerten der Wassertiefe $\widetilde{H}$ und der Gewichtsfunktion w, die das Gebiet beschreibt, an den drei Eckpunkten jedes Dreiecks berechnet und gespeichert. Daher gibt es zehn mögliche Splinekombinationen



Splinekombinationen:
b_1b_1 , b_1b_2 , b_1b_3 , b_1b_4 , b_2b_2
b_2b_3 , b_2b_4 , b_3b_3 , b_3b_4 , b_4b_4

b_i, $i = 1 \ldots, 4$ gibt die Ecke an, an der der Spline eins ist

und 2^4 mögliche Schnittfälle, da jede Ecke entweder innerhalb oder außerhalb des Gebietes liegt. Es müssen also $2^4 \cdot 10$ Polynome für die Massenmatrix und die Steifigkeitsmatrix bestimmt werden. Für die Massenmatrix muss man nur die Wassertiefe $\widetilde{H}$ über das entsprechende Dreieck integrieren. Beispielsweise ergibt sich für das Dreieck aus Abbildung 7.4 folgende Integrationsformel

$$a_{i,j} = \underbrace{\nabla {b_i}^{\mathrm{t}} \nabla b_j}_{\text{const}} \int_{\triangle} \widetilde{H} = \frac{1}{6} s_1 s_2 \left(h_1 + h_2 + h_3\right) .$$

s_1 und s_2 erhält man aus der Gewichtsfunktion an den Eckpunkten und h_1, h_2 und h_3 mit s_1, s_2 und den Wassertiefen an den Eckpunkten (vgl. Beispiel 1). Man benötigt für die Integration nur die Wassertiefen und Gewichte an den Eckpunkten des Gitters.

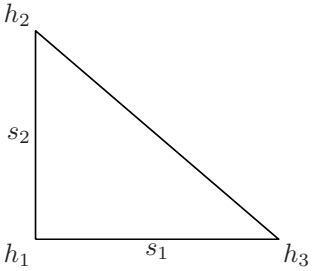


Abbildung 7.4: Konstanten für die Integration

Die Integration für die Steifigkeitsmatrix wurde mit einer anderen Integrationsformel über Dreiecke berechnet.

$$\begin{aligned} b_{i,j} &= \int_{\triangle} b_i b_j \\ &= \frac{1}{24} s_1 s_2 h_x h_y \left(\sum_{k=1}^{3} \left(b_i\left(P_k\right) b_j\left(P_k\right) \right) + \sum_{k=1}^{3} b_i\left(P_k\right) \cdot \sum_{k=1}^{3} b_j\left(P_k\right) \right) \end{aligned}$$

$b_i\left(P_k\right)$ gibt dabei den Funktionswert von b_i am Punkt P_k an. P_1, P_2 und P_3 sind die drei Eckpunkte des Dreiecks und h_x die Gitterweite in x-Richtung bzw. h_y die Gitterweite in y-Richtung. Da $b_i\left(P_k\right),\ k = 1, 2, 3$ nur von den Gewichten an den Gitterzellen abhängt, kann es als Polynom von diesen dargestellt werden (vgl. Beispiel 1).
Damit können also beide Integrale mit Funktionswerten an den Eckpunkten der Gitterzellen dargestellt werden und in Abhängigkeit von diesen im Voraus abgespeichert werden.

Beispiel 1:
Für den möglichen Schnittfall, dass die untere rechte Ecke abgeschnitten wird, sehen bei der Splinekombination b_1, der Spline, der links unten eins ist, und b_2, der Spline, der rechts unten eins ist, die vorab gespeicherten Integrale beispielsweise folgendermaßen aus

$$\int \nabla b_1 \widetilde{H} \nabla b_2 = -\frac{1}{6} s_1 s_2 \left(\widetilde{H}\left(P_5\right) + \widetilde{H}\left(P_2\right) + \widetilde{H}\left(P_6\right) \right)$$

$$\int b_1 b_2 = \frac{1}{24} h_x h_y s_1 s_2 \left(4 s_1 - 2 s_1^2 - s_1 s_2 \right)$$

mit den Eckpunkten $P_i\,,\ i = 1, \cdots, 4$ und

$$s_1 = \frac{w(P_2)}{w(P_2) - w(P_1)}\,, \quad s_2 = \frac{w(P_2)}{w(P_2) - w(P_3)}$$
$$\widetilde{H}(P_5) = s_1 \left(\widetilde{H}(P_1) - \widetilde{H}(P_2)\right) + \widetilde{H}(P_2)$$
$$\widetilde{H}(P_6) = s_2 \left(\widetilde{H}(P_3) - \widetilde{H}(P_2)\right) + \widetilde{H}(P_2)$$
$$b_1(P_2) = 0\,, \qquad b_1(P_5) = s_1\,, \qquad b_1(P_6) = 0$$
$$b_2(P_2) = 1\,, \qquad b_2(P_5) = 1 - s_1\,, \qquad b_2(P_6) = 1 - s_2\,.$$

Dabei musste nur über die grau gefärbte Fläche in Abbildung 7.5 integriert werden, da b_1 im anderen Teilbereich Null ist.

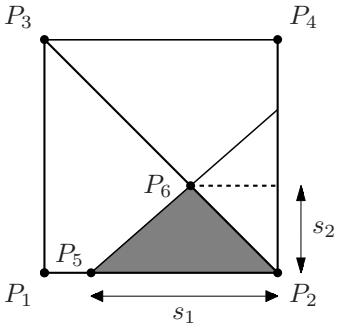


Abbildung 7.5: Beispiel für Integration

7.4 Beispiel

In der Praxis kennt man bei konstanter Höhe h beim Kreis die Eigenwerte k^2 auf der Einheitskreisscheibe (vgl. [Lam07]). Diese sind bei Höhe eins die Extremstellen der Bessel-Funktionen. Die ersten zehn Eigenwerte sind also (vgl. Abbildung 7.6).

3.39, 3.39, 9.33, 9.33, 14.68, 17.65, 17.65, 28.28, 28.28, 28.42

Allerdings ist dies kein gutes Beispiel um zu zeigen, dass die Eigenwerte die richtige Konvergenzrate besitzen, da hier die Wassertiefe konstant ist und die lineare Approximation $\widetilde{H} = H$ exakt ist. Die richtige

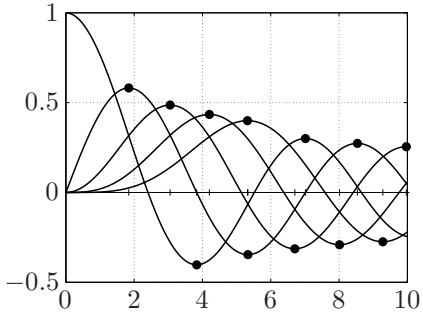


Abbildung 7.6: Extremstellen der Bessel-Funktionen für n=0, 1, 2, 3, 4

Konvergenzrate zwei erkennt man in den Abbildungen 7.7 und 7.8. Die durchgezogene Linie gibt immer die feinere Gitterweite an. Über die x-Achse wurden die Eigenwerte abgetragen. Es wurden immer die ersten 40 Eigenwerte betrachtet, wobei $e_h = \|\lambda_h - \lambda_{exakt}\|$ ist. λ_h ist der approximierte Eigenwert bei Gitterweite h und λ_{exakt} der genaue Eigenwert.

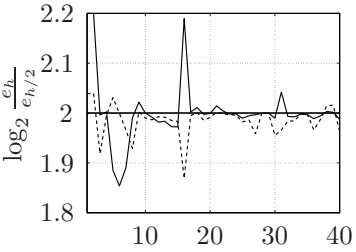


Abbildung 7.7: Gitterweite $h = \frac{1}{40}$ und $h = \frac{1}{80}$

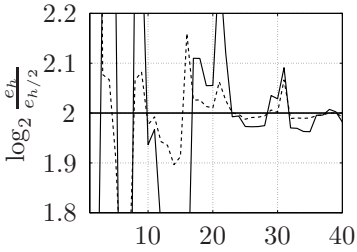


Abbildung 7.8: Gitterweite $h = \frac{1}{160}$ und $h = \frac{1}{320}$

Die Programme wurden in Fortran90 implementiert und sie liefen auf einem Pentium 4 PC, 3,2 GHz und 2 GB memory. In Tabelle 7.1 sind die Laufzeiten wiedergegeben. Die 40 kleinsten Eigenwerte wurden mit 100 Lanczos-Vektoren approximiert.

Gitterweite	$\frac{1}{40}$	$\frac{1}{80}$	$\frac{1}{160}$	$\frac{1}{320}$	$\frac{1}{640}$
Anzahl Unbekannte	6561	25921	103041	410881	1640861
innere Zellen	4856	19752	79732	320368	1284160
Randzellen	316	636	1276	2556	5116
äußere Zellen	1552	5856	22676	89240	354248
Laufzeit (CPU sec.)	6,67	26,45	109,35	461,57	1833,44

Tabelle 7.1: Laufzeiten bei verschiedenen Gitterweiten

Im zweiten Beispiel wurde das Gebiet aus Abbildung 3.3 gewählt und die Wassertiefe wurde mit

$$H(x, y) = 2 + \text{abs}\left(1 - x^2 - y^2\right)$$

beschrieben. Die durchgezogene Linie gibt in den Abbildungen 7.9 und 7.10 wieder die feinere Gitterweite an. Im linken Bild wurden die ersten neun Eigenwerte und im rechten Bild die ersten 45 Eigenwerte untersucht. In diesem Beispiel berechnete sich $e_h = \|\lambda_h - \lambda_{h/2}\|$, da man die genauen Eigenwerte nicht kennt.

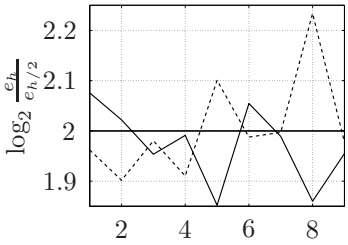


Abbildung 7.9: Gitterweite $h = \frac{1}{8}$ und $h = \frac{1}{16}$

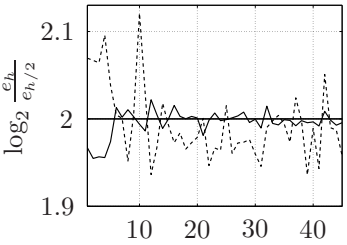


Abbildung 7.10: Gitterweite $h = \frac{1}{32}$ und $h = \frac{1}{64}$

Auch in diesem Beispiel erkennt man die richtige Konvergenzrate zwei der Eigenwerte. Die Wassertiefe H wurde mit $\widetilde{H}$ linear approximiert und gibt diese nicht exakt wieder. In der Abbildung 7.11 sind die Höhenlinien

der ersten vier Eigenschwingungsformen zu sehen. Man erkennt deutlich, dass auf dem Rand $\partial\Omega$ die Normalenableitung $\frac{\partial u}{\partial n}$ Null ist, da die Höhenlinien senkrecht auf den Rand zulaufen.

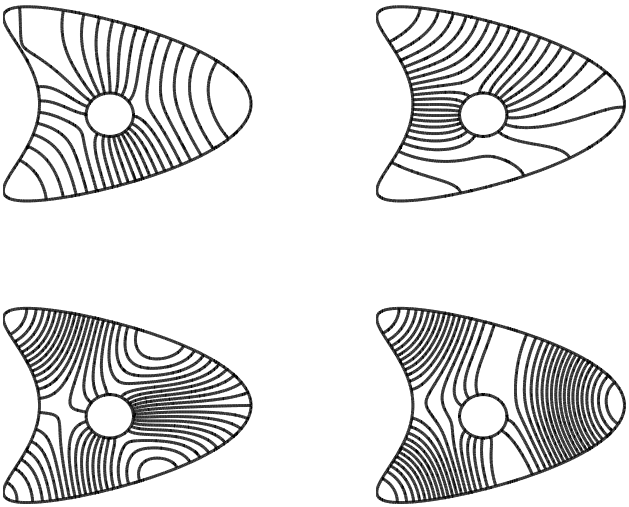


Abbildung 7.11: Eigenschwingungsformen zu den ersten vier Eigenwerten

7.5 Parallelisierung

Die Berechnung der Integrale für die Massenmatrix und Steifigkeitsmatrix ist parallelisierbar, da alle Integrale unabhängig voneinander berechenbar sind. Beim Löser ist allerdings der SSOR-Glätter nicht parallelisierbar, da bei einem Glättungsschritt auf Werte zurückgegriffen wird, die im selben Schritt berechnet wurden. Daher muss man entweder auf andere Glätter zurückgreifen oder den SSOR-Glätter modifizieren. Andere parallelisierbare Glätter sind die Richardson- und Jacobi-Iteration, die keine so guten Konvergenzraten haben und deren Konvergenzbereiche kleiner sind. Eine Modifizierung des SSOR-Glätters ist zum Beispiel der Farben-SSOR-Algorithmus. In einem Glättungsschritt wird das Gitter mehrmals durchlaufen und es werden alle Werte in einem Schritt berechnet, die nicht voneinander abhängen. Man unterteilt das Gebiet

in mindestens zwei disjunkte Teilmengen, und innerhalb der Teilmengen soll keine Datenabhängigkeit existieren. Im Fall der linearen Box-Splines benötigt man mindestens drei Teilmengen. Diese Teilmengen haben auf dem Gitter die in Abbildung 7.12 dargestellte Aufteilung. Es sind auch andere Aufteilungen oder mehr Teilmengen möglich, allerdings ist hier drei die minimale Anzahl.

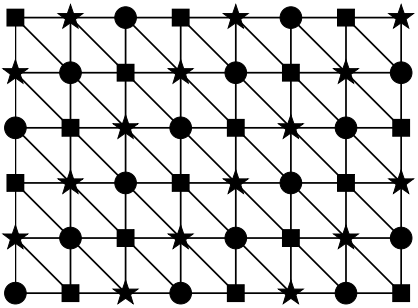


Abbildung 7.12: Mögliche Aufteilung von drei Farben im SSOR-Glätter

$x^\bullet$ sind die x-Werte mit den entsprechenden Indizes, also an den Stellen mit ●. Bei einer Matrix $A \in \mathbb{R}^{n\times n}$ oder einem Vektor $b \in \mathbb{R}^n$ sollen bei $A^\bullet$ und bei $b^\bullet$ nur die Zeilen der Matrix A und des Vektors b übrig bleiben, die an den Positionen ● stehen.

Damit erhält man den Algorithmus 7.1 des Farben-SSOR-Glätters.

Algorithmus 7.1 ein SSOR-Schritt des Farben-SSOR-Glätters

for i ∈ {● ■ ★} **do**

$$x^i_{k+1} = x^i_k - \omega \left(D^i\right)^{-1} \left(Lx_{k+1} + Dx_k + rx_k\right) + \omega \left(D^i\right)^{-1} b$$

end for

Damit hat man einen SSOR-Glätter erhalten, der parallelisierbar ist und eine bessere Konvergenzrate als der Richardson-Glätter oder der Jacobi-Glätter hat. Die Konvergenzrate des ursprünglichen SSOR-Glätters ist zwar besser, dies wird allerdings durch die Parallelisierung wieder ausgeglichen.

7.6 Ostsee

Als weiteres Anwendungsbeispiel für Wasserwellen wurde ein Teil der Ostsee genommen. Dieses spiegelt die Wirklichkeit nicht ganz wider ist aber ein interessantes, anschauliches und anwendungsorientiertes Gebiet. Es wurden mit Ansatzgrad eins, die Matrizen berechnet und mit Multigrid-Löser gelöst.

Abbildung 7.13: untersuchtes Gebiet der Ostsee

Das feinste Gitter ist dabei 769×641. Man hatte auf dem feinsten Gitter 492929 Basisfunktionen. Es wurden 7 Subdivisionsschritte verwendet, somit ist das gröbste Gitter 13×11 mit insgesamt 143 Basisfunktionen. Eine mögliche Darstellung des Fehlers sieht man in Abbildung 7.14.

Dabei nahm man immer an, dass der Eigenwert bei halber Gitterweite der genaue Eigenwert ist und berechnete damit den relativen Fehler. Da der Fehler über einer logarithmischen Skala aufgetragen wurde, ist der Betrag an der Stelle am größten, an welcher der Fehler am kleinsten ist. Dabei sieht man, dass der Fehler beim kleinsten Eigenwert bei der kleinsten Gitterweite am geringsten ist. Der Fehler steigt bei zunehmender Gitterweite und bei zunehmendem Eigenwert an. Es wurde auch nur die Lösung auf den vier feinsten Gitterweiten bestimmt. Bei zu großen Gitterweiten erhielt man keine sinnvollen Ergebnisse. Für die Gitterwei-

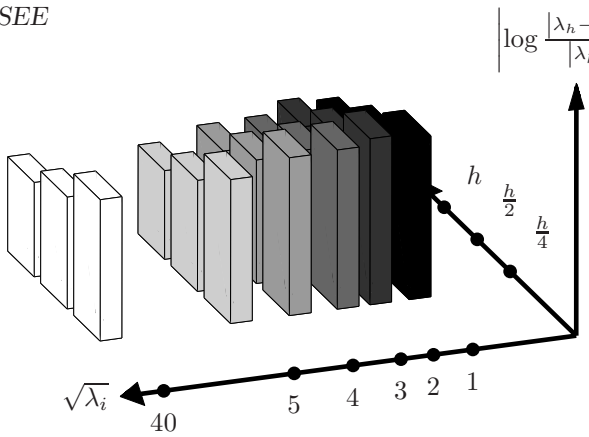


Abbildung 7.14: Fehlerplot der Ostsee

ten, die zu grob waren, war die Randapproximation zu schlecht und es traten Schnittfälle auf (vgl. letztes Bild in Abbildung 7.1), die verboten waren, da sie nicht eindeutig zugeordnet werden konnten. In den Abbildungen 7.15 bis 7.18 sind die berechneten Höhenlinien von verschiedenen Eigenfunktionen in der Ostsee eingezeichnet. Für die Wassertiefe wurde konstant $H(x, y) = 1$ genommen.

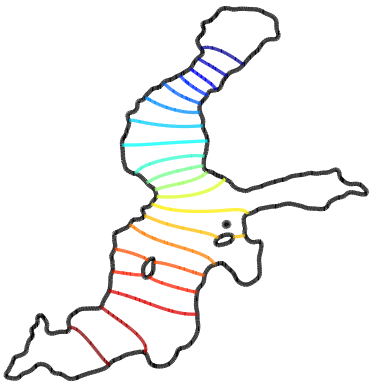


Abbildung 7.15: erste Eigenfunktion

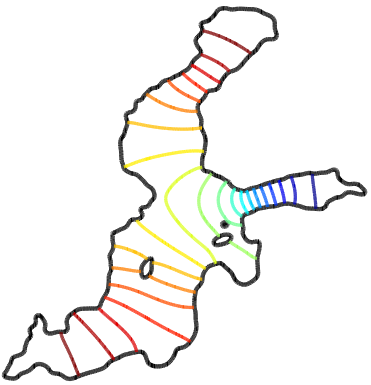


Abbildung 7.16: zweite Eigenfunktion

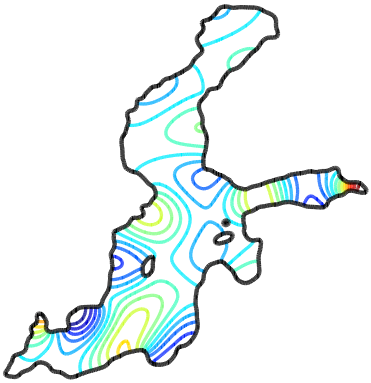


Abbildung 7.17: 20. Eigenfunktion

Abbildung 7.18: 30. Eigenfunktion

Zusammenfassung und Ausblick

Die Finite-Elemente-Methode ist ein wichtiges Hilfsmittel zur numerischen Simulation. Außerdem wird dieses Verfahren häufig zur Lösung partieller Differentialgleichungen angewendet, wie sie z. B. in der Strukturmechanik, Thermodynamik oder bei elektromagnetischen Feldern auftreten. Hier wird im Speziellen das Eigenwertproblem $-\Delta u = \lambda u$ mit Dirichlet-Nullrandbedingungen auf einem beschränkten, zusammenhängenden Gebiet Ω untersucht. Das Eigenwertproblem tritt bei Schwingungsproblemen auf. Als Beispiel hierfür kann die Schwingung großer Gebäude, Brücken, Schornsteine, usw. genannt werden.

Dabei wird nicht mit gewöhnlichen Finite-Elementen gerechnet, sondern mit WEB-Splines. WEB-Splines basieren auf B-Splines. Dies sind stückweise polynomiale Funktionen auf regulären Gittern.

Durch Multiplikation mit einer Gewichtsfunktion werden Dirichlet-Randbedingungen erzwungen. Damit erhält man die WB-Basis aus gewichteten B-Splines. Als Gewichtsfunktion können z. B. die geglättete Abstandsfunktion oder die R-Funktionen nach Rvachev genommen werden.

Bei den gewichteten B-Splines treten noch Stabilitätsprobleme auf. Um dies zu vermeiden, werden die B-Splines mit kleinem Träger im Gebiet Ω an B-Splines mit größerem Träger, die in der Nähe liegen, gekoppelt. Durch die Erweiterung der WB-Basis erhält man die WEB-Basis.

Im Folgenden werden einige Vorteile der WEB-Splines gegenüber den gewöhnlichen Finite-Elemente-Methoden erwähnt.

Der Ansatzgrad der WEB-Splines kann frei gewählt werden. Jeder Gitterpunkt entspricht einer Ansatzfunktion, unabhängig vom Ansatzgrad. Daher werden auch bei höheren Graden nur wenige Ansatzfunktionen benötigt, um eine große Genauigkeit zu erreichen. Die Approximations-

ordnung beim Eigenwertproblem ist dieselbe wie bei den gewöhnlichen Finite-Elementen.
Die reguläre und einfache Gitterstruktur ist ebenfalls ideal für hierarchische Verfeinerungen und den Multigrid-Löser.
Bei gewöhnlichen Finite-Elementen ist die Gittergenerierung zeitaufwendig. Um die Approximationsordnung zu erreichen, benötigt man eine gute Randapproximation, z. B. mit isoparametrischen Elementen. Dieser zeitaufwendige Schritt fällt bei der WEB-Basis weg und ist ein weiterer Vorteil dieser Methode.
Man verbindet bei den WEB-Splines also die Vorteile von B-Splines und gewöhnlichen Finite-Elementen.
Die Aufstellung der Massen- und Steifigkeitsmatrix in der Variationsformulierung erfolgt mit der Gauss-Quadratur-Formel. Nach der Assemblierung der beiden Matrizen wird das Eigenwertproblem mit dem Lanczos-Algorithmus gelöst. Dabei wird der Lanczos-Algorithmus verbessert, indem man implizite Neustarts einführt und die Spektraltransformation anwendet. Im Algorithmus muss ein lineares Gleichungssystem gelöst werden. Wie bereits erwähnt, ist die vorhandene Gitterstruktur ideal für den Multigrid-Löser. Dabei können verschiedene Glätter eingesetzt werden, oder der Stufenwechsel kann variieren.
Allerdings basieren die Beweise der Approximationsordnung auf einer Gewichtsfunktion, die genügend glatt ist. Daher wurden Gebiete mit Ecken noch nicht ausreichend untersucht. Dies ist ein weiteres großes Themengebiet, welches noch weiter erforscht werden muss. Man möchte bei der WEB-Basis bei Gebieten mit Singularitäten mindestens dieselbe Approximationsgüte erhalten wie bei den gewöhnlichen Finite-Elementen. Mögliche Ansätze sind zusätzliche Basisfunktionen oder hierarchische Gitterverfeinerungen an den Ecken, dies bietet ebenfalls die reguläre Gitterstruktur an.
Ein weiterer Schritt sind die Anwendungen im Dreidimensionalen, wobei hier erste Berechnungen mit der Poisson-Gleichung auch auf Vektorrechnern gemacht wurden. Dafür stehen ebenfalls nur Dirichlet-Nullrandbedingungen zur Verfügung. Kompliziertere Differentialgleichungen, wie sie z. B. in der Struktur- oder Aerodynamik auftreten, sind ein weiteres interessantes Forschungsgebiet.

Summary

In the automobile, aerospace, and civil engineering industries free undamped vibration of structures is a very important topic. The vibration of buildings, flutter of airplane wings, vibration of membranes, etc leads to an eigenvalue problem with boundary conditions. In this thesis the eigenvalue problem

$$\begin{aligned} -\Delta u &= \lambda u && \text{in } \Omega \\ u &= 0 && \text{on } \partial\Omega \end{aligned} \tag{7.5}$$

is solved with WEB-splines. Ω should be a bounded domain in $\mathbb{R}^n$. The basis of the finite elements is introduced in chapter 1. In section 1.1 the difference between an elliptical, parabolic or hyperbolic partial differential equation of second order and the different boundary conditions are explained. In the next section 1.2 we derive the variational formulation of the eigenvalue problem in equation (7.5). Here, we obtain by multiplication with a test function v in the Sobolev space H_0^1 and integration over Ω the variational formulation:

$$\text{find } u \in H_0^1 \text{ such that:} \quad \int_\Omega \nabla u \, \nabla v = \lambda \int_\Omega u \, v \qquad \forall v \in H_0^1 \tag{7.6}$$

This is an infinite dimensional linear problem, so we choose a finite dimensional subspace $w^e B_h^n = \{B_1, \ldots, B_n\}$ with $w^e B_h^n \subset H_0^1$. The basis functions are introduced in section 1.3. Different basis functions are shown in figure 1.7. Now we have a finite dimensional variational formulation:

$$\text{find } u = \sum_{i=1}^{n} c_i B_i \text{ such that:} \quad \int_\Omega \nabla u \, \nabla v = \lambda \int_\Omega u \, v \qquad \forall v \in w^e B_h^n \, .$$

Hence, we obtain the discrete eigenvalue problem $Au = \lambda Bu$ with a $n \times n$ dimensional mass matrix A and stiffness matrix B. The entries of the matrices are

$$a_{i,j} = \int_{\Omega} \nabla B_i \, \nabla B_j \, , \qquad b_{i,j} = \int_{\Omega} B_i \, B_j \, .$$

First, the domain Ω is discretized into a finite non-overlapping set

$$\Omega = \bigcup_{\ell=1}^{m} \Omega_\ell \, .$$

This is called the triangulation of the domain Ω, and an example is shown in figure 1.2. In 2 dimensions the Ω_ℓ could be triangles or rectangles and in 3 dimensions tetrahedrons or cuboids.
In chapter 2 we establish the basis of the weighted extended B-spline space $w^e \mathbb{B}_h^n$. B-splines play an important role in approximation of functions and data, computer graphics and computer aided design. We get the uniform B-spline b^n of degree n with

$$b^n(x) = \int_{x-1}^{x} b^{n-1}.$$

Here, we start with the characteristic function b^0 of the unit interval $[0, 1)$. For B-splines on a grid

$$h\mathbb{Z} : \quad \ldots, -2h, -h, 0, h, 2h, \ldots$$

we must transform the standard uniform B-spline b^n

$$b_{k,h}^n = b^h \left(\frac{x}{h-k} \right) .$$

The finite element space $\mathbb{B}_h^n(\Omega)$ is

$$\sum_{k \in K} c_k b_{k,b}^n$$

with relevant B-splines. The set K consists of relevant indices with $b_{k,h}^n(x) \neq 0$ for some $x \in \Omega$. In a m-dimensional space we take the tensor-product B-spline

$$b_{k,h}^n(x) = \prod_{\nu=1}^{m} b_{k_\nu,h}^{n_\nu}(x_\nu)$$

with order n_ν in the ν^{th} variable. The Dirichlet boundary condition is enforced by a weight function w with

$$w(x) = \begin{cases} > 0 & \text{in } \Omega \\ 0 & \text{on } \partial\Omega \end{cases} .$$

We multiply the weight function with the space $\mathbb{B}_h^n$ and we define the weighted B-spline space $w\mathbb{B}_h^n$

$$wB_h^n(\Omega) = \text{span}_{k\in K} wb_k .$$

The weighted B-splines, which have a very small support in the domain Ω are not stable for small grid width h. To solve this problem, we need a new space, the WEB-splines. Here, we divide the grid cells up into interior, boundary, and exterior grid cells. If the interior of a grid cell Q intersects $\partial\Omega$, it is a boundary cell and for $Q \subseteq \overline{\Omega}$ it is an interior cell. The support of an inner B-spline b_i consists of at least one inner cell Q. The support of an outer B-spline b_j consists of boundary and exterior cells. Now, we can distinguish inner B-splines b_i , $i \in I$ and outer B-splines b_j , $j \in J = K\backslash I$. For the WEB-spline basis we adjoin to an inner B-spline b_i outer B-splines b_j with $j \in J(i)$, which are near b_i

$$B_i = \frac{w}{w(x_i)} \left(b_i + \sum_{j\in J(i)} e_{i,j} b_j \right) , \qquad i \in I .$$

x_i could be the middle point of an inner grid cell of the inner B-spline b_i and $e_{i,j}$ are the extension coefficients. Figure 2.4 illustrates the extension coefficients of an outer B-spline and figure 2.3 an inner B-spline and outer B-spline.
Now, we have the finite element space

$$w^e\mathbb{B}_h^n = \text{span}_{i\in I} B_i .$$

This space was taken for the finite element approximation of the eigenvalue problem.
In chapter 3, we proved that the WEB-space has the same approximation order for the eigenvalue problem as other finite element spaces. The approximation order for WEB-splines with degree n is for the eigenvalue λ_ℓ and eigenfunction u_ℓ

$$\lambda_\ell^h - \lambda_\ell \le c_1 h^{2n} \lambda_\ell^{n+1} ,$$
$$\left\| u_\ell - u_\ell^h \right\|_0 \le c_2 h^{n+1} \lambda_\ell^{(n+1)/2} .$$

In section 3.1 we prove the approximation order. Here the boundary $\partial\Omega$ must be sufficiently regular and the weight function sufficiently smooth. In section 3.2 we calculate some examples. The first example is the unit circle. Here the eigenvalues are the roots of the bessel functions squared. In figure 3.1 the approximation order of the first and 10^{th} eigenvalue is illustrated. The order of the WEB-spline basis was one, two, three, and four. The second and third domain is shown in figure 3.3 and 3.5. We do not get the right approximation order in the last domain. This domain has edges.

When we have the mass and the stiffness matrices of the eigenvalue problem we can approximate the eigenvalues. There are many approaches: calculate the roots of the characteristic polynomial, inverse iteration, QR-iteration, ... The expenditure of time and the computational effort are too high in these procedures. For this reason we use another method. In this thesis the solver is based on the Lanczos algorithm and is explained in chapter 4. Here one approximates the eigenvalues and eigenvectors in Krylov subspaces

$$V_k(x) = \operatorname{span}\left\{x, Ax, \ldots, A^{k-1}x\right\} .$$

The basis of the Lanczos algorithm is a three term recursion. We receive for the eigenvalue problem

$$Au = \lambda u$$

algorithm 7.2.

Algorithmus 7.2 Lanczos Algorithm

choose v_1 with $\|v_1\| = 1$,
set $\beta_1 = 0$ and $v_0 = 0$
for $i = 1 : m$ **do**
  $w_i = Av_i$
  $\alpha_i = (w_i, v_i)$
  $w_i = w_i - \alpha_i v_i - \beta_i v_{i-1}$
  $\beta_{i+1} = \|w_i\|$
  $v_{i+1} = \frac{w_i}{\beta_{i+1}}$
end for

We get with the orthogonal matrix $X_k = [v_1, \ldots, c_k]$ the tridiagonal

Lanczos matrix T

$$X_k^t A X_k = T_k\,, \qquad T_k = \begin{pmatrix} \alpha_1 & \beta_2 & & & \\ \beta_2 & \alpha_3 & \beta_3 & & \\ & \ddots & \ddots & \ddots & \\ & & \beta_{k-1} & \alpha_{k-1} & \beta_k \\ & & & \beta_k & \alpha_k \end{pmatrix}.$$

The eigenvalues of the matrix T_k approximate k eigenvalues of A. Since T_k is a smaller and a tridiagonal matrix the eigenvalues and eigenvectors can easier be found as these of matrix A. Here the QR algorithm or the QL algorithm can be used. The approximate eigenvectors of the matrix A are $Xu_i\,, i = 1\,, \ldots\,, k$ with u_i the eigenvectors of T_k. When we obtain $\beta_\ell = 0$ for a $\ell = 1, \ldots, k$ then the subspace X_ℓ is invariant and

$$AX_\ell = X_\ell T_\ell\,.$$

Now the eigenvalues of T_ℓ are the exact eigenvalues of A.
The Lanczos vectors v_i should be orthogonal. But after some steps of the Lancos algorithm the orthogonality is lost. To avoid this phenomenom we could reorthogonalize the vectors v_i with full, partial or selective reorthogonalization. These procedures are explained in section 4.3.
There are some possibilities to improve algorithm 7.2. One is the implicitly restarted Lanczos algorithm. The vector v_0 in algorithm 7.2 is a random vector. When we want some eigenvalues with a desired property we can make a special linear combination of the Lanczos vectors and restart the Lanczos algorithm with this new vector. Now the new vector has bigger components in the subspace spanned by desired eigenvectors and smaller components in the direction of unwanted eigenvectors. The implicitly shifted Lanczos algorithm is based on the implicitly shifted QR factorization. For choosing the shifts there are several strategies. One is the exact shift strategy. Here we take the p eigenvalues which are furthest away from the wanted eigenvalues as shifts.
Needing eigenvalues in the middle of the spectrum, near σ, the spectral transformation

$$\underbrace{(A - \sigma B)}_{C}^{-1} u = \frac{1}{\lambda - \sigma} u \tag{7.7}$$

is used. Now we approximate the eigenvalues $\nu_i = \frac{1}{\lambda_i - \sigma}$. As seen in figure 4.3 the new eigenvalues have a large absolute value when they are near σ. The extreme eigenvalues are also well separated from the rest. When we

take the spectral lanczos algorithm the lanczos vectors $X_k = [v_1, \ldots, c_k]$ are B orthonormal, that means

$$X_k^t B X_k = I$$

with the identity matrix I.
The linear system of equations in 7.7 is solved with the multigrid algorithm. The main parts of the multigrid algorithm are the smoother and the grid transfer operator. The restriction is used to map a solution to a coarser grid and the prolongation is to get on a finer grid. The property of a smoother is that the error of the approximation becomes smoother. Figure 5.2 shows the smoothing property of the Richardson iteration. Here we see the error at the beginning, after 50 iterations and after 100 iterations. The error after 50 iterations, compared to 100 iterations, is not really smaller but smoother. Iterative solvers, such like the Gauss-Seidel iteration or the Jacobi iteration, converge very slowly but have the effect of a smoother. If the error is smooth, one can approximate the error on a coarser grid. With the spectral transformation the eigenvalues near σ were approximated well. In equation (7.7) it is possible, that the matrix C is not positive-definite. For this reason, we take another smoother in the multigrid cycle: incomplete LU matrix decomposation (ILU). The smoothing properties are good for non-positive-definite matrices. For non-positive-definite matrices the Gauss-Seidel smoother and Jacobi smoother fail. Without the spectral transformation these smoothers could be taken. But here preconditioning of the matrices upgrade the smoother properties. The smoothers are introduced in section 5.1. The multigrid algorithm which is explained in chapter 5 uses a nested sequence of grids with widths

$$h_{\max} > \cdots > 2h > h > \cdots > h_{\min}\,.$$

Two grids and the domain are shown in figure 5.9. We get the grid transfer operator with the subdivision formula for uniform B-splines

$$b_{k,h}^n = 2^{-n} \sum_{\ell=0}^{n+1} \binom{n+1}{\ell} b_{2k+\ell,\frac{h}{2}}^n$$

with $\binom{n+1}{\ell} = 0$ for $\ell < 0$ or $\ell > n+1$. Figure 2.2 illustrates the subdivision formula with a quadratic B-spline. The WB-space with grid width $2h$ is a subspace of the WB-space with grid width h. Because of the extension this fact does not apply to the WEB-space. The projection of a

function $\widetilde{U} = \sum \widetilde{u}_\ell \widetilde{B}_\ell$ on a grid with width $2h$ to a function $U = \sum u_i B_i$ on a grid with width h is $U = P\widetilde{U}$ with matrix P. The matrix P has the entries

$$\widetilde{p}_{i,\ell} = \frac{w(x_i)}{w(\widetilde{x}_\ell)} \left(s_{i-2\ell} + \sum_{u \in \widetilde{U}} \widetilde{e}_{\ell,u} s_{i-2u} \right) .$$

For a multi-dimensional WEB-spline s_i is the product of the binomial coefficients in each dimension. As a result we get the multigrid algorithm for the linear system

$$Ax = b .$$

Algorithmus 7.3 Static Multigrid Algorithm

$u_{lev} = \text{Multigrid}(u_{lev}, b_{lev}, lev)$
$u_{lev} = S^\alpha(u_{lev}, b_{lev})$
$r = R(A_{lev} u_{lev} - b_{lev})$
if $lev + 1 < lev_{max}$ **then**
  for $j = 1 : \beta$ **do**
    $v_{lev+1} = \text{Multigrid}(0, r, lev + 1)$
  end for
else if $lev + 1 = lev_{max}$ **then**
  $v_{lev+1} = A^{-1}_{lev+1} r$
end if
$u_{lev} = u_{lev} - P(u_{lev+1})$
$u_{lev} = S^\alpha(u_{lev}, b_{lev})$

In algorithm 7.3 we apply α smoothing operations $S^\alpha(u, b)$ with the vector b and the approximate solution u in a first step. Then we restrict the residual $Au - b$ to a coarser grid $R(Au - b)$. When we reach the coarsest grid we solve the equation exactly $u = A^{-1}b$. Now we project the solution on finer grids. β denotes the number of coarse multigrid iterations. We get for $\beta = 1$ the V-cycle and for $\beta = 2$ the W-cycle, as it is shown in figure 5.8. To improve the multigrid algorithm we can control the grid transfer dynamically. Here we need some new parameters. We apply the smoother as long as the smoothing operator converges fast. We project for a small error the solution to a finer grid otherwise we restrict the error to a coarser grid. We can get a cycle like in figure 5.10. Another improvement is starting multigrid algorithm not on the finest grid but on the coarsest grid. Then we reach the finest grid with a nested iteration. The combination of an iterative multigrid algorithm with a

nested iteration is more efficient than without the nested iteration. We get the full multigrid technique. An example is shown in figure 5.11. Chapter 6 describes the implementation of this eigenvalue problem in MATLAB. In Section 6.1 the buildup of the program is explained. We calculate the entries of the mass matrix and the stiffness matrix

$$a_{i,j} = \int_{\Omega} \nabla B_i \left(\nabla B_j\right)^t , \qquad b_{i,j} = \int_{\Omega} B_i B_j$$

with the gaussian quadrature rule. The integral is a weighted sum of function values at specific points x_i and with specific weights g_i

$$\int_a^b f(x)\,dx \approx \sum_{i=1}^{\ell} g_i f(x_i)\,.$$

The formula integrates polynomials up to order $2\ell-1$ exactly. The Gauss points x_i and the Gauss weights g_i are given for $a = -1$ and $b = 1$ in table 6.1. We calculate multi dimensional domains with tensor-product gaussian quadrature rule and we transform the points and weights for a boundary cell. First we calculate the mass matrix A and the stiffness matrix B with weighted B-splines. We obtain two sparse matrices. Then the extension matrix E is assembled. Matrix A and matrix B in the WEB-space are

$$A^e = E \cdot A \cdot E^t\,, \qquad B^e = E \cdot B \cdot E^t\,.$$

The matrices A^e and B^e are also sparse matrices. The allocation structure of matrix A and A^e is shown in figure 6.1 and 6.2, the extension matrix in 6.3. We need these matrices on every grid for the multigrid solver. Now we solve the eigenvalue problem with the implicitly restarted Lanczos method. The smoother in the multigrid cycle is the ILU factorization. In section 6.2 we explain the input values in MATLAB. The boundary should be a Bézier curve. An example for a Bézier curve is shown in figure 6.4. We also need an implicite weight function. For an ellipse with semi-major axis a und semi-minor axis b we get the weight function

$$w(x,y) = 1 - \frac{x^2}{a^2} - \frac{y^2}{b^2}\,.$$

We can calculate the matrices A and B with the given boundary and weight function or load matrices which were calculated earlier. We need

some parameters for the multigrid cycle. How many Lanczos steps and smoothing steps should be made? We can declare the range in which we want to know the eigenvalues and the maximal number of eigenvalues. The finest grid could be displayed in the main menu as it is shown in figure 6.5. We can choose between a V-cycle, a W-cycle or something else. In section 6.3 we compute many eigenvalues in three domains. First we take the unit circle, because here we exactly know the eigenvalues. We calculate the eigenvalues between zero and 5000. In this region we find 1214 eigenvalues. Here the grid width is 2^{-7} and the degree of the WEB splines is three. The size of the matrices without extension is 67600×67600. The second domain is shown in figure 3.3 and the third domain is the square with edge length π. In the second domain we calculate the eigenvalues between zero and 2500 (2771 eigenvalues) and in the third domain between zero and 2000 (1529 eigenvalues).
In chapter 7 surface waves in water in closed domains $\Omega \subset \mathbb{R}^2$ with the partial differential equation

$$\begin{aligned} -\nabla^{\mathrm{t}} \left(H \nabla u\right) &= \lambda u \qquad \text{in } \Omega \\ \frac{\partial u}{\partial n} &= 0 \qquad \text{on } \partial\Omega \end{aligned} \tag{7.8}$$

are investigated. Here the partial differential equation is the wave equation. Also acoustic and electromagnetic waves can be represented with equation (7.8). The domain Ω could be a closed 2-dimensional basin with varying depth. $H(x, y)$ is the water depth at every point (x, y), λ the natural frequencies of oscillation of the water and u the mode shape of the water surface. There are analytical solutions for the circle with a constant depth. For a circle with radius one and depth one, the eigenvalues are the maxima and minima of the bessel functions squared. The grid width must be small enough to ensure an accurate solution. The wave elevation u should be small in comparison with the water depth H. In this example we have Neumann boundary conditions. The weight function is only needed to describe the domain Ω and not for the finite element basis. The boundary conditions describe a vertical, perfectly reflecting wall. Since in this example time is more important than accuracy only linear box-splines b_i without extension are implemented. The boundary is also approximated with a linear function $f(x, y) = ax + by + c$. The integrals of the variational formulation

$$\int_{\Omega} H \nabla b_i \, \nabla b_j = \lambda \int_{\Omega} b_i \, b_j$$

were calculated and stored with parameters of the water depth and weight function at every edge of a cell. Thus the stiffness and mass matrices can be easily and quickly assembled. In this chapter we also focus on parallelization. The program was implemented in Fortran90. We calculate only the smallest eigenvalues. For this reason all matrices are positive definite since in equation (7.7) $\sigma = 0$ thus $A = C$. The eigenvector to the eigenvalue zero is $w_1 = (1, 1, \ldots, 1)^{\mathrm{t}}$. We run through the Lanczos algorithm with vectors which are orthogonal to w_1. Hence the SSOR iteration converges. The SSOR iteration is not parallelizable but the multi-color SSOR iteration is. So we take the multi-colour SSOR iteration as solver in the dynamic multigrid. The main part of the Baltic Sea is the last example for water waves. In section 7.1 the theoretical background is explained, for example the partial differential equations, the linearization of the weight function and the entries in the matrices. In section 7.2 the basis of the finite element space and the solver is explained. Figures 7.2 and 7.3 describe the support and the shape of a linear box-spline. In section 7.3 the integration is described. We assert the computation of the formulas which are saved. Later we must only insert the values of the weight function and the water depth at every edge of a cell. In section 7.4 some examples are calculated. The first example is the unit circle with water depth one. Here we know the exact eigenvalues. We detect in figure 7.7 and 7.8 the approximation order two of the eigenvalues. Figure 3.3 illustrates the second domain. We take

$$H(x, y) = 2 + \mathrm{abs}\left(1 - x^2 - y^2\right)$$

as water depth. Now we do not know the exact eigenvalues. Figure 7.9 and 7.10 map the approximation order in this example. The first four mode shapes are displayed in figure 7.11. The aspect of parallelization is shown in section 7.5.

In this thesis we show that the WEB-spline basis is a good alternative to standard finite element approximations for the eigenvalue problem. Some advantages of the WEB-splines are:

- no grid generation
- exact fulfillment of essential boundary conditions
- hierarchical basis for multigrid
- natural parallelism
- same approximation order

The WEB-splines combine the advantages of uniform B-splines and finite elements on unstructured grids. B-splines are used in geometric modeling and numerical simulation, two parts which appear in many engineering applications.

Anhang A

Formeln

A.1 Dreiecksungleichung

Für komplexe Zahlen oder Vektoren a, b und c gilt:

$$|a| - |b| \leq |a - b| \leq |a - c| + |c - b|$$

A.2 Rayleigh-Quotient

Die Funktion mit einer symmetrischen Matrix $K \in \mathbb{R}^{n \times n}$ und einer symmetrischen regulären Matrix $M \in \mathbb{R}^{n \times n}$

$$R(q) = \frac{q^T K\, q}{q^T M\, q}$$

bezeichnet man als Rayleigh-Quotient. Dabei ist $q = [q_1, q_2, \ldots, q_n]^{\mathrm{t}}$. M und K können die entsprechenden FEM-Matrizen sein.

A.3 Galerkin-Orthogonalität

Sei u die Lösung der Differentialgleichung

$$-\Delta u = f$$

und Pu die WEB-Standard- Projektion, dann gilt

$$a(u - Pu, w_h) = 0 \qquad \forall w_h \in \mathbb{B}_h\,.$$

A.4 Minimum-Maximumprinzip von Courant-Fischer

Für die Eigenwerte $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ des Eigenwertproblems

$$Ku = \lambda Mu$$

mit einer symmetrischen Matrix $K \in \mathbb{R}^{n\times n}$ und einer symmetrischen regulären Matrix $M \in \mathbb{R}^{n\times n}$ gilt

$$\begin{aligned}
\lambda_\ell^h &= \min_{S_\ell \subset \mathbb{R}^n} \max_{v^h \in S_\ell} R\left(v^h\right) && \text{mit } \dim S_\ell = \ell\,, \\
\lambda_\ell^h &= \max_{S_{n-\ell+1} \subset \mathbb{R}^n} \min_{v^h \in S_{n-\ell+1}} R\left(v^h\right) && \text{mit } \dim S_{n-\ell+1} = n-\ell+1\,.
\end{aligned}$$

A.5 Jackson-Ungleichung

Ist w eine Standardgewichtsfunktion, dann gilt

$$\|u - Pu\|_\ell \leq \text{const}(D, w, n)\, h^{k-\ell} \|u\|_k\,, \qquad l < k \leq n+1\,,$$

für jede Funktion $u \in H^k$ mit Nullrandbedingungen.

Literaturverzeichnis

[Alt06] H. W. Alt, *Lineare Funktionalanalysis, Eine anwendungsorientierte Einführung*, Springer-Verlag, Berlin, 2006.

[Arg57] J. H. Argyris, *Die Matrizentheorie der Statik*, Seiten 174–192, Ingenieur-Archiv 25, 1957.

[B66] P. Bézier, Définition numérique des courbes et surfaces I, *Automatisme XI*, Seiten 625–632, 1966.

[B67] P. Bézier, Définition numérique des courbes et surfaces II, *Automatisme XII*, Seiten 17–21, 1967.

[B72] P. Bézier, *Numerical Control-Mathematics and Applications*, John Wiley & Sons, London, 1972.

[Bac04] H. Bachmann, „Lebendige“ Fußgängerbrücken–eine Herausforderung, *Bautechnik*, Band 81, Nummer 4, Seiten 227–236, 2004.

[BO91] I. Babuska und O. Osburn, Eigenvalue problems, in P. G. Ciarlet und J. L. Lions (Herausgeber), *Finite element Methods (Part 1)*, Band 2 von *Handbook of Numerical Analysis*, Seiten 641–788, Elsevier, Amsterdam, 1991.

[Bra77] A. Brandt, Multi-level adaptive solutions to boundary-value problems, *Mathematics of Computation*, Band 31, Nummer 138, Seiten 333 – 390, 1977.

[Bra03] D. Braess, *Finite Elemente: Theorie, schnelle Löser und Anwendungen in der Elastizitätstheorie*, Springer, Berlin, 2003.

[BSMM01] I. N. Bronstein, K. A. Semenjajew, G. Musiol und H. Mühlig, *Taschenbuch der Mathematik*, Harri Deutsch, Frankfurt am Main, 2001.

[BW83] G. Berendt und E. Weimar, *Mathematik für Physiker*, Band 2, Physik-Verlag, Weinheim, 1983.

[Böh80] W. Böhm, Inserting new knots into B-spline curves, *Computer-Aided Design*, Band 12, Seiten 199–201, 1980.

[CLR80] E. Cohen, T. Lyche und R.Riesenfeld, Discrete B-splines and subdivision techniques in computer aided geometric design and computer graphics, *Computer Graphics Image Processing*, Band 14, Seiten 87–111, 1980.

[Cou43] R. Courant, Variational methods for the solution of problems of equilibrium and vibrations, *Bulletin of the American Mathematical Society*, Band 49, Seiten 1–23, 1943.

[CW85] J. K. Cullum und R. A. Willoughby, *Lanczos Algorithms for Large Symmetric Eigenvalue Computations, Vol. 1 Theory*, Birkhäuser, Boston, 1985.

[dB72] C. de Boor, On calculating with B-splines, *Journal of the Approximation Theory*, Band 6, Seiten 50 – 62, 1972.

[dB77] C. de Boor, Package for calculating with B-splines, *SIAM Journal on Numerical Analysis*, Band 14, Seiten 441 – 472, 1977.

[dBHR93] C. de Boor, K. Höllig und S. Riemenschneider, *Box Splines*, Springer, New York, 1993.

[dC59] P. de Casteljau, Courbes et surfces à pôles, Technischer Bericht, André Citröen Automobiles SA, Paris, 1959.

[Dem97] J. W. Demmel, *Applied Numerical Linear Algebra*, Kapitel Iterative Methods for Eigenvalue Problems, Seiten 361–388, siam, Philadelphia, 1997.

[DH91] P. Deuflhard und A. Hohmann, *Numerische Mathematik: Eine algorithmisch orientierte Einführung*, de Gruyter, Berlin, 1991.

[ER80] T. Ericsson und A. Ruhe, The spectral transformation lanczos method for the numerical solution of large sparse generalized symmetric eigenvalue problems, *Mathematics of Computation*, Band 35, Seiten 1251 – 1268, 1980.

[HAS05] K. Höllig, C. Apprich und A. Streit, Introduction to the web-method and its applications, *Advances in Computational Mathematics*, Band 23, Seiten 215–237, 2005.

[Heu03] V. Heuveline, On the computation of a very large number of eigenvalues for selfadjoint elliptic operators by means of multigrid methods, *Journal of Computational Physics*, Band 183, Nummer 1, Seiten 321–337, 2003.

[HHK03] K. Höllig, J. Hörner und A. Kopf, Web-spline Methods for Linear Elasticity, in A. Cohen, J.-L. Merrien und L.L. Schumaker (Herausgeber), *Curve and Surface Fitting: Saint-Malo 2002*, Seiten 219–228, Nashboro Press, Brentwood, 2003.

[HHP07] K. Höllig, J. Hörner und M. Pfeil, Parallel Finite Element Methods with Weighted Linear B-Splines, Preprint, IMNG, Universität Stuttgart, 2007.

[HR03] K. Höllig und U. Reif, Nonuniform Web-Splines, *Computer Aided Geometric Design*, Band 20, Nummer 5, Seiten 277–294, 2003.

[HRW01a] K. Höllig, U. Reif und J. Wipper, B-Spline approximation of Neumann problems, Preprint 2001-2, Mathematisches Institut A, Universität Stuttgart, 2001.

[HRW01b] K. Höllig, U. Reif und J. Wipper, Error estimates for the web-method, in T. Lyche und L.L. Schumaker (Herausgeber), *Mathematical Methods for Curves and Surfaces: Oslo 2000*, Seiten 195–209, Vanderbilt University Press, Nashville, TN, 2001.

[HRW01c] K. Höllig, U. Reif und J. Wipper, Weighted extended b-spline approximation of Dirichlet problems, *SIAM Journal on Numerical Analysis*, Band 39, Nummer 2, Seiten 442–462, 2001.

[HRW02] K. Höllig, U. Reif und J. Wipper, Multigrid methods with web-splines, *Numerische Mathematik*, Band 91, Nummer 2, Seiten 237–256, 2002.

[HÖuP98] W. Heinze, C.M. Österheld und H. Kossira und P.Horst, Berücksichtigung aeroelastischer Fragestellungen beim Flugzeugvorentwurf, Technischer Bericht, Weinheim, 1998.

[Höl98] K. Höllig, *Grundlagen der Numerik*, Math Text, Zavelstein, 1998.

[Höl02] K. Höllig, Finite Element Approximation with Splines, in G. Farin, J. Hoschek und M.S. Kim (Herausgeber), *Handbook of Computer Aided Geometric Design*, Seiten 283–308, Elsevier, 2002.

[Höl03] K. Höllig, *Finite Element Methods with B-Splines*, siam, Philadelphia, 2003.

[Kom03] L. Komzsik, *The Lanczos Method: Evolution and Application*, siam, Philadelphia, 2003.

[Kui00] A.B.J. Kuijlaars, Which eigenvalues are found by the lanczos method?, *SIAM Journal on Matrix Analysis and Applications*, Band 22, Nummer 1, Seiten 306–321, 2000.

[Lam07] H. Lamb, *Lehrbuch der Hydrodynamik*, Teubner, Leipzig und Berlin, dritte Auflage, 1907.

[Lan50] C. Lanczos, An iteration method for the solution of the eigenvalue problem of linear differential and integral operators, *Journal of research of the National Bureau of Standards*, Band 45, Nummer 4, Seiten 255–282, 1950.

[Mei05] A. Meister, *Numerik linearer Gleichungssysteme: Eine Einführung in moderne Verfahren*, Vieweg, Wiesbaden, 2005.

[Mö06] B. Mößner, *B-Splines als Finite Elemente*, Shaker Verlag, Aachen, 2006.

[NOPEJ87] B. Nour-Omid, B. N. Parlett, T. Ericsson und P. S. Jensen, How to implement the spectral transformation, *Mathematics of Computation*, Band 48, Nummer 178, Seiten 663 – 673, 1987.

[PS79] B. N. Parlett und D. S. Scott, The lanczos algorithm with selective orthogonalization, *Mathematics of Computation*, Band 33, Nummer 145, Seiten 217–238, 1979.

[Saa03] Y. Saad, *Iterative Methods for Sparse Linear Systems*, siam, Philadelphia, 2003.

[Sch51] K. Schellbach, Probleme der Variationsrechnung, *Journal für die reine und angewandte Mathematik*, Band 41, Nummer 4, Seiten 293–363, 1851.

[Sch46] I. J. Schönberg, Contributions to the problem of approximation of equidistant data by analytic functions, *Quarterly of Applied Mathematics*, Band 4, Seiten 45–99 und 112–141, 1946.

[SF88] G. Strang und G. J. Fix, *An Analysis of the Finite Element Method*, Kapitel Eigenvalue Problems, Seiten 216–240, Wellesley-Cambridge Press, Massachusetts, 1988.

[Sim84] D. Simon, The lanczos algorithm with partial reorthogonalization, *Mathematics of Computation*, Band 42, Nummer 165, Seiten 115–142, 1984.

[Sor95] D. C. Sorensen, Implicitly restarted Arnoldi/Lanczos methods for large scale eigenvalue calculations, 1995.

[TB97] L. N. Trefethen und D. Bau, *Numerical Linear Algebra*, siam, Philadelphia, 1997.

[TCMT56] M. J. Turner, R. W. Clough, H. C. Martin und L. J. Topp, Stiffness and deflection analysis of complex structures, *J. Aeronaut. Sci.*, Band 23, Nummer 9, Seiten 805–823, 1956.

[TOS01] U. Trottenberg, C. W. Oosterlee und A. Schüller, *Multigrid*, Academic Press, San Diego, California, 2001.

[Wip05] J. Wipper, *Finite-Elemente-Approximation mit WEB-Splines*, Shaker Verlag, Aachen, 2005.

[Zie67] O. C. Zienkiewicz, *The Finite Element Method in Structural and Continuum Mechanics*, McGraw-Hill, New York, erste Auflage, 1967.

[ZT00] O. C. Zienkiewicz und R. L. Taylor, *The Finite Element Method, Volume 3: Fluid Dynamics*, Butterworth-Heinemann, Oxford, fünfte Auflage, 2000.

Index