Pseudoinverse für zell-basierte finiten Elemente Operatoren

Bachelorarbeit

eingereicht von

Enes Witwit

betreuut von

Prof. Dr. Kanschat

Fakultät für Mathematik und Infromatik

Universität Heidelberg

Hiermit versichere ich, Enes Witwit, dass ich die vorliegende Bachelorarbeit selbständig verfasst habe. Ich versichere, dass ich keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommenen Aussagen als solche gekennzeichnet habe, und dass die eingereichte Arbeit weder vollständig noch in wesentlichen Teilen Gegenstand eines anderen Prüfungsverfahren gewesen ist.

14. Mai 2017 Heidelberg Unterschrift

Zusammenfassung

Die folgende Arbeit konzentriert sich auf die Berechnung der Pseudoinversen von der Masse Matrix und der Steifigkeitsmatrix der Laplace Bilinearform, insbesondere die effiziente Berechnung des Matrix-Vektor Produkts mit der Pseudoinversen der genannten Matrizen. Insgesamt werden zwei verschiedene Ansätze hergeleitet. Der erste versucht das Ziel durch eine Singulärwertzerlegung höherer Ordnung zu erreichen. Der zweite Ansatz nutzt die Struktur der genannten Matrizen um eine einfache Berechnung der Pseudoinversen über eine einfache Singulärwertzerlegung. Mit der zweiten Methode erreichen wir insgesamt für die Berechnung der Pseudoinversen und dem Matrix-Vektor-Produkt eine Komplexität von $O(n^3)$, was erstaunlich ist. Hingegen stoßen wir beim ersten Ansatz auf diverse Probleme bezüglich der einfachen Berechnung des Produktes und der Komplexität.

Inhaltsverzeichnis

Notation

$a(\cdot,\cdot)$	Bilinearform
D^α	Ableitung $\|\alpha\|$-ter Ordnung zum Multiindex α
I	Identität
$J(\cdot)$	Funktional
$O(\cdot)$, $o(\cdot)$	Landau Symbole
P_l	Menge aller Polynome vom Höchstgrad l
$\mathbb{R}, \mathbb{N}$	reelle Zahlen, natürliche Zahlen
V	Banachraum
$dim(V)$	Dimension von V
V_h	endlichdimensionaler Finite-Elemente-Raum
$\|\|\cdot\|\|_V$	Norm von V
$(\cdot,\cdot)$	Skalarprodukt in V, wenn V Hilbertraum
$f(v)$ oder $\langle f, v\rangle$	Wert des Funktionals $f \in V^*$ bei Anwendung auf $v \in V$
Ω	Grundgebiet bezüglich der räumlichen Veränderlichen
$\delta\Omega$	Rand von Ω
$int\Omega$	Inneres von Ω
$C^l(\Omega), C^{l,\alpha}(\Omega)$	Räume differezierbarer bzw. Hölder-stetiger Funktionen
$L_p(\Omega)$	Räume zur p-ten Potenz integrabler Funktionen $(1 \leq p \leq \infty)$
$W_p^l(\Omega)$	Sobolev-Raum
$H^l(\Omega), H_0^l(\Omega)$	Sobolev-Räume für p=2
$supp(v)$	Träger der Funktion v
Δ	Gradient

div	Divergenz
∇	Laplace-Operator
∇_h	Diskreter Laplace-Operator
h_i , h	Diskretierungsparamter bezüglich der räumlichen Veränderlichen
$\mathfrak{X}$	Skript Buchstaben für Höher-dimensionale Tensoren
$\mathbf{A}$	Große fette Buchstaben für Matrizen
$\mathbf{A}^{(n)}$	bezeichnet die n-te Matrix einer Matrixfolge

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einführung

Hochleistungsrechnen ist eine neue Disziplin, die mit dem Drang entstand, immer komplexere Probleme zu lösen. Diese neue Art an Probleme ranzugehen, löst sie nicht. Doch sie eröffnet uns eine neue Sichtweise auf dasselbe Probleme und gibt uns am Ende womöglich neue Lösungsansätze. Parallelisierung ist ein großer Zweig des Hochleistungsrechnens. Das Grundgerüst dieser Methodik besteht darin komplexe Probeme modular zu lösen, das heißt sie in viele wenig komplexere Subprobleme zu unterteilen, die unabhängig voneinander berechenbar sind. Am Ende fasst man die Ergebnisse der Subprobleme zu einem Ergebnis zusammen, welches die Lösung des komplexen Problems darstellt.

$$v = A(u) = \sum_{k=1}^{n_{cells}} C^T P_k^T A_k (P_k C u) \tag{1.1}$$

ist die Gleichung, welche wir mit Hilfe der finite Elemente Methode lösen wollen. A ist ein möglicherweise nichtlinearer Operator, der Vektor u als Input nimmt und das Integral vom Operator multipliziert mit Testfunktionen ϕ_i mit $i = 1, \ldots, n$, berechnet [**?**, 136]. Damit wir diese Gleichung besser nachvollziehen können, werden wir sie im nächsten Kapitel herleiten. Doch um diese Arbeit zu motivieren kann man sich diese Gleichung so vorstellen, dass man ein großes Problem in eine Summe von kleineren Problemen unterteilt.

Nun die Matrix A_k kann durchaus, wie wir nachher erkennen werden, sehr groß werden. Wenn sie so groß wird, dass sie nicht mehr im Cache liegt, bekommen wir das Problem, dass der Zugriff auf Elemente der Matrix sehr teuer wird. Dementsprechend wollen wir uns vorallem bei der Herleitung unserer Methoden um diese Subprobleme zu lösen auf sogenannten matrix-freie Heransgehensweisen konzentrieren. Das bedeutet wir wollen nicht explizit die Matrix A_k ausrechnen, sondern stattdessen ihre Elemente dort berechnen wo sie auch gebraucht werden. Ob dies möglich ist, ist für diese Arbeit von großer Bedeutung.

Der Sinn dieser Arbeit ist einen effiziente Ansatz herzuleiten, der uns

$$A_k^+ v_k \tag{1.2}$$

berechnet, wobei A_k^+ eine Pseudoinverse darstellt. Dies kann als Präkonditionierer genutzt werden, um (**??**) Gleichung zu lösen. Das heißt die Resultate dieser Arbeit werden benutzt, um damit einen Präkonditionierer zu bauen. Daraus folgern wir,

dass die Exaktheit der Lösung von (**??**) eine redundante Rolle spielt, vielmehr sollten wir eine optimale Lösung im Trade-Off zwischen Genauigkeit und Komplexität anstreben.

Im zweiten Kapitel werden wir erstmal en theoretisches Grundgerüst schaffen, um die beiden Gleichungen besser zu durchleuchten und nachvollziehen zu können. Wir werden uns die Grundlagen der numerischen Behandlung von partiellen Differentialgleichungen anschauen und versuchen die oberen Gleichungen herzuleiten. Im zweiten Teil des zweiten Kapitels werden wir uns mit Tensor Dekomposition beschäftigen. Dies liegt daran, dass wir uns den Operator A als Tensor umdefinieren können und die Pseudoinverse mit Hilfe der sogenannten Singulärwertzerlegung höherer Ordnung berechnen können.

Im dritten Kapitel werden wir uns zwei Methoden anschauen die Pseudoinverse zu berechnen. In der erste Methode die Struktur des Operators A zu nutze und leiten eine Repräsentation von A her die uns die Pseudoinverse mit wenig Aufwand gibt. In der zweiten Methode werden wir uns wie bereits erwähnt, A als Tensor umdefinieren und versuchen die SIngulärwertzerlegung höherer Ordnung effizient zu berechnen und uns dort auch Strukturen vom Tensor zu nutze zu machen.

Im vierten Kapitel sprechen wir über die effiziente Implementierung beider Algorithmen und im fünften Kapitel fassen wir alle Resultate zusammen und schließen mit einem Fazit ab.

2 Theorie

Das Ziel dieses Kapitels ist es die Verständigung über Notation zu klären und eine theoretische Grundlage für das dritte Kapitel, das Herz dieser Arbeit, zu schaffen.

Zu erst behandeln wir einen funktionalanalytischen Ansatz für elliptische Probleme. Begriffe wie schwache Lösung, klassische Lösung, Variationsgleichung und Sobolev Raum werden geklärt. Dies liefert uns das Basiswissen für die Galerkin Methode. Im dritten Unterkapitel *Methode der Finiten Elemente* ist das Ziel die Gleichung $Au = f$ herzuleiten mit A als globale Steifigkeitsmatrix. Im darauffolgenden Kapitel wollen wir uns eine flexiblere Methode anschauen, als die Methode der Finiten Elemente und die Wichtigkeit zu dieser Arbeit soll besonders hervorgehoben werden.

Dann kommen wir zu einem ganz anderen Thema, nämlich dem der Tensor Dekomposition. Erstmal schauen wir uns grundlegende Definition an um mit diesem Basiswissen die Singulärwertzerlegung höherer Ordnung einzuführen. Mit dieser werden wir uns gewisse Finite Elemente Operatoren als Tensoren umdefinieren und uns deren Zerlegung anschauen. Wir werden diese Untersuchungen in Kapitel drei und vier durchführen. In Hoffnung natürlich, dass die Zerlegung einfach ist und daraus die Pseudoinverse herleitbar ist.

2.1 Numerische Behandlung partieller Differentialgleichungen

2.1.1 Schwache Lösungen

Gegeben sei folgendes Randwertproblem

$$\begin{aligned} -\Delta u &= f \text{ in } \Omega \\ u &= 0 \text{ in } \delta\Omega \end{aligned} \tag{2.1}$$

Der funktionalanalytische Ansatz schlägt vor, dass wir uns zu erst mit notwendigen Funktionenräumen beschäftigen und uns auf analytischer Ebene eine Umformulierung der Differentialgleichung zu nutze machen, welche uns letztlich die Grundlage für die finiten Elementen Methode liefert.

Wir sehen von der Form der Differentialgleichung, dass die gesuchte Lösung bestimmte Differenzierbarkeits- und Stetigkeitsbedingungen zu erfüllen hat. Nämlich, dass u zweimal stetig differenzierbar sein sollte. Dementsprechend legen die Differenzierbarkeitsanforderungen den Raum $u \in C_0^2$ nahe.

Nun kann es aber sein, dass eine Lösung für dieses Problem nicht in diesem Raum existiert. Wir können uns dafür als Beispiel die Betragsfunktion anschauen.

$$f(x) = |x| \tag{2.2}$$

Offensichtlich ist diese Funktion in Null nicht stetig differenzierbar. Trotzdem können wir eine Ableitung finden, die wir *schwache Ableitung* nennen.

$$f'(x) = \begin{cases} -1 \text{ für } x < 0 \\ 0 \text{ für } x = 0 \\ 1 \text{ für } x > 0 \end{cases} \tag{2.3}$$

Nun um das was wir jetzt für die Betragsfunktion gemacht haben, für unser Randwertproblem zu machen nutzen wir eine funktionalanalytische Idee die aus der Distributionstheorie stammt. Wir multiplizieren mit einer Testfunktion ψ und integrieren über das Gebiet.

$$\int_\Omega -\Delta u \psi dx = \int_\Omega f \psi dx \tag{2.4}$$

Der nächste Schritt, welcher durchwegs fundamental für die Herleitung ist, ist die intuitive Nutzung der Struktur und Integrationswerkzeuge um zu erreichen, dass an u weniger Differenzierbarkeitsanforderungen gebunden sind. Für diesen Schritt ist der Satz von Green und die partielle Integration von Wichtigkeit.

Lemma 2.1. *Satz von Green*

Lemma 2.2. *Partielle Integration*

Für unsere Differentialgleichung (**??**) nutzen wir die partielle Integration und erhalten.

$$\int_\Omega -\nabla u \nabla \psi dx = \int_\Omega f \psi dx \tag{2.5}$$

Dies ist die so genannte Variationsgleichung. Sie ist ein erster Indiz für die später gewünschte Bilinearform der zugrundeliegenden Topologie. Die Lösung u von (**??**) nennt man *schwache Lösung* für das Problem (**??**). Die Lösung $u \in C_0^2$ von (**??**) nennt man *klassische Lösung*. Nun wissen wir in welchem Raum die klassische Lösung liegt, aber welche Topologie ist für die schwache Lösung sinnvoll? Ein funk-

tionalanalytischer Ansatz versucht nun die Räume zu definieren, in der die Lösung u für (**??**) liegt. In unserem Fall wäre folgender Raum ergiebig:

$$H_0^1(\Omega) = \{v \in L_2(\Omega) : \frac{\delta v}{\delta x_i} \in L_2(\Omega), v = 0 \text{ in } \delta\Omega \ , \ i = 1, ..., d\}$$

Diese Räume nennt man Sobolev Räume. Allgemein sind sie definiert durch:

Definition 2.3. *Sobolev Raum*

Das heißt Sobolev Räume sind eine Teilmenge von den L_2 Räumen. Von der analytischen Perspektive ist die Wahl des Funktionenraumes essentiell für den Nachweis der Existenz der Lösung. Von der Perspektive der finiten Elementen Methode ist dies für die Fehlerabschätzung wichtig, da wir dann die induzierte Norm des Funktionenraumes benutzen [**?**, 36]. Beide genannten Themen würden den Rahmen dieser Bachelorarbeit sprengen, daher verweis ich an gegebenen Stellen an weiterführende Literatur.

Die Sobolev Räume wurden mit Skalarprodukten ausgestattet, sodass unsere Variationsgleichung intuitiv als Skalarprodukt der zugrundeliegenden Sobolev Räume geschrieben werden kann.

Lemma 2.4. *Sobolev Norm*

Lemma 2.5. *Sobolev Skalarprodukt*

Lemma 2.6. *Falls die Bilinearform symmetrisch ist $u \in V$ ein Minimierer der Gleichung*

$$J(u) = \min_{v \in V} J(v) = \frac{1}{2}a(u, v) - f(v) \tag{2.6}$$

genau dann wenn u die schwache Formulierung löst.

2.1.2 Galerkin Verfahren

(Numerik 2 Skript Kanschat) Unser Ausgangspunkt ist nun

$$\text{Finde } u \in \Omega : a(u, v) = f(v) \forall u \in \Omega \text{ und } v \in V \tag{2.7}$$

Da die Sobolev Räume unendlich dimensional sind, ist es schwer sich mit der Konstruktion einer Lösung vertraut. Daher ist die Kernidee des Galerkin Verfahrens unseren Banachraum zu diskretisieren. Dazu führen wir eine sogenannte konforme

Approximation durch. Wir wählen $V_n \subset V$, sodass $dim V_n = n < \infty$. Nun gilt für die Minimierung in (**??**)

$$u = \arg\min_{v \in V} J(v) \text{ (stetig)}$$
$$u_n = \arg\min_{v_n \in V_n} J(v_n) \text{ (diskret)}$$

Daraus folgt

$$J(u_n) \geq J(u)$$

Dies folgt direkt aus der Wahl des Raumes als Teilraum des ursprünglichen Raumes. Man nennt diese Methode konforme Ritz-Galerkin Methode aus dem Grund, dass der diskrete Raum ein Teilraum von dem ursprüngichen Raum is und die Funktion J gleich bleibt. Was hat uns das Ganze gebracht? Nun da, $dim V_n = n < \infty$ können wir eine Basis für V_n. Das bringt uns den Vorteil, dass wir u_n als Linear Kombination der Basiselemente in V_n approximieren können.

Die schwache Formulierung sieht nun wie folgt aus

Lemma 2.7. *Schwache Formulierung (Galerkin)*
Finde $u_n \in V_n$, sodass $a(u_n, v_n) = f(v_n)\ \forall v_n \in V_n$.

Sei nun $e_1, \ldots, e_n$ eine Basis von V_n. Es ist nun ausreichend nur die Basis zum Testen zu nutzen. Die obere Gleichung in Lemma **??** reduziert sich auf

$$a(u_n, e_i) = f(e_i) \forall i \in \{1, \ldots, n\} \tag{2.8}$$

Im nächsten Schritt erweitern wir u_n als Linearkombination wie folgt

$$u_n = \sum_{j=1}^{n} u_j e_j \tag{2.9}$$

und setzen dies in **??** ein und erhalten.

$$\begin{aligned} &a(\sum_{j=1}^{n} u_j e_j, e_i) = f(e_i) \Longleftrightarrow \\ &\sum_{j=1}^{n} u_j a(e_j, e_i) = f(e_i) \end{aligned} \tag{2.10}$$

Das können wir zusammenfassen in einem Linearen Gleichungssystem mit $A_{ij} =$

$a(e_j, e_i)$ und $u = (u_1, \ldots, u_n)^T$. Insgesamt erhalten wir

$$Au = f \tag{2.11}$$

Bemerkung 2.8. *Eigenschaften Galerkin*

1. *Galerking Orthogonalität*
   Eine Kerneigenschaft der Galerkin Methode ist, dass der Fehler orthogonal zu dem Teilraum von V liegt.

2. *Symmetrie*
   Die Matrix A ist genau dann symmetrisch, wenn die Bilinearform symmetrisch ist.

2.1.3 Methode der finiten Elemente

Im Vorherigen Kapitel haben wir das Ritz-Galerkin Verfahren kennengelernt. Der Kernaspekt dieser konformer Approximation war eine diskretisierung des Raumes und mit einhergehend global einheitlich definierte Funktionen. Nun öffnen wir die letzt genannte Einschränkung und fordern nur noch stückweise definierte Funktionen, in der Regel Polynome. Wo genau eine Funktion definiert ist, hängt von unsererer Gebietszerlegung ab. Das heißt für die Finite Elemente Methode (FEM) ist es zu erst notwendig das Grundgebiet in geometrisch einfache Teilgebiete $\Omega_h = \{\Omega_k\}_{k=1\ldots N}$ z.B. Dreiecke und Rechtecke bei Problemen in der Ebene oder Tetraeder und Quader bei Problemen im dreidimensionalen Raum. Dann folgt eine Definition von Ansatz- und Testfunktionen über Teilgebieten. Da wir zwischen den Teilgebieten eine Stetigkeit fordern, definiert man Übergangsbedigungen die dann die Sicherung der Stetigkeit global sichert. (Grossmann Seite 175). Die Stetigkeit wird gefordert damit wir $V_n \subset V$ bekommen, da beispielsweise gelten kann $V \subset H^1(\Omega)$.

Bemerkung 2.9. *Natürliche Voraussetzungen an Zerlegung (Grossman S.176)*
Die Voraussetzungen an die Zerlegung $\mathscr{Z}=\{\Omega_j\}_{j=1}^m$ sind

1. $$\bar{\Omega} = \bigcup_{j=1}^{m} \bar{\Omega}_j$$

2. $$int\Omega_i \cap int\Omega_j = \emptyset \text{ , } falls\ i \neq j.$$

Beispiel 2.10. $\Omega = [a, b]$ *(Grossmann S.184)*
Wir definieren Gitterpunkte $\{x_i\}_{i=0}^N$ *über* $\bar{\Omega}$ *beschrieben wie folgt:*

$$a = x_0 < x_1 < x_2 < \cdots < x_{N-1} < x_N = b$$

und eine Zerlegung $\mathscr{Z}=\{\Omega_j\}_{j=1}^m$ *mit* $\Omega_i := (x_{i-1}, x_i)$ $i = 1, \ldots, N$.

Ferner sei $h_i := x_i - x_{i-1}, i = 1, \ldots, N$. Wir wählen lineare Ansatzfunktionen damit gilt $V_h = lin\{\phi_i\}_{i=0}^N$ wobei die Ansatzfunktionen definiert sind durch

$$\phi_i(x) = \begin{cases} \dfrac{1}{h_i}(x - x_{i-1}), \text{ für } x \in \Omega_i \\ \dfrac{1}{h_{i+1}}(x_{i+1} - x) \text{ für } x \in \Omega_{i+1} \\ 0, \text{ sonst} \end{cases} \tag{2.12}$$

Es gilt nach Konstruktion $\phi \in C(\bar{\Omega})$ sowie $\phi_i\Big|_{\Omega_j} \in C^1(\bar{\Omega}_j)$, somit hat man insgesamt $\phi_i \in H^1(\Omega)$. Die folgende Abbildung stellt die Graphen von Ansatzfunktionen ϕ_i dar.

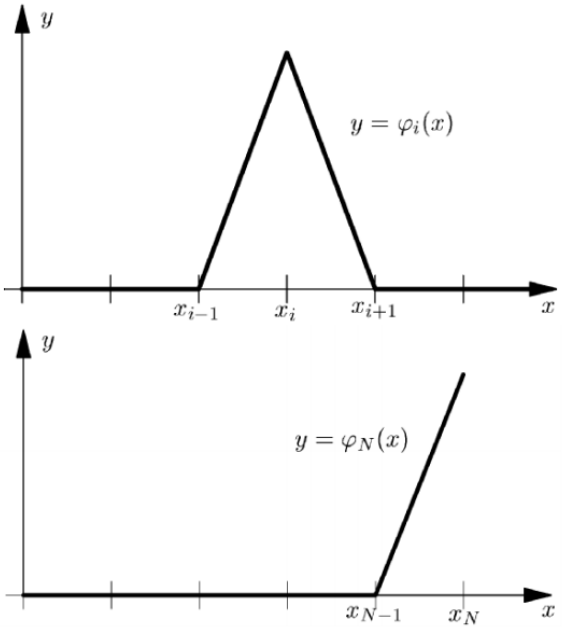

Abbildung 1: Ansatzfunktionen ϕ_i

Es gilt demnach $\phi_i(x_k) = \delta_{ik}, i, k = 0, 1, \ldots, N$.

Bemerkung 2.11. *Referenzzelle*

Satz 2.12. *Referenzzelle Transformation*

Definition 2.13. *Masse Matrix*

Definition 2.14. *Laplace Bilinearform*

Satz 2.15. *Quadratur*

Um nochmal auf die Gleichung von der Einleitung zurück zu kommen.

2.1.4 Diskontinuierliche Galerkin-Methode

Die diskontinuierliche Galerkin-Methode wurde 1973 erstmals von Reed und Hill eingeführt für hyperbolische Gleichung erster Ordnung. Es gab eine Reihe von Untersuchungen seither bezüglich hyperbolische Probleme erster Ordnung als auch für die Diskretisierung instationärer Probleme. Unabhängig davon wurde die diskontinuierliche Galerkin-Methode für elliptische Gleichungen vorgeschlagen. Man wollte mehr Flexibilität bezüglich der Stetigkeitsvoraussetzungen an unsere lokalen Funktionen und mehr Freiraum bei der Gitter Generierung, insbesondere bei adaptiven h-p-Methoden. Die Art von dGFEM-Code erleichtert Parallelisierbarkeit was in Zeiten von GPU Programmierung eine große Effizienzsteigerung erlaubt. Die Idee von dGFEM ist maßgeblich Strafterme einzuführen, welche Unstetigkeit zwar erlauben aber in ihrem Ausmaß einschränkt. Diese Freiheit erlaubt uns aber zum Beispiel lokal Polynome höherer Ordnung zu benutzen und Singularitäten damit gekonnt zu beseitigen, ohne von Vorne zu beginnen zu müssen. Zu mal pro Element ein hängender Knoten erlaubt ist. D.h. eine Verfeinerung des Gitters ist lokal erlaubt, ohne direkt Probleme zu bekommen. (S.292 Grossmann)

2.2 Tensor Dekomposition

2.2.1 Einführung in die Tensor Architektur

Definition 2.16. *Tensor*
Unter einem Tensor verstehen wir eine multidimensionale Matrix $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$
Die Ordnung ist die Anzahl der Dimensionen, in diesem Fall also N.

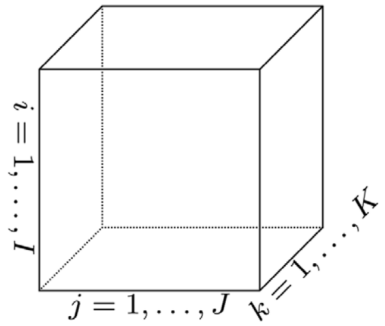

Abbildung 2: Tensor 3.Ordnung $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$

Definition 2.17. *Rang Eins Tensor*
Ein Tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ *ist von Rang Eins wenn es als äußeres Produkt von N Vektoren*

$$\mathcal{X} = a^{(1)} \circ \ldots a^{(N)}$$

geschrieben werden kann.

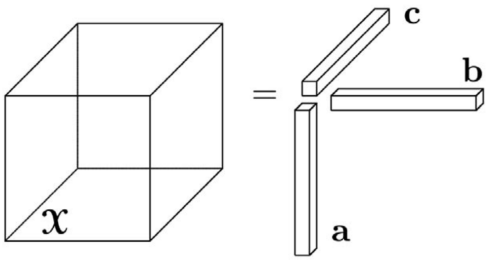

Abbildung 3: Rang Eins Tensor 3.Ordnung $\mathcal{X} = a \circ b \circ c$

Bemerkung 2.18. *Symmetrie*

- *Ein Tensor nennt man kubisch genau dann wenn jeder Mode dieselbe Dimension hat.*
- *Einen kubischen Tensor nennt man supersymmetrisch genau dann wenn die Elemente des Tensors konstant bleiben unter jeglicher Permutation der Indizes*
- *Ein Tensor kann stückweise symmetrisch sein wenn die Elemente konstant bleiben unter der Permutation von mindestens 2 Indizes.*

Definition 2.19. *Diagonal*
Einen Tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ nennt man diagonal, wenn $x_{i_1,\ldots,i_N} \neq 0$ genau dann wenn $i_1 = \cdots = i_N$.

Definition 2.20. *Faser*
Eine Faser ist das multidimensionale Analog zu Matrixspalten und Matrixzeilen. Wir definieren eine Faser, indem wir jeden Index abgesehen von einem festhalten.

Bemerkung 2.21. *Entfaltung*
Einen Tensor kann man entfalten. Dies impliziert eine neu ordnung der Tensorelemente in eine Matrix. Wir betrachten nur die sogenannte Ordnung-n Entfaltung, da dies die einzig relevante Form der Entfaltung ist. Eine Ordnung-n Entfaltung eines Tensors $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ wird mit $\mathbf{X}_{(n)}$ geschrieben und ordnet die Ordnung-n Fasern in die Spalten der Ergebnismatrix. Formal ist es eine Abbildung des Indize N-tupels $(i_1, \ldots, i_N)$ auf Matrixindizes (i_n, j)

$$j = 1 + \sum_{\substack{k=1 \\ k \neq n}}^{N} (i_k - 1) J_k \text{ mit } J_k = \prod_{\substack{m=1 \\ m \neq n}}^{k-1} I_m \tag{2.13}$$

Nun fehlt uns noch eine Tensor Multiplikation um mit der Dekomposition von Tensoren anzufangen.

Definition 2.22. *n-Ordnung Produkt*
Das n-Ordnung Produkt eiens Tensors $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ mit einer Matrix $U \in \mathbb{R}^{J \times I_n}$ wird geschrieben als $\mathcal{X} \times_n U$ und die Ergebnis Matrix hat die Größe $I_1 \times \ldots I_{n-1} \times J \times I_{n+1} \times \ldots I_N$

$$(\mathcal{X} \times_n U)_{i_1 \ldots i_{n-1} j i_{n+1} \ldots i_N} = \sum_{i_n=1}^{I_n} x_{i_1 \ldots i_N} u_{j i_n} \tag{2.14}$$

Bemerkung 2.23. *Jedes n-Ordnung Produkt kann mit Hilfe von entfaltenen Tensoren äquivalent ausgedrückt werden.*

$$\mathcal{Y} = \mathcal{X} \times_n U \Longleftrightarrow Y_{(n)} = UX_{(n)} \tag{2.15}$$

2.2.2 Singulärwertzerlegung höherer Ordnung

Die Higher Order Singular Value Decomposition oder auch bekannt unter Tucker Decomposition ist eine uminterpretierte multidimensionale Hauptkomponentenanalyse. Man versucht durch Hauptachsentransformationen die Korrelation zwischen den verschiedenen Merkmalen durch Überführung in eine neue Basis zu minimieren. Die Hauptachsentransformation lässt sich durch eine orthogonale Matrix angeben, die aus den Eigenvektoren der Kovarianzmatrix gebildet wird. (Wikipedia: Hauptkomponentenanalyse).

Die HOSVD zerlegt den Tensor in einen Kern-Tensor (Core Tensor) multipliziert mit einer Matrix in jeder Ordnung.

Beispiel 2.24. *HOSVD Tensor 3.Ordnung* $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$

$$\mathcal{X} \approx \mathcal{G} \times_1 A \times_2 B \times_3 C \tag{2.16}$$

Wobei $A \in \mathbb{R}^{I \times P}$, $B \in \mathbb{R}^{J \times Q}$ *und* $C \in \mathbb{R}^{K \times R}$ *die Faktormatrizen sind, welche orthogonal sind.* $\mathcal{G}$ *bezeichnet den Kern-Tensor und zeigt wie hoch die Korrelation zwischen den verschiedenen Komponenten ist.*

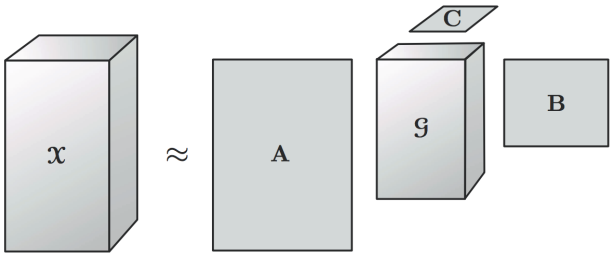

Abbildung 4: HOSVD eines Tensors 3.Ordnung

Bemerkung 2.25. *Entfaltete HOSVD*

$$X_{(n)} = A^{(n)} G_{(n)} (A^{(N)} \otimes \cdots \otimes A^{(n+1)} \otimes A^{(n-1)} \otimes \cdots \otimes A^{(1)})^T \tag{2.17}$$

Definition 2.26. *n-Rang*
$\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$. Der n-Rang eines Tensor wird geschrieben mit $rang_n(\mathcal{X})$ und ist der Spaltenrang des entfalteten Tensors $X_{(n)}$.

Bemerkung 2.27. *Sei $R_n = rang_n(\mathcal{X})$ für $n = 1, \ldots, N$. Dann können wir sagen, dass $\mathcal{X}$ ein $rang - (R_1, \ldots, R_N)$ Tensor ist.*

Bemerkung 2.28. *Berechnung der HOSVD (Wikipedia)*
Die Berechnung der HOSVD von $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ geht wie folgt

1. *Berechne die Ordnung-k Entfaltungen $\mathcal{X}_{(k)}$ für alle k*
2. *Berechne die SVD $\mathcal{X}_{(k)} = U_k \Sigma_k V_k^T$ und speichere U_k*
3. *Der Kerntensor $\mathcal{S}$ ergibt sich aus der Projektion des Tensors auf die Tensorbasis geformt von den Faktormatrizen $\{U_k\}_{k=1}^N$ also $\mathcal{S} = \mathcal{X} \times_{n=1}^N U_n^T$*

Bemerkung 2.29. *Eindeutigkeit der HOSVD*
Die HOSVD ist keine eindeutige Zerlegung. Dies führen wir an einem Tensor 3.Ordnung an. Sei $U \in \mathbb{R}^{P \times P}$, $V \in \mathbb{R}^{Q \times Q}$ und $W \in \mathbb{R}^{R \times R}$. Dann folgt

$$\mathcal{X} = \mathcal{G} \times_1 A \times_2 B \times_3 C = (\mathcal{G} \times_1 U \times_2 V \times_3 W) \times_1 AU^{-1} \times_2 BV^{-1} \times_3 CW^{-1} \quad (2.18)$$

In anderen Worten: Wir können den Kerntensor $\mathcal{G}$ modifizieren ohne die Gleichung zu ändern, solange wir das Inverse der Modifizierung auf den zugehörigen Faktormatrizen multiplizieren.

Mit diesen Kenntnissen können wir nun zum Beispiel versuchen soviele Elemente des Kerntensor wie möglich auf 0 zu bekommen oder so klein wie möglich zu machen. Weiterführende Liteartur S.479 TENSOR DECOMPOSITIONS AND APPLICATIONS.

Wir brauchen noch einige Eigenschaften zum Kronecker Produkt, die wir uns später für die Berechnung der Pseudoinversen zu Nutze machen wollen.

Lemma 2.30. *Sind A,B invertierbar so ist auch $(A \otimes B)$ invertierbar. Mit der Inversen*

$$(A \otimes B)^{-1} = A^{-1} \otimes B^{-1}$$

Für die Moore Penrose Pseudoinversen gilt:

$$(A \otimes B)^{+} = A^{+} \otimes B^{+}$$

3 Pseudoinverse für zell-basierte finiten Elemente Operatoren

3.1 Summenfaktorisierung

Im Exa-dg Teachlet wird die effektive Berechnung der Masse Matrix mit einem Vektor vorgestellt. Diese Methodik sollten wir uns kurz vor Augen führen und daraufaufbauend um einige eigene Gedanken erweitern um diese Methodik für unsere Anwendung zugänglich zu machen. Das Ziel ist es die Tensorprodukt Struktur für die Massematrix und den Laplace Bilinearform herzuleiten und für die Berechnung der Pseudoinversen zu nutzen.

Angenommen wir hätten eine Lösung $u \in V$ der Form

$$u(x,y) = \sum_{h=1}^{n^2} \psi_h(x,y) u_h,\ u_h \in \mathbb{R}\ \forall h = 1, \dots, n^2 =: N \tag{3.1}$$

mit ψ_h ist eine globale Ansatzfunktion. Diese Anstazfunktionen können als das Tensorprodukt von zwei ein dimensionalen Polynomen ausgedrückt werden. Die Überführung ist definiert mit

$$h = j(n-1) + i,\ i < n \leftrightarrow (i,j) \tag{3.2}$$

was uns eine alternative Representation von u gibt

$$u(x,y) = \sum_{i=1}^{n} \sum_{j=1}^{n} \varphi_i(x) \varphi_j(y) u_{ij},\ i,j = 1, \dots n \tag{3.3}$$

Seien $q = (q_1, \dots, q_n)^T$ Quadraturpunkte einer 1-dimensionalen Quadraturegel mit zugehörigen Gewichten $w = (w_1, \dots, w_n)^T$. Wir können die 2-dimensionale Regel erhalten indem wir wieder das Tensorprodukt ausnutzen und erhalten $(\boldsymbol{q}_1, \dots, \boldsymbol{q}_N)$ und Gewichten $(\boldsymbol{w}_1, \dots, \boldsymbol{w}_N)$ mit $\boldsymbol{q}_h = (q_i, q_j)$ and $\boldsymbol{w}_h = w_i w_j$.
Erstmals wollen wir u in jedem Quadraturpunkt evaluieren. Das können wir mit einem Matrix Vektor Produkt ausdrücken:

$$\begin{pmatrix} \psi_1(\boldsymbol{q}_1) & \dots & \psi_N(\boldsymbol{q}_1) \\ \vdots & \ddots & \vdots \\ \psi_1(\boldsymbol{q}_N) & \dots & \psi_N(\boldsymbol{q}_N) \end{pmatrix} \begin{pmatrix} u_1 \\ \vdots \\ u_N \end{pmatrix} = \begin{pmatrix} u(\boldsymbol{q}_1) \\ \vdots \\ u(\boldsymbol{q}_N) \end{pmatrix} \tag{3.4}$$

Die Matrix kann als Tensorprodukt zweier, in diesem Fall sogar identischer, Matrizen:

$$\mathcal{N}^T \otimes \mathcal{N}^T = \begin{pmatrix} \varphi_1(q_1) & \dots & \varphi_n(q_1) \\ \vdots & \ddots & \vdots \\ \varphi_1(q_n) & \dots & \varphi_n(q_n) \end{pmatrix} \otimes \begin{pmatrix} \varphi_1(q_1) & \dots & \varphi_n(q_1) \\ \vdots & \ddots & \vdots \\ \varphi_1(q_n) & \dots & \varphi_n(q_n) \end{pmatrix} \tag{3.5}$$

Sei $(\mathcal{U}_{ij}) \in \mathbb{R}^{n \times n}$ die Matrix mit den Koeffizienten u_{ij} als Elemente. Durch die Nutzung der Summenfaktorisierung können wir diese Matrizen multiplizieren anstatt der Formel in (4). Wir erhalten:

$$\mathcal{N}^T \mathcal{U} \mathcal{N} = \bar{\mathcal{U}}, \quad \bar{\mathcal{U}}_{ij} = u(q_i, q_j) \tag{3.6}$$

Die Quadraturgewichte und die Summe über die Elemente kann in einer effiziente Weise vollzogen werden, wenn wir die Tensorprodukt Struktur ausnutzen. Die Multiplizierung $\bar{\mathcal{U}}$ mit den Quadraturgewichten w von der rechten Seite wird uns die gewichteten Spalten aufsummieren, , multipliziert mit w^T wird den Rest erledigen. Die ganze Operation sieht wie folgt aus:

$$\int \int u(x,y) dx \; dy \approx w^T \mathcal{N}^T \mathcal{U} \mathcal{N} w \tag{3.7}$$

Wir haben nun u integriert. Jedoch können wir nun mit Ansatzfunktionen testen ohne viel zu verändern. Wir schauen uns $f(x,y) = g(x)h(y)$ an. Also ist f seperabal und somit müssen wir nur den letzten Schritt ändern. Anstatt nur den Gewichten zu multiplizieren, werden wir auch mit den Ansatzfunktionen, an den bestimmten Quadraturpunkten evaluiert, multiplizieren:

$$\bar{w}^x = (w_1 g(q_1), \dots, w_n g(q_n))^T \qquad \bar{w}^y = (w_1 h(q_1), \dots, w_n h(q_n))^T$$
$$\int \int u(x,y) f(x,y) dx \; dy \approx (\bar{w}^x)^T \mathcal{N}^T \mathcal{U} \mathcal{N} \bar{w}^y \tag{3.8}$$

Wenn wir einen Menge von Ansatzfunktionen haben, welche hergeleitet sind von dem Tensorprodukt von identischen 1-dimensionalen Ansatzfunktionen, können wir die Matrizen multiplizieren anstatt die Vektoren im letzten Schritt, was nun die Operation vollständig macht. Wir definieren:

$$W = \begin{pmatrix} w1\varphi_1(q_1) & \dots & w_n\varphi_1(q_n) \\ \vdots & \ddots & \vdots \\ w1\varphi_n(q_1) & \dots & w_n\varphi_n(q_n) \end{pmatrix}$$

Da $\mathcal{W}\mathcal{N}^T$ auf Beiden Seiten auftaucht, können wir diese Operationr mit 3 Matrixmultiplikationen ausdrücken. Finale Form der Gleichung:

$$\mathcal{V} = \mathcal{W}\mathcal{N}^T\mathcal{U}(\mathcal{W}\mathcal{N}^T)^T \tag{3.9}$$

Die Formel leicht geändert kann benutzt werden um andere Bilinearformen auszudrücken:
$(\nabla u, \nabla v)$.

$$(\nabla u, \nabla v) = (\partial_x u, \partial_x v) + (\partial_y u, \partial_y v) \tag{3.10}$$
$$= \sum_{i,j} u_{ij}(\varphi_i'(x)\varphi_j(y), \phi'(x)\phi(y)) \tag{3.11}$$
$$+ \sum_{i,j} u_{ij}(\varphi_i(x)\varphi_j'(y), \phi(x)\phi'(y)) \tag{3.12}$$

Nun werden wir 2 Matrizen $\mathcal{W}'$ und $\mathcal{N}'$ vorstellen. Diese Matrizen sind ähnlich zu den oben, aber anstatt die Ansatzfunktionen zu evaluieren, evaluieren sie die Ableitung der Ansatzfunktionen. Die resultierende Form ergibt sich aus:

$$\mathcal{V} = \mathcal{W}'\mathcal{N}'^T\mathcal{U}(\mathcal{W}\mathcal{N}^T)^T + \mathcal{W}\mathcal{N}^T\mathcal{U}(\mathcal{W}'\mathcal{N}'^T)^T$$

Sei $(-\Delta u, v) = -(\partial_{xx}u, v) + -(\partial_{yy}u, v)$.
Wir führen $\mathcal{N}''$ ein, was die Werte der 2.Ableitung der Ansatzfunktionen in den Quadraturgewichten speichert. Bemerke, dass die Matrix $\mathcal{W}$ nicht modifziert wurde. Als Resultat erhalten wir

$$\mathcal{V} = -\mathcal{W}\mathcal{N}''^T\mathcal{U}(\mathcal{W}\mathcal{N}^T)^T - \mathcal{W}\mathcal{N}^T\mathcal{U}(\mathcal{W}\mathcal{N}''^T)^T$$

Zusammengefasst erhalten wir für die Masse Matrix die Form

$$M * u = (\mathcal{W}\mathcal{N}^T) \otimes (\mathcal{W}\mathcal{N}^T) * u \tag{3.13}$$

und für die Laplace Bilinearform

$$\mathcal{V} * u = -((\mathcal{W}\mathcal{N}''^T) \otimes (\mathcal{W}\mathcal{N}^T) - (\mathcal{W}\mathcal{N}^T) \otimes (\mathcal{W}\mathcal{N}''^T)) * u \tag{3.14}$$

DIe Berechnung der Pseudoinversen passiert mit diesem Ansatz quasi on the fly, da wir hier gleich als Ergebnis nicht die Pseudoinverse bekommen, sondern das Matrix

Vektor Produkt mit der Pseudoinversen als Matrix.

Für die effiziente Berechnung der Inversen brauchen wir folgendes Ergebnis:

Bemerkung 3.1.

$$(A \otimes B)^{-1} = A^{-1} \otimes B^{-1} \tag{3.15}$$

Nun erhalten wir fast direkt unser gewünschtes Ergebnis

$$M^{-1} * u = ((\mathcal{W}\mathcal{N}^T) \otimes (\mathcal{W}\mathcal{N}^T))^{-1} * u = ((\mathcal{W}\mathcal{N}^T)^{-1} \otimes (\mathcal{W}\mathcal{N}^T)^{-1}) * u \tag{3.16}$$

$\mathcal{W}\mathcal{N}^T \in \mathbb{R}^{n \times n}$ somit also quadratisch. Das heißt um das Pseudoinverse-Vektor Produkt zu berechnen müssen wir nur eine SVD von einer $n \times n$ Matrix berechnen. Falls uns das zu teuer wird können wir auch eine truncated SVD benutzen. In welchem Komplexitätsbereich wir uns befinden, wird im nächsten Subkapitel angeführt.

Für die Laplace Bilinearform ähnlich:

$$\mathcal{V}^{-1} * u = -((\mathcal{W}\mathcal{N}''^T) \otimes (\mathcal{W}\mathcal{N}^T) - (\mathcal{W}\mathcal{N}^T) \otimes (\mathcal{W}\mathcal{N}''^T))^{-1} * u = -((\mathcal{W}\mathcal{N}''^T)^{-1} \otimes (\mathcal{W}\mathcal{N}^T)^{-1} - (\mathcal{W}\mathcal{N}^T)^{-1} \otimes (\tag{3.17}$$

Praktisch müssen wir also nur die Inversen von $(\mathcal{W}\mathcal{N}''^T)$ und von $(\mathcal{W}\mathcal{N}^T)$ berechnen.

Nun war dies der 2 Dimensionale Fall. Die Erweiterung auf den 3 Dimensionalen Fall ist analog und kann im "Efficient Evaluation ..... Teachlet" angeschaut werden.

Zu der Implementierungsstrategie kann man in Kapitel 4 mehr erfahren.

3.2 Singulärwertzerlegung höherer Ordnung

Wir wollen nun mit Hilfe von der Theorie zur Singulärwertzerlegung höherer Ordnung eine Theorie entwickeln, wie wir die Pseudoinverse zur Masse Matrix und zur globalen Steifigkeitsmatrix der Laplace Bilinearform effiizient berechnen können.

Allgemein wissen wir:

$$\mathcal{A} = \mathcal{S} \times_{n=1}^{N} U^{(n)} \tag{3.18}$$

D.h. wir können einen Tensor $\mathcal{A}$ in einen Kerntensor $\mathcal{S}$ und zugehörige Faktormatrizen U zerlegen. Wie bekommen wir nun die Pseudoinverse zu A? Was bedeutet in dem Kontext eines Tensors überhaupt Pseudoinverse?

Wir kennen noch die Eigenschaften der Moore Penrose Pseudoinverse für Matrizen:

Lemma 3.2. *Moore Penrose Pseudoinverse (Wikipedia)*

- $AA^+A = A$
- $A^+AA^+ = A^+$ *(A^+ ist eine schwache Inverse der multiplikativen Halbgruppe)*
- $(AA^+)^* = AA^+$ *(AA^+ ist hermitisch)*
- $(A^+A)^* = A^+A$ *(A^+A ist hermitisch)*

Jetzt gilt es diese Eigenschaften für Tensoren zu übertragen. Da wir erstmal keine intuitive Tensor Tensor Multiplikation haben, gilt es diese zu definieren. Diese Tensor-Tensor Multiplikation macht dann nur für unsere Anwendung einen Sinn und ist sonst zweckfrei.

Definition 3.3. *Tensor-Tensor Multiplikation* $ttp(\dot{;})$

Nun ist kommt auch ein weiterer Trick den wir nutzen können. Trick kann nur mit Vorsicht genossen werden. Die Gleichheit ist mit einem Rechner nicht zu erzielen, daher wird die Bemerkung abgeschwächt und für Tensoren. Wir erhalten:

Lemma 3.4. *Moore Penrose Pseudoinverse für Tensoren*

- $ttp(A, ttp(A^+, A)) - A < \epsilon$
- $ttp(A^+, ttp(A, A^+)) - (A^+ < \epsilon$ *(A^+ ist eine schwache Inverse der multiplikativen Halbgruppe)*
- $(ttp(A, A^+))^* - ttp(A, A^+) < \epsilon$ *(AA^+ ist hermitisch)*
- $(ttp(A^+, A))^* - ttp(A^+, A) < \epsilon$ *(A^+A ist hermitisch)*

Die Wahl des Epsilons ist hier entscheidend. Man könnte Maschinengenauigkeit wählen, doch ist für unser Zweck vielleicht zu Hoch gezielt. Letztlich wollen wir mit unserer Pseudoinversen einen Präkonditionierer bauen. Wenn wir durch die Wahl eines etwas größeren Epsilons erheblichen Aufwand sparen, sollten wir dies in Erwägunge ziehen. Nun wissen wir, wie wir einen Tensor als Pseudoinverse klassifizieren können. Doch wie bekommen wir die Pseudoinverse?

Die Pseudoinverse setzt sich zusammen aus:

$$\mathcal{A}^\dagger = \mathcal{S}^\dagger \times_{n=1}^{N} U^{(n)^T} \tag{3.19}$$

Da die Faktormatrizen orthogonal sind, reicht es also einfach die Transponierte zu nehmen. Das Invertieren des Kerntensors erweist sich nun aber als problematisch.

Hier ist es nützlich die Struktur des Kerntensors zu kennen. Der Kerntensor ist leider in den meisten Fälle vollbesetzt. Doch genaueres Hinsehen zeigt 2 Arten von Zahlen. Ziemlich große Zahlen von größer als 1 und ziemlich kleine Zahlen von kleiner als 10^{-10}. Die kleinen Zahlen sind in diesem Fall unbrauchbar und beinhalten wenig Informationen. Doch das Auslöschen vieler kleiner Zahlen nimmt uns in der Summe vielleicht relevante Informationen. Wir können also kleine Zahlen einfach ausradieren und erhalten plötzlich einen super-diagonalen Tensor. Die Invertierung des Tensors beschränkt sich darauf einfach jedes Diagonalelement zu Invertieren.

Als nächstes wollen wir klären wie wir überhaupt von Massen Matrix Form und Laplace Bilinearform zu einer Tensorform kommen. Die Masse Matrix M besteht aus Einträgen $M_{ij} = (\phi_i, \phi_j)$, wobei $\dot{()}$ die zugehörige Bilinearform ist und ϕ_k eine 2 dimensionale Ansatzfunktion. Dies kann man vereinfachen mit $M_{ij} = (\phi_i, \phi_j) = (\phi_{i_1}\phi_{i_2}, \phi_{j_1}\phi j_2) = M^*_{i_1 i_2 j_1 j_2}$, wobei wir nun 1 dimensionale Ansatzfunktionen haben. M^* ist ein Tensor 4.Ordnung, den wir nun mit Hilfe der Singulärwertzerlegung höherer Ordnung zerlegen können. Für die Laplace Bilinearform ist es ähnlich, denn wir haben in schwacher Form $L_{ij} = (\nabla\phi_i, \nabla\phi_j) = (\nabla\phi_{i_1}\nabla\phi_{i_2}, \nabla\phi_{j_1}\nabla\phi j_2) = L^*_{i1i2j1j2}$

Wir wissen nun wie wir unsere Tensoren berechnen können und wissen auch wie die Pseudoinverse sich gewinnen lässt mittles der Singulärwertzerlegung höherer Ordnung. Der nächste Punkt ist die effiziente Berechnung der Pseudoinversen.

4 Effiziente Implementierung

In Kapitel 3 haben wir uns zwei Möglichkeiten angeschaut, das Matrix-Vektor Produt zu berechnen mit der Pseudoinversen. Beide Alternativen bargen eine Tensorprodukt Struktur, in Form des Matrix-Kronecker Produkt.

Das heißt um dies effizient zu implementieren sollten wir uns Gedanken dadrüber machen wie wir diese Struktur ausnutzen können.

Im Exa-dg-teachlet wird eine Strategie vorgestellt ein Matrix-vektor Produkt mit Kronecker Matrizen also $z = (\mathcal{B} \otimes \mathcal{A})y$ berechnet werden kann.

Sei $\mathcal{A} \in \mathbb{R}^{m \times n}$ und $\mathcal{B} \in \mathbb{R}^{p \times q}$. Das Kronecker Produkt dieser Matrizen kann man schreiben als,

$$\mathcal{B} \times \mathcal{A} = \begin{pmatrix} b_{11}\mathcal{A} & \dots & b_{1q}\mathcal{A} \\ \vdots & \ddots & \vdots \\ b_{p1}\mathcal{A} & \dots & b_{pq}\mathcal{A} \end{pmatrix}$$

Nun wir sehen also die sich wiederholende Struktur von $\mathcal{A}$. Genau diese wollen wir uns nun zu nutze machen. Nehmen wir an y sei geordnet in der Indexierung.

$$y = (y_1, y_2, \dots, y_n, \dots, \dots, y_{(q-1)n+1}, y_{(q-1)n+2}, \dots, y_{qn})^T$$

Wir denken uns nun die Faktoren b_{ij} die mit $\mathcal{A}$ multipliziert werden erstmal weg. Definiere $y^{(1)} = (y_1, y_2, \dots, y_n)^T$.

$$w^{(1)} = \begin{pmatrix} w_1 \\ \vdots \\ w_m \end{pmatrix} = \begin{pmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \dots & \vdots \\ a_{m1} & \dots & a_{mn} \end{pmatrix} \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} = \mathcal{A}y^{(1)}$$

Auf ähnliche Weise können wir uns $y^{(2)} = (y_{n+1}, \dots, y_{2n})^T$ definieren und dann

$$w^{(2)} = \mathcal{A}y^{(2)}$$

Wir führen dies so weiter und erhalten

$$w = ((w^{(1)})^T, \dots, (w^{(q)})^T) \in \mathbb{R}^{mq}$$

Nun müssen wir die Informationen der Matrix B noch mit reinbringen. Dazu berechnen wir wie im Teachlet auch vorgeschlagen z_i mit

$$z_i = \sum_{i=1}^{q} b_{1i} w_m^{(i)}$$

Mit Hilfe diesen Algorithmus haben wir die Komplexität von $2m^4$ auf $4m^3$ reduziert. Nun wollen wir dies erweitern auf $z = (\mathcal{C} \otimes \mathcal{B} \otimes \mathcal{A})v$.

$$z := \begin{pmatrix} c_{11}b_{11}A & \dots & c_{11}b_{1n}A & \dots & \dots & c_{1n}b_{11}A & \dots & c_{1n}b_{1n}A \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \vdots \\ c_{11}b_{n1}A & \dots & c_{11}b_{nn}A & \dots & \dots & c_{1n}b_{n1}A & \dots & c_{1n}b_{nn}A \\ c_{21}b_{n1}A & \dots & c_{21}b_{nn}A & \dots & \dots & c_{2n}b_{n1}A & \dots & c_{2n}b_{nn}A \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \vdots \\ c_{n1}b_{n1}A & \dots & c_{n1}b_{nn}A & \dots & \dots & c_{nn}b_{n1}A & \dots & c_{nn}b_{nn}A \end{pmatrix} * v \tag{4.1}$$

Wir sehen hier sich zwei wiederholende Strukturen die wir ausnutzen können um bei einem Matrix-Vektor Produkt operationen zu sparen.

$$z = \begin{pmatrix} c_{11}b_{11}A & \dots & c_{11}b_{1n}A & \dots & \dots & c_{1n}b_{11}A & \dots & c_{1n}b_{1n}A \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \vdots \\ c_{11}b_{n1}A & \dots & c_{11}b_{nn}A & \dots & \dots & c_{1n}b_{n1}A & \dots & c_{1n}b_{nn}A \\ c_{21}b_{n1}A & \dots & c_{21}b_{nn}A & \dots & \dots & c_{2n}b_{n1}A & \dots & c_{2n}b_{nn}A \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \vdots \\ c_{n1}b_{n1}A & \dots & c_{n1}b_{nn}A & \dots & \dots & c_{nn}b_{n1}A & \dots & c_{nn}b_{nn}A \end{pmatrix} * v \tag{4.2}$$

Nun wollen wir uns dies zu nutze machen. Wir schauen uns erstmal die einzelnen Einträge von z an und bekommen. Vorher definieren wir unser v um zu einem Tensor $\mathcal{V} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$. Der erste Index repräsentiert in welcher Spalteneintrag von C wir uns befinden, der zweite in welchem Spalteneintrag von B und der Dritte in welchem Spalteneintrag von A.

$$\begin{aligned} z_1 = \mathcal{V}(1,1,1)c_{11}b_{11}a_{11} + \dots + \mathcal{V}(1,1,n)c_{11}b_{11}a_{1n} + \dots + \mathcal{V}(1,n,1)c_{11}b_{1n}a_{11} \\ + \dots + \mathcal{V}(n,1,1)c_{1n}b_{11}a_{11} + \dots + \mathcal{V}(n,n,n)c_{1n}b_{1n}a_{1n} \end{aligned} \tag{4.3}$$

Definiere $w_1(i,j) := \mathcal{V}(i,j,1)a_{11} + \dots + \mathcal{V}(i,j,n)a_{1n}$. Dann erhalten wir:

$$z_1 = w_1(1,1)c_{11}b_{11} + \dots + w_1(1,n)c_{11}b_{1n} + \dots + w_1(n,1)c_{1n}b_{11} + \dots + w_1(n,n)c_{1n}b_{1n}$$

Damit haben wir uns die sich wiederholende Struktur von der Matrix A zu nutze gemacht. Im nächsten Schritt machen wir uns die sich wiederholende Struktur von b_{ij} zu nutze. Definiere hierfür $\mathfrak{W}_{1,k}(i) := w_k(i,1)b_{11} + \cdots + w_k(i,n)b_{1n}$. Damit erhalten wir:

$$z_1 = \mathfrak{W}_{1,1}(1)c_{11} + \cdots + \mathfrak{W}_{1,1}(n)c_{1n} \tag{4.4}$$

Wir wollen nun z genau so umformen wie wir das auch für v gemacht haben. Damit erhalten wir für allgemeines z_i folgende Formel:

$$\mathcal{Z}(i,j,k) = \mathfrak{W}_{j,k}(1)c_{i1} + \cdots + \mathfrak{W}_{j,k}(n)c_{in} \tag{4.5}$$

Wobei j und k den Zeilen jeweils in den Matrizen B und C entsprechen.

Der komplette Algorithmus würde nun wie folgt aussehen:

for k=1 < n **do**
  for i= 1 < n **do**
    for j= 1 < n **do**
      $w_k(i,j) = \mathcal{V}(i,j,1)a_{k1} + \cdots + \mathcal{V}(i,j,n)a_{kn}$
    end for
  end for
end for
for k=1 < n **do**
  for i= 1 < n **do**
    for j= 1 < n **do**
      $\mathfrak{W}_{i,j}(k) := w_k(i,1)b_{11} + \cdots + w_k(i,n)b_{1n}$
    end for
  end for
end for
for k=1 < n **do**
  for i= 1 < n **do**
    for j= 1 < n **do**
      $\mathcal{Z}(i,j,k) = \mathfrak{W}_{j,k}(1)c_{i1} + \cdots + \mathfrak{W}_{j,k}(n)c_{in}$
    end for
  end for
end for

Wenn wir annehmen, dass die Matrizen $A, B, C \in \mathbb{R}^{n\times n}$. Dann haben wir bei **??**

eine Matrix-Vektor Multiplikation von einer Matrix der Größe $n^3 \times n^3$. Dementsprechend hätten wir n^6 Multiplikationen und n^6 Additionen. Die Komplexität des vorgeschlagenen Algorithmuses reduziert sich auf $3n^3$ Multiplikationen und genau so viele Additionen. Ein enorme Reduktion, vor allem für großes n.

Wie können wir uns nun diese Algorithmen zu nutze machen für die in Kapitel 3 besprochenen Strategien?

Summenfaktorisierung

Rekapituliere die Formeln für die Masse Matrix und für die Laplace Bilinearform
Masse Matrix

$$M^{-1} * u = ((\mathcal{W}\mathcal{N}^T) \otimes (\mathcal{W}\mathcal{N}^T))^{-1} * u = ((\mathcal{W}\mathcal{N}^T)^{-1} \otimes (\mathcal{W}\mathcal{N}^T)^{-1}) * u$$

Laplace Bilinearform

$$\mathcal{V}^{-1} * u = -((\mathcal{W}\mathcal{N}''^T)^{-1} \otimes (\mathcal{W}\mathcal{N}^T)^{-1} - (\mathcal{W}\mathcal{N}^T)^{-1} \otimes (\mathcal{W}\mathcal{N}''^T)^{-1}) * u$$

Um dies effizient zu berechnen nutzen wir einfach die Formeln aus dem Teachlet und erhalten eine Komplexität von: Matrizenmultiplikation ist kubisch also n^3. Das Kroneckerprodukt von 2 Matrizen ist wie wir bereits wissen $4n^3$. Nun müssen wir uns um die Invertierung und deren Komplexität kümmern. Nach einer normalen Singulärwertzerlegung bekommen wir $O(min(mn^2, m^2n))$. Nun müssen wir wieder die Inverse daraus herleiten. Oder wir können den Gauß Algorithmus oder mit der Neumann Reihe arbeiten. Gauß Algorithmus gibt uns eine Komplexität von $O(n^3)$. Neumann Reihe macht keinen Sinn, da die Komplexität mindestens genau so hoch ist wie bei Gauß. Insgesamt haben wir eine Komplexität mit Gauß: $O(n^3) + O(n^3) + O(4n^3) = O(6n^3) \in O(n^3)$. Das heißt wir sind immer noch kubisch, was ein großer Vorteil ist. Die Singulärwertzerlegung bringt uns erst einen Vorteil, wenn wir eine trunkierte Zerlegung wählen. Die mögliche Ersparnis ist aber nicht so hoch. **BEGRÜNDUNG**.

Singulärwertzerlegung höherer Ordnung Nun haben wir uns mit Hilfe der Tucker Dekomposition eine Herleitung für die Pseudoinverse erarbeitet. Nun geht es um die effiziente Berechnung dieser Formel. Dazu wollen wir uns die Summenfaktorisierung zu nutze machen, wie sie auch in [**?**, 9-11] vorgeschlagen wird. Sei $\mathcal{A} \in \mathbb{R}^{I_1 \times \cdots \times I_n}$. Die Formel für die Pseudoinverse lautet:

$$\mathcal{A}^{\dagger} = \mathcal{S}^{\dagger} \times_{n=1}^{N} U^{(n)^T} \tag{4.6}$$

Wobei $\mathcal{S} \in \mathbb{R}^{I_1 \times \cdots \times I_n}$ und $U^{(n)} \in \mathbb{R}^{J_n \times I_n}$ Man kann **??** nach [**?**, 462] äquivalent umformen zu

$$\begin{aligned} & \mathcal{A}^{\dagger}_{(n)} = U^{(n)^T} \mathcal{S}^{\dagger}_{(n)} (U^{(N)^T} \otimes \cdots \otimes U^{(n+1)^T} \otimes U^{(n-1)^T} \otimes \cdots \otimes U^{(1)^T})^T \\ \iff & \mathcal{A}^{\dagger}_{(n)} = U^{(n)^T} \mathcal{S}^{\dagger}_{(n)} (U^{(N)} \otimes \cdots \otimes U^{(n+1)} \otimes U^{(n-1)} \otimes \cdots \otimes U^{(1)}) \end{aligned} \tag{4.7}$$

Nun betrachten wir uns das Matrix-Vektor Produkt und überlegen uns wie wir uns die Strukturen dort zu nutze machen.

$$\mathcal{A}^{\dagger}_{(n)} v = U^{(n)^T} \mathcal{S}^{\dagger}_{(n)} (U^{(N)} \otimes \cdots \otimes U^{(n+1)} \otimes U^{(n-1)} \otimes \cdots \otimes U^{(1)}) v \tag{4.8}$$

Wir schauen uns die Struktur mal für den Fall, dass $\mathcal{A} \in \mathbb{R}^{I_1 \times \cdots \times I_4}$. Das heißt **??** reduziert sich auf:

$$\mathcal{A}^{\dagger}_{(n)} v = U^{(n)^T} \mathcal{S}^{\dagger}_{(n)} (U^{(N_1)} \otimes U^{(N_2)} \otimes U^{(N_3)}) v \tag{4.9}$$

mit $N_i \neq n$.

Wir erkennen auf der Rechten seite können wir unsere Methodik, die wir entwickelt haben, um dieses doppelte Kronecker Produkt mit dem Vektor effizient zu berechnen. Wir können dies mit einer Komplexität von $O(6n^3)$ machen. Nun geht es darum den Kerntensor zu invertieren. Wie wir wissen müssen wir dazu erstmal kleine Zahlen, die wir Rauschen nennen, auf 0 setzen und die Diagonalelemente invertieren. Dazu müssen wir, da der Kerntensor vollbesetzt ist, durch alle Elemente des Tensors durchiterieren! Das heißt wir haben eine Komplexität von $O(n^4)$, da unser Tensor Ordnung 4 hat. Nun folgt eine letzte Matrix Matrix Multiplikation, aber von Matrizen der Größe $n^2 \times n^2$. Das heißt eine Komplexität von $O(n^6)$ was uns die ganze Komplexität mit raketengeschwindigkeit zerbombt. Doch dank der Struktur des Kerntensors reduziert sich die Komplexität auf $O(n^4)$. Insgesamt haben wir eine Komplexität von $O(6n^3) + 2O(n^4) = O(6n^3 + 2n^4)$. Dies ist aber nicht annähernd so gut wie Option 1. Wie können wir unsere effizienz weiter steigern? Wir können uns an die trunkierte HOSVD ranmachen oder eine andere Alternative finden. Nun anstatt den Kerntensor zu modifizieren, können wir bei der Matrix-Matrix Multiplikation einfach nur die Diagonalelemente für die Multiplikation in Erwägung ziehen. Damit sparen wir uns schon $O(n^4)$ Operationen. Was wir tun könnten ist den Kerntensor in

das Kronecker Produkt reinmultiplizieren bevor wir dieses ausrechnen. Selbst wenn wir das hinkriegen würden wir es nicht hinbekommen die Komplexität von Option 1 zu übertreffen. Außerdem ist das Problem, dass wir nicht mit der Pseudoinverse eine Matrix-Vektor Multiplikation durchführen, sondern mit dem entfalteten Tensor, der die Pseudoinverse darstellt. Doch wie dieser Tensor entfaltet wird, ist für die korrekte Berechnung von enormer Wichtigkeit. D.h. selbst wenn wir es schaffen dies effizient zu berechnen, ist das Ergebnis das was wir wollen? Die Antwort ist: Es kommt drauf an. Worauf? Wie wir unser v aufsetzen. Unser v könnte so aufgebaut werden, dass dies in Übereinstimmung mit den Elementen des entfalteten Tensors übereinstimmt. Dadurch würden wir genau das erreichen was wir erreichen wollten. Doch wie gesagt, macht uns die Komplexität einen gewaltigen Schnitt durch die Rechnung.

Eine letzte Alternative zeigt wäre eben die trunkierte HOSVD. Wir hauen eine Dimension raus und die korrespondierenden Singulärwerte raus. Am Meisten macht es Sinn natürlich die kleinsten Singulärwerte rauszuhauen.

Wir erhalten:

$$\mathbf{A}^{\dagger}_{(n)} v = U^{(n)^T} \mathbf{S}^{\dagger}_{(n)} (U^{(N_1)} \otimes U^{(N_2)}) v \tag{4.10}$$

DIe Komplexität dies auszurechnen beträgt für das Kronecker-Matrix Vektor Produkt $O(4n^3)$. Nun folgt ein weiterer Trick. Wir nutzen folgende Bemerkung:

Bemerkung 4.1. *Sei A eine Diagonalmatrix mit $A \in \mathbb{R}^{n^3 \times n^3}$ und $y \in \mathbb{R}^{n^3}$. Dann gilt:*

$$A(\mathcal{B} \otimes \mathcal{C})y = (\mathcal{B} \otimes \mathcal{C}) \begin{pmatrix} a_{11}y_1 \\ \vdots \\ a_{n^3 n^3} y_n \end{pmatrix}$$

Um diese Transformation zu berechnen brauchen wir $O(n^3)$ Operationen. Insgesamt erhalten wir$O(5n^3)$. Nun müssen wir noch das letzte Matrix-Matrix Produkt berechnen, was uns wieder alles kaputt macht.

Wenn wir noch eine Dimension rausstreichen enden wir bei:

$$\mathbf{A}^{\dagger}_{(n)} v = U^{(n)^T} \mathbf{S}^{\dagger}_{(n)} (U^{(N_1)}) v \tag{4.11}$$

5 Resultate

Literatur

[Joh08] Claes Johnson. *Numerical Solution of Partial Differential Equations by the Finite Element Method*. Dover Publications, 2008.

[MK12] Katharina Kormann Martin Kronbichler. *A generic interface for parallel cell-based finite element operator application*. Elsevier, 2012.

[Tea] *Efficient evaluation of weak forms in discontinuous Galerkin methods*.

[TK09] Brett Bader Tamara Kolda. *Tensor Decompositions and Applications*. SIAM, 2009.