Tensorstruktur der Zellmatrizen bei finiten Elementen

Bachelorarbeit

eingereicht von

Enes Witwit

betreut von

Prof. Dr. Kanschat

Fakultät für Mathematik und Informatik

Ruprecht-Karls-Universität Heidelberg

5. Juni 2017

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.

Heidelberg, 5. Juni 2017 Unterschrift

Abstract

The following work attempts to derive efficient algorithms and approaches to compute the matrix vector product with the pseudoinverse of specific cellmatrices in finite elements. The cellmatrices that we take into account are from the mass matrix and the laplace bilinearform. We derive two approaches to compute the pseudoinverse. The first one uses the tensor structure of the cellmatrices to calculate the pseudoinverse and an efficient algorithm to compute the matrix vector product. The second one uses a higher order singular value decomposition to compute the pseudoinverse. The first approach is highly efficient since it has the complexity class $O(N^3)$, where N is the polynomial order. But it lacks unfortunatley of flexibility. The second approach is very flexible but it has a high complexity order of $O(N^5)$.

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 Tensorstruktur der genannten Matrizen, um eine einfache Berechnung der Pseudoinversen zu ermöglichen. Mit der zweiten Methode erreichen wir für die Berechnung der Pseudoinversen und dem Matrix-Vektor-Produkt eine geringe Komplexität von $O(N^3)$, wobei N der Polynomgrad ist. Hingegen stoßen wir beim ersten Ansatz auf diverse Probleme bezüglich der einfachen Berechnung des Produktes und der hohen Komplexität von $O(N^5)$. Der zweite, effizientere Ansatz hat jedoch das Problem, dass er individuell für jede Bilinearform hergeleitet werden muss, während der erste Ansatz universell für alle Bilinearformen gültig ist.

Inhaltsverzeichnis

Notation **I**

Abbildungsverzeichnis **II**

1 Einführung **1**

2 Theorie **3**

2.1 Numerische Behandlung partieller Differentialgleichungen 3

2.1.1 Schwache Lösungen 3

2.1.2 Galerkin Verfahren 7

2.1.3 Methode der finiten Elemente 9

2.1.4 Quadratur 13

2.2 Tensor Dekomposition 15

2.2.1 Einführung in die Tensor Architektur 15

2.2.2 Singulärwertzerlegung höherer Ordnung 16

3 Pseudoinversen der Zellmatrizen **19**

3.1 Summenfaktorisierung 19

3.1.1 Tensorstruktur der Elementmassenmatrix 19

3.1.2 Tensorstruktur der Elementsteifigkeitsmatrix der Laplace Bilinearform 21

3.1.3 Pseudoinverse 23

3.2 Singulärwertzerlegung höherer Ordnung 24

4 Effiziente Implementierung und Komplexitätsanalyse **30**

4.1 Effizientes Matrix-Vektor Produkt 30

4.1.1 Erweiterung 32

4.2 Tensorstruktur 35

4.3 Singulärwertzerlegung höherer Ordnung 38

4.3.1 Naiver Ansatz 39

4.3.2 Entfaltete HOSVD 40

5 Resultate **43**

Literatur **44**

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
$(\cdot,\cdot)_V$	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
$\partial\Omega$	Rand von Ω
$int(\Omega)$	Inneres von Ω
$C^l(\Omega))$	Räume stetig-differenzierbarer 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
$\mathbf{A}^{(n)}$	bezeichnet die n-te Matrix einer Matrixfolge
N	Anzahl der Freiheitsgrade pro Dimension
Q^{1D}	Anzahl der Quadraturpunkte
$\varphi(\cdot)$	Ansatzfunktion

Abbildungsverzeichnis

1 Ansatzfunktionen φ_i [CG05, 184] 10
2 Tensor dritter Ordnung $\mathfrak{X} \in \mathbb{R}^{I \times J \times K}$ [TK09, 456] 15
3 HOSVD eines Tensors dritter Ordnung [TK09, 475] 17
4 Lexikographische Ordnung [Tea, 3] 19

1 Einführung

Das Hochleistungsrechnen ist eine neue Disziplin, die mit dem Drang entstand, immer komplexere Probleme zu lösen. Es ist eine neue Art an Probleme heranzugehen. Sie eröffnet uns eine neue Sichtweise auf dasselbe Probleme und fordert von uns andere Lösungsstrategien. Parallelisierung ist ein großer Zweig des Hochleistungsrechnens. Das Grundgerüst dieser Methodik besteht darin, komplexe Probeme modular zu lösen. Das bedeutet, 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.

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

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 [MK12, 136]. Damit wir diese Gleichung besser nachvollziehen können, werden wir sie im nächsten Kapitel herleiten. Die Matrix A_k kann durchaus, wie wir nachher erkennen werden, 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 ist. Dementsprechend wollen wir uns vor allem bei der Herleitung unserer Methoden, um diese Subprobleme zu lösen, auf sogenannte *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. Je nachdem ob es möglich ist, interessieren wir uns auch für die Berechnung von A_k^{-1}. Dies kann als Präkonditionierer genutzt werden, um (1.1) 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 (2.15) eine redundante Rolle spielt, vielmehr sollten wir eine optimale Lösung im Trade-Off zwischen Genauigkeit und

Komplexität anstreben.

Im zweiten Kapitel werden wir zuerst ein 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 obigen Gleichungen herzuleiten. Im zweiten Teil des zweiten Kapitels werden wir uns mit *Tensor Dekomposition* beschäftigen. Dies wird dadurch begründet, dass wir 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, um die Pseudoinverse zu berechnen. In der erste Methode machen wir uns die Struktur des Operators A_k zunutze und leiten eine Repräsentation von A_k her, die uns die Pseudoinverse mit wenig Aufwand gibt. In der zweiten Methode werden wir uns wie bereits erwähnt, A_k als Tensor umdefinieren und versuchen, die Singulärwertzerlegung höherer Ordnung effizient zu berechnen und uns dort auch Strukturen vom Tensor zunutze 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 zu schaffen.

Zuerst 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.

Daraufhin wird das Thema *Tensor Dekomposition* behandelt. Zuerst schauen wir uns grundlegende Definitionen an. Mit diesem Basiswissen führen wir die Singulärwertzerlegung höherer Ordnung ein. Mit dieser werden wir uns gewisse Finite Elemente Operatoren als Tensoren umdefinieren und uns deren Zerlegung anschauen. Wir werden diese Untersuchungen in Kapitel 3 und 4 mit der Hoffnung durchführen, 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

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

mit $\Omega = [\,0\,,1\,]^2$ und $f : \bar{\Omega} \to \mathbb{R}$ zweimal stetig partiell differenzierbar. Der funktionalanalytische Ansatz beschäftigt sich erst mit notwendigen Funktionenräumen, um dann auf analytischer Ebene eine Umformulierung der Differentialgleichung durchzuführen, welche letztlich die Grundlage für die Finiten-Elementen Methode liefert.

Wir sehen an der Form der Differentialgleichung, dass die gesuchte Lösung bestimmte Differenzierbarkeits- und Stetigkeitsbedingungen zu erfüllen hat, und zwar dass u zweimal partiell stetig differenzierbar sein sollte. Die Lösung sollte folglich in dem Raum der zweimal partiell stetig differenzierbaren Funktionen liegen.

Jedoch ist es möglich, dass eine Lösung für dieses Problem nicht in diesem Raum

existiert. Wir können dafür als Beispiel die Betragsfunktion $b : \mathbb{R} \to \mathbb{R}$ anschauen.

$$b(x) = |\, x \,| \tag{2.2}$$

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

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

Dies zeigt uns, dass wir potentiell auch eine Art von Lösungen, in Räumen finden können, deren Elemente nicht alle notwendigen Regularitätsbedingungen erfüllen. Diese Lösungen nennen wir *schwache Lösungen*. Doch wie finden wir schwache Lösungen? Dafür nutzen wir eine Idee, die aus der Distributionstheorie stammt. Wir multiplizieren mit einer Testfunktion $v \in C(\bar{\Omega})$ und integrieren über das Gebiet.

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

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. *(Partielle Integration)*
Es sei $\Omega \subset \mathbb{R}^n$, $n = 2, 3$ ein beschränktes Lipschitz-Gebiet. Dann gilt

$$\int_\Omega \frac{\partial\, u}{\partial\, x_i} \, v \, dx = \int_{\partial\,\Omega} u \, v \, cos(n, e^i) \, ds \; - \int_\Omega u \, \frac{\partial\, v}{\partial\, x_i} \, dx$$

für beliebige u , $v \in C^1(\bar{\Omega})$ und n bezeichne die äußere Normale im jeweiligen Randpunkt [CG05, 139].

Für unsere Differentialgleichung (2.4) nutzen wir die partielle Integration und erhalten

$$\int_\Omega -\nabla u \, \nabla v \, dx = \int_\Omega f \, v \, dx \,. \tag{2.5}$$

Dies ist die so genannte *Variationsgleichung*. Sie ist ein erstes Indiz für die später gewünschte Bilinearform der zugrundeliegenden Topologie. Die Lösung u von (2.5) nennt man *schwache Lösung* für das Problem (2.1). Die Lösung $u \in C_0^2$ von (2.1) nennt man *klassische Lösung*. An diesem Punkt steht fest, in welchem Raum die klassische Lösung liegt. Jedoch stellt sich die Frage, welche Topologie für die schwache Lösung am sinnvolsten ist. Man versucht die Räume zu definieren, in der die Lösung u für (2.5) liegt. In unserem Fall wäre folgender Raum ergiebig

$$H_0^1(\,\Omega\,) = \{\, v \,\in\, L_2(\Omega) \,:\, \frac{\partial\, v}{\partial\, x_i} \,\in\, L_2(\Omega)\,,\; v = 0 \;\text{ in }\; \partial\Omega \;,\; i = 1, 2\,\}\,.$$

Diesen Raum nennt man Sobolev Raum. Allgemein sind Sobolov Räume definiert durch:

Definition 2.2. *(Sobolev Raum)*
Es sei $\alpha = (\,\alpha_1\,,\,\ldots\,,\,\alpha_n\,)$ *mit* $\alpha_i \,\geq\, 0$ *ganzzahlig. Weiterhin sei* $|\,\alpha\,| = \sum_i \alpha_i$ *und*

$$D^\alpha\, u := \frac{\partial^{|\,\alpha\,|}}{\partial\, x_1^{\alpha_1} \,\ldots\, \partial\, x_n^{\alpha_n}}\, u\,.$$

Für $k = 1, 2, \ldots$ *definieren wir nun den Sobolev Raum durch*

$$W^{k,p}(\Omega) = \{v \,\in\, L_p(\Omega) \,:\, D^\alpha\, v \,\in\, L_p(\Omega)\,,\, |\alpha| \,\leq\, k\}.$$

Sobolev Räume sind eine Teilmenge der L_p Räume. 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 [Joh08, 36]. Beide genannten Themen würden den Rahmen dieser Bachelorarbeit deutlich überschreiten, daher verweise ich an gegebenen Stellen auf weiterführende Literatur.

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

Definition 2.3. *(Sobolev Norm)*

$$|| v ||_{W^{k,p}(\Omega)} = \begin{cases} (\sum\limits_{|\alpha| \leq k} ||D^\alpha v||^{1/p}_{L^p(\Omega)} \, dx)^{1/2} \; \textit{für} \; p < \infty \\ \max\limits_{|\alpha| \leq k} \; ||D^\alpha v||_{L^\infty(\Omega)} \, dx \quad \textit{für} \; p = \infty \end{cases}$$

$W^{k,p}(\Omega)$ ist mit der Norm $\|\cdot\|_{W^{k,p}(\Omega)}$ ein vollständiger Vektorraum, somit also ein Banachraum. Für $p = 2$ wird die Norm durch das Skalarprodukt induziert.

Definition 2.4. *(Sobolev Skalarprodukt)*

$$(u, v)_{W^{k,2}(\Omega)} := \sum_{|\alpha| \leq k} (\partial^\alpha u, \partial^\alpha v)_{L^2(\Omega)}$$

$W^{k,2}(\Omega)$ ist daher ein Hilbertraum und wir schreiben $H^k(\Omega) := W^{k,2}(\Omega)$. Für die Untersuchung elliptischer Randwertaufaben zweiter beziehungsweise vierter Ordnung sind vor allem Sobolev Räume $H^1(\Omega)$ bzw. $H^2(\Omega)$ und deren Unterräume von Bedeutung [CG05, 134].

Wir wollen noch ein wichtiges Resultat erwähnen, das Relevanz für die Herleitung der Variationsgleichung hat. Durch wiederholte Anwendung und Beachtung von Grenzübergängen von Lemma (2.1) erhält man folgendes Resultat

Satz 2.5. *(Green)*
Es sei $\Omega \subset \mathbb{R}^n$, $n = 2, 3$ *beschränktes Lipschitz-Gebiet. Dann gilt*

$$\int\limits_\Omega \nabla u \, v \, dx = \int\limits_{\partial\Omega} \frac{\partial u}{\partial n} \, v \, ds \; - \int\limits_\Omega \Delta u \, \Delta v \, dx$$

für beliebige u , $v \in C^1(\bar{\Omega})$ *und* n *bezeichne die äußere Normale im jeweiligen Randpunkt [CG05, 140].*

Der Satz von Green liefert uns ein mächtiges Werkzeug zur Herleitung der Variationsgleichung.

Nun kommen wir zurück zu unserem ursprünglichen Problem.

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

Wir wissen jede Lösung dieses Problems erfüllt die Variationsgleichung

$$\int\limits_\Omega -\nabla u \, \nabla v \, dx = \int\limits_\Omega f \, v \, dx \quad \text{für alle } v \in H^1_0(\Omega) \tag{2.6}$$

Wir definieren uns $a(\cdot,\cdot) : H_0^1(\Omega) \times H_0^1(\Omega) \to \mathbb{R}$ mit

$$a(u,v) := \int_\Omega \nabla u \, \nabla v \, dx$$

Wir können zeigen, dass die Lösung des Problems (2.6) unter gewissen Bedingungen äquivalent ist zum Lösen des folgenden Minimierungsproblems. Das Minimierungsproblem nennen wir auch *Variationsaufgabe* oder *Variationsproblem.*

Satz 2.6. *(Verbindung zwischen Variationsaufgabe und Variationsgleichung)*
Es sei $a(\cdot,\cdot) : V \times V \to \mathbb{R}$ eine Bilinearform. Sei ferner $f \in V^*$. Falls die Bilinearform $a(\,\cdot\,,\,\cdot\,)$ symmetrisch ist, ist $u \in V$ ein Minimierer der Gleichung

$$J(u) = \min_{v \in V} J(v) = \frac{1}{2} a(u,v) - f(v)\,, \tag{2.7}$$

genau dann, wenn u die schwache Formulierung löst.

2.1.2 Galerkin Verfahren

Unser Ausgangspunkt ist nun die Lösung des Problems

$$\text{Finde } u \in V : \; a(u,v) = f(v) \;\; \forall\, v \in V\,, \tag{2.8}$$

wobei V ein Banachraum, $a(\cdot,\cdot)$ eine Bilinearform und f ein Funktional auf V. Der Raum V ist in unserem Fall ein Sobolev Raum. Da Sobolev Räume unendlich dimensional sind, ist es schwer sich mit der Konstruktion einer Lösung vertraut zu machen. Daher ist die Kernidee des Galerkin Verfahrens, unseren Banachraum zu diskretisieren. Dazu führen wir eine sogenannte *konforme Approximation* durch. Wir wählen $V_n \subset V$ mit $dim(V_n) = n < \infty$. Wir erhalten ein neues diskretes Problem und haben mit (2.7) nun zwei Probleme:

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

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 *konform*, da der diskrete Raum ein Teilraum von dem ur-

sprünglichen Raum ist und das Funktional J gleich bleibt. Was hat uns das Ganze gebracht? Da $dim(V_n) = n < \infty$, können wir eine Basis für V_n, welche wir später Ansatzfunktionen nennen, bestimmen. Das verschafft den Vorteil, dass wir u_n als lineare Kombination der Basiselemente von V_n, mit endlich vielen Parametern approximieren können .

Die diskrete schwache Formulierung sieht nun wie folgt aus:

$$\text{Finde } u_n \in V_n \,:\, a(u_n, v_n) = f(v_n) \quad \forall\, v_n \,\in\, V_n\,. \tag{2.9}$$

Sei nun $e_1, \ldots, e_n$ eine Basis von V_n. Es ist ausreichend, lediglich die Basis zum Testen zu nutzen. Das obige Problem (2.9) reduziert sich auf:

$$\text{Finde } u_n \in V_n \,:\, a(u_n, e_i) = f(e_i) \quad \forall\, i \,\in\, \{1, \ldots, n\}\,. \tag{2.10}$$

Da $u_n \in V_n$ können wir u_n als Linearkombination der Basiselemente von V_n schreiben.

$$u_n = \sum_{j=1}^{n} u_j e_j \tag{2.11}$$

Setzen wir dies in Gleichung (2.10) ein, erhalten wir eine neue Darstellung des diskreten Problems.

$$\sum_{j=1}^{n} u_j a(e_j, e_i) = f(e_i) \tag{2.12}$$

Dies können wir in einem linearen Gleichungssystem mit $A_{ij} = a(e_j, e_i)$ und $u = (u_1, \ldots, u_n)^T$ zusammenfassen. Insgesamt erhalten wir:

$$Au = f \tag{2.13}$$

Somit gilt es, einen Vektor u zu finden, der diese Gleichheit erfüllt, um das diskrete Variationsproblem (2.9) zu lösen und damit eine Approximation für unser stetiges Problem (2.8) zu erhalten. Man kann allgemein davon ausgehen, dass für größeres ein n die Approximation besser wird. Das Verhalten des Fehlers in Bezug zum Diskretierisierungsparameter n wird in [CG05, 154] ausgiebig untersucht.

Bemerkung 2.7. *(Galerkin Eigenschaften)*

1. *Galerkin Orthogonalität*
   Der Fehler liegt orthogonal auf dem Teilraum von V.

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 *Galerkin Verfahren* kennengelernt. Der Kernaspekt dieser konformen Approximation war eine Diskretisierung des Raumes und damit einhergehend global einheitlich definierte Basisfunktionen des diskreten Raumes. Nun öffnen wir die zuletzt genannte Einschränkung und fordern nur noch stückweise definierte Funktionen. Wo genau eine Funktion, in der Regel ein Polynom, definiert ist, hängt von unsererer Gebietszerlegung ab. Das heißt für die *finite Elemente Methode (FEM)* sind folgende Schritte nötig:

1. Das Grundgebiet in geometrisch einfache Teilgebiete $\Omega_h = \{ \Omega_k \}_{k=1,\dots,N}$ zu unterteilen, zum Beispiel Dreiecke und Rechtecke bei Problemen in der Ebene oder Tetraeder und Quader bei Problemen im dreidimensionalen Raum.

2. Definition von Ansatz- und Testfunktionen über Teilgebieten

3. Da wir zwischen den Teilgebieten eine Stetigkeit fordern, definiert man Übergangsbedigungen, die uns globale Stetigkeit sichern

Die Stetigkeit der globalen Lösung wird gefordert, damit wir $V_n \subset V$ bekommen, mit V ein Sobolev Raum. [CG05, 175].

Bemerkung 2.8. *(Voraussetzungen an Zerlegung) [CG05, 176]*
Die Voraussetzungen an die Zerlegung $Z = \{ \Omega_j \}_{j=1}^m$ sind

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

2. $int(\Omega_i) \cap int(\Omega_j) = \emptyset$ *, falls* $i \neq j$

Beispiel 2.9. *E sei* $\Omega = [a, b]$. *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 $Z = \{ \Omega_j \}_{j=1}^{m}$ *mit* $\Omega_i := (x_{i-1} , x_i)$ *für* $i = 1 , \ldots , N$. *Ferner sei* $h_i := x_i - x_{i-1}$, $i = 1, \ldots, N$. *Wir wählen lineare Ansatzfunktionen* $V_h = lin \{ \varphi_i \}_{i=0}^{N}$, *wobei die Ansatzfunktionen durch*

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

definiert sind. Es gilt nach Konstruktion $\varphi \in C(\bar{\Omega})$ *sowie* $\varphi_i \mid_{\Omega_j} \in C^1(\bar{\Omega}_j)$, *somit hat man insgesamt* $\varphi_i \in H^1(\Omega)$ *[CG05, 184]. Die folgende Abbildung stellt die Graphen von Ansatzfunktionen* φ_i *dar.*

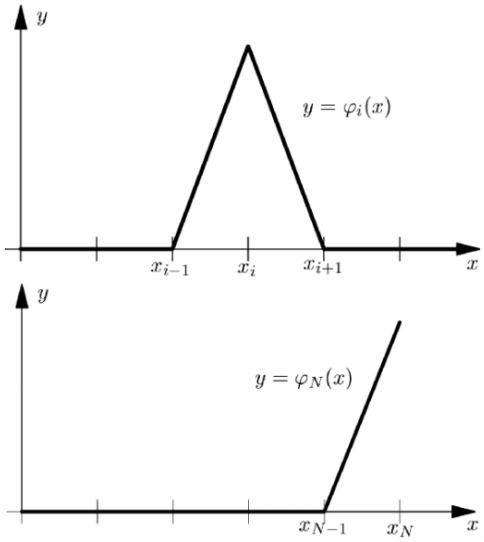

Abbildung 1: Ansatzfunktionen φ_i [CG05, 184]

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

Nun haben wir eine Vorstellung davon, wie man ein Gebiet zerlegt, wie Ansatzfunktionen aussehen könnten und welche Eigenschaften sie zu erfüllen haben. Doch wie genau sieht das diskrete Problem bei der *finite Elemente Methode* aus? Dazu

müssen wir uns die sogenannte *Assemblierung* der *Steifigkeitsmatrix* A_n anschauen. Wir werden uns die Teilelemente der Zerlegung dazu einzeln anschauen und sogenannte *Elementsteifigkeitsmatrizen* ausrechnen. Im *Assemblierungsschritt* werden wir dann die *Elementsteifigkeitsmatrizen* zur *globalen Steifigkeitsmatrix* zusammensetzen. Wir werden hier nodale Basisfunktionen benutzten. Diese sind durch $\varphi_k(x_l) = \delta_{kl}$ definiert. Weiterhin sei N die Anzahl der Freiheitsgrade. Rekapitulieren wir das zu lösende Problem:

1. Finde ein $u_n \in V_n : a(u_n, v_n) = f(v_n)$ für alle $v_n \in V_n$
2. Sei $\{\varphi_i\}_{i=1}^N$ die Basis von V_n
3. Definiere $A_n = (a(\varphi_k, \varphi_i))_{i,k=1}^N$ und $f_n = (f(\varphi_i))_{i=1}^N$
4. Löse lineares Gleichungssystem $A_n u_n = f_n$ zur Bestimmung der Koeffizienten u_i der Darstellung $u_n(x) = \sum_{i=1}^N u_i \varphi_i(x)$.

In unserem Fall war $a(u,v) = \int_\Omega \Delta u \, \Delta v \, dx$ bzw. $f(v) = \int_\Omega f \, v \, dx$. Die zu Ω_j gehörige Elementsteifigkeitsmatrix besitzt die Form

$$\begin{aligned} A_h^j &= (a_{ik}^j)_{i,k \in I_j}\,, \\ a_{ik}^j &= \int_{\Omega_j} \Delta\varphi_i \, \Delta\varphi_k \, dx \quad \text{mit} \quad I_j = \{i \,:\, supp\,\varphi_i \cap \Omega_j \neq \emptyset\}. \end{aligned} \tag{2.14}$$

Analog dazu die elementweise rechte Seite

$$f^j = (f_i^j)_{i \in I_j} \quad \text{mit} \quad f_i^j = \int_\Omega f \, \varphi_i \, dx\,.$$

Die globale Steifigkeitstmatrix A_n und die rechte Seite f_n ergeben sich dann durch die Addivität des Integrals als Summen

$$A_n = (a_{ik})_{i,k=1}^N \quad \text{mit} \quad a_{ik} = \sum_{j=1\,,\, i \in I_j\,,\, k \in I_j}^{M} a_{ik}^j$$

und

$$f_h = (f_i)_{i=1}^N \quad \text{mit} \quad f_i = \sum_{j=1\,,\, i\in I_j}^{M} f_i^j \,.$$

I_j ist gerade die Liste der Eckpunkte der Elemente. Wir sehen für die Elemente der Elementsteifigkeitsmatrix in (2.14) ein Integral über ein Teilgebiet Ω_j. Hier liegt eine gute Chance viele Operationen zu sparen, indem wir uns die Struktur des Integrals zunutze machen und mit Hilfe vom Transformationssatz für Integrale eine einheitlichere Form herleiten.

Dazu definieren uns sogenannten *Referenzelemente*. Beispielsweise wenn wir eine Zerlegung in Rechtecke wählen, würden wir uns ein Referenzrechteck definieren oder im Falle von Dreiecken ein Referenzdreieck. Weiterhin definieren wir uns eine lineare Transformation, die die Ecken unseres Teilgebiets auf die Ecken des korrespondierenden Referenzelements abbildet. Mit Hilfe des Transformationssatzes formen wir das Integral um und erhalten für alle Elementsteifigkeitsmatrizen dieselben Integrale, bloß mit verschiedenen Konstanten multipliziert. Die Konstanten sind gerade die Beträge der Determinanten der linearen Transformationen.

Um die Elemente der Elementsteifigkeitsmatrizen in die globale Steifigkeitsmatrix einzubetten, ist dies eine Frage des Umdenkens von einer lokalen Struktur in die globale Struktur. Dieses Umdenken ist ausschlaggebend für das Verstehen des *finiten Elemente Ansatzes* und auch der späteren Arbeit zur Herleitung der Pseudoinversen.

Wir rekapitulieren die erste Gleichung dieser Arbeit

$$v = A(u) = \sum_{k=1}^{n_{cells}} P_k^T A_k (P_k u). \tag{2.15}$$

Die Variable n_{cells} ist somit die Anzahl der Teilgebiete Ω_j. Die Matrix A_k ist der elementspezifische Operator A. Die Matrix P_k ist jene zuvor erwähnte Transformation, nämlich die Transformation von den lokalen Freiheitsgraden in die globalen Freiheitsgrade.

Die zu betrachtenden Strukturen sollen im Folgenden erwähnt werden. Das ist einerseits die Massenmatrix mit der Bilinearform

$$a_M(u, v) = \int_\Omega u\, v \, dx \,. \tag{2.16}$$

Eine einfache Struktur, die dazu dient sich an die Thematik ranzutasten und ein Gefühl dafür zu bekommen.

Des Weiteren wollen wir die Bilinearform des bereits erwähnten elliptischen Problems untersuchen mit

$$a_L(u,v) = \int_\Omega \Delta v \, \Delta v \, dx \,. \tag{2.17}$$

Außerdem benötigen wir etwas Hintergrundwissen, wie wir die Integrale in (3.1) und in (2.17) berechnen können. In der Praxis erfolgt die Berechnung von Integralen über *Quadraturformeln.*

2.1.4 Quadratur

Allgemein beschäftigt uns das Integrationsproblem

$$I(f) \;=\; \int_a^b f(x)\, dx \,, \tag{2.18}$$

mit $a, b \in \mathbb{R},\ a < b$ und $f \in C[\,a, b\,]$. Wir werden von *interpolatorischen Quadraturformeln* Gebrauch machen. Dahinter steckt eine Polynominterpolation der zu integrierenden Funktion f und eine Summation über die Stützstellen.

Es seien $x_i \in \mathbb{R}\,,\ a \le x_0 < \ldots < x_n \le b$ Stützstellen mit Gewichten $\alpha_i \in \mathbb{R}$. Dann können wir das Integral (2.19) approximieren durch

$$I(f) \;=\; \int_a^b f(x)\, dx \;\approx\; \sum_{i=0}^{n} \alpha_i \, f(x_i) \;= I^{(n)}(f)\,. \tag{2.19}$$

Hierbei stellt sich die Frage nach der Präzision der Approximation. Interpolatorische Quadraturformeln $I^{(n)}(\cdot)$ zu $n+1$ Stützstellen sind mindestens von Ordnung $n+1$. Das heißt sie integrieren alle Polynome von maximalen Grad n exakt.

Eine Quadraturformel soll besonders hervorgehoben werden, da diese für unsere Anwendung außerordentliche Praktikabilität gezeigt hat.

Definition 2.10. *(Gauss Lobatto)*
Es sei x_i die $(i-1)$ te Nullstelle des Legendre Polynoms $P_{n-1}^{`}(x)$ und n die Anzahl der Stützstellen, bzw. $n-1$ der Grad unseres Legendre Polynoms. Dann können wir das Integral auf dem Gebiet $[-1,1]$ der Funktion f approximieren durch

$$\int_{-1}^{1} f(x)\, dx \approx \frac{2}{n(n-1)}[f(1)+f(-1)] + \sum_{i=2}^{n-1} w_i f(x_i),$$

mit Gewichten

$$w_i = \frac{2}{n(n-1)[P_{n-1}(x_i)]^2}, \qquad x_i \neq \pm 1\,.$$

Gauss Lobatto liefert uns eine exakte Approximation von Polynomen bis zu Grad $2n-3$. Für eine ausführliche Ausarbeitung der Thematik wird auf [Ran05, 79] verwiesen.

2.2 Tensor Dekomposition

2.2.1 Einführung in die Tensor Architektur

In Kapitel 3 werden wir sehen, dass es möglich ist die Elementsteifigkeitsmatrix der Massenmatrix und der Laplace Bilinearform als Tensor zu definieren. Daraufhin werden wir die Strukturen dieser Tensoren untersuchen. Das Ziel dieser Untersuchungen ist einen einfachen Weg zu finden, die korrespondierende Pseudoinverse zu bestimmen. Was genau ein Tensor ist, was wir unter der Pseudoinversen eines Tensors verstehen und wie wir einen Tensor analysieren, wird in diesem Unterkapitel beantwortet.

Definition 2.11. *(Tensor)*
Ein Tensor ist 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* N.

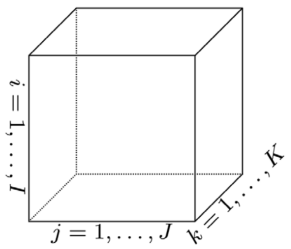

Abbildung 2: Tensor dritter Ordnung $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$ [TK09, 456]

Um Tensoren zu klassifizieren und charakterisieren, benötigen wir Eigenschaftsbegriffe. Es sei $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ ein Tensor.

Definition 2.12. *(Symmetrie)*

a) *Den Tensor* $\mathcal{X}$ *nennt man kubisch genau dann, wenn* $I_i = I_j$ *für alle* i, j.

b) *Einen kubischen Tensor nennt man supersymmetrisch genau dann, wenn die Elemente des Tensors konstant bleiben unter jeglicher Permutation der Indizes.*

c) *Einen Tensor nennt man stückweise symmetrisch, wenn die Elemente konstant bleiben unter der Permutation von mindestens 2 Indizes.*

Definition 2.13. *(Diagonal)*
Den Tensor $\mathcal{X}$ *nennt man diagonal, wenn* $x_{i_1, \ldots, i_N} \neq 0$ *genau dann wenn* $i_1 = \cdots = i_N$.

Definition 2.14. *(Faser)*
Eine Faser ist das multidimensionale Analog zu Matrixspalten und Matrixzeilen. Wir definieren eine Faser, indem wir jeden Index abgesehen von einem festhalten.

Einen Tensor kann man entfalten. Dies impliziert eine Neuordnung der Tensorelemente in eine Matrix. Wir betrachten nur die sogenannte *mode-n Entfaltung*, da dies die einzig relevante Form der Entfaltung für uns ist.

Bemerkung 2.15. *(Entfaltung)*
Eine mode-n Entfaltung des Tensors $\mathcal{X}$ *wird mit* $\mathbf{X}_{(n)}$ *geschrieben und ordnet die mode* $-$ *n Fasern in die Spalten der Ergebnismatrix. Formal ist es eine Abbildung des Indize N-tupels* $(i_1, \ldots, i_N)$ *auf das Matrixindize-Tupel* (i_n, j)

$$j = 1 + \sum_{\substack{k=1 \\ k \neq n}}^{N} (i_k - 1) J_k \text{ mit } J_k = \prod_{\substack{m=1 \\ m \neq n}}^{k-1} I_m \,. \tag{2.20}$$

An dieser Stelle wird noch eine Definition der Tensor Multiplikation benötigt, um mit der Dekomposition von Tensoren anzufangen.

Definition 2.16. *(n-mode Produkt)*
Das n-mode Produkt des Tensors $\mathcal{X}$ *mit einer Matrix* $\mathbf{U} \in \mathbb{R}^{J \times I_n}$ *wird als* $\mathcal{X} \times_n \mathbf{U}$ *notiert. Die Ergebnismatrix hat die Größe* $I_1 \times \ldots I_{n-1} \times J \times I_{n+1} \times \ldots I_N$

$$(\mathcal{X} \times_n \mathbf{U})_{i_1 \ldots i_{n-1}\, j\, i_{n+1} \ldots i_N} = \sum_{i_n=1}^{I_n} x_{i_1 \ldots i_N}\, u_{j\, i_n} \,. \tag{2.21}$$

Bemerkung 2.17. *Jedes n-mode Produkt kann mit Hilfe von entfalteten Tensoren äquivalent ausgedrückt werden.*

$$\mathcal{Y} = \mathcal{X} \times_n \mathbf{U} \Longleftrightarrow \mathbf{Y}_{(n)} = \mathbf{U}\mathbf{X}_{(n)} \,. \tag{2.22}$$

2.2.2 Singulärwertzerlegung höherer Ordnung

Die *Singulärwertzerlegung höherer Ordnung* bzw. *Higher Order Singular Value Decomposition (HOSVD)*oder auch bekannt unter *Tucker Dekomposition*, ist eine uminterpretierte multidimensionale Hauptkomponentenanalyse. Die HOSVD zerlegt den Tensor in einen sogenannten *Core Tensor* multipliziert mit einer Matrix in jeder Ordnung beziehungsweise mode.

Allgemein ist die HOSVD des Tensors $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ gegeben durch

$$\mathcal{X} = \mathcal{G} \times_1 A^{(1)} \ldots \times_N A^{(N)} . \tag{2.23}$$

Man kann äquivalent die HOSVD, wie in [TK09, 462] praktiziert, auch mit entfalteten Tensoren wie folgt angeben

$$\mathbf{X}_{(n)} = A^{(n)} \mathbf{G}_{(n)} (A^{(N)} \otimes \ldots \otimes A^{(n+1)} \otimes A^{(n-1)} \otimes \cdots \otimes A^{(1)})^T . \tag{2.24}$$

Beispiel 2.18. *(HOSVD Tensor dritter Ordnung)*
Es sei $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$. Dann kann man den Tensor $\mathcal{X}$ zerlegen in

$$\mathcal{X} \approx \mathcal{G} \times_1 A \times_2 B \times_3 C , \tag{2.25}$$

wobei $A \in \mathbb{R}^{I \times P}$, $B \in \mathbb{R}^{J \times Q}$ und $C \in \mathbb{R}^{K \times R}$ die orthogonalen Faktormatrizen sind. Der Tensor $\mathcal{G}$ bezeichnet den Kerntensor und zeigt wie hoch die Korrelation zwischen den verschiedenen Komponenten ist.

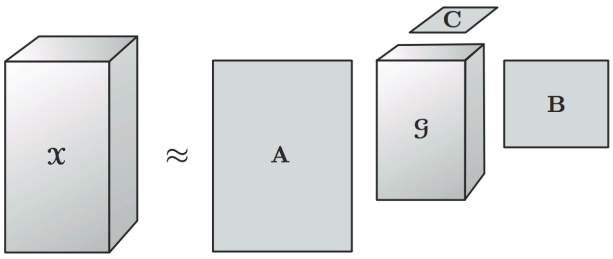

Abbildung 3: HOSVD eines Tensors dritter Ordnung [TK09, 475]

Bemerkung 2.19. *(Berechnung der HOSVD)*
Die Berechnung der HOSVD von $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ funktioniert wie folgt:

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

Die *HOSVD* existiert für alle Tensoren, jedoch ist die *HOSVD* keine eindeutige Zerlegung. Dies führen wir an einem beliebigen Tensor dritter Ordnung vor.

Beispiel 2.20. *(Eindeutigkeit der HOSVD)*
Es seien $U \in \mathbb{R}^{P\times P}$, $V \in \mathbb{R}^{Q\times Q}$ *und* $W \in \mathbb{R}^{R\times R}$ *. Es gilt*

$$\mathcal{X} = \mathcal{G} \times_1 A \times_2 B \times_3 C = (\mathcal{G} \times_1 U \times_2 V \times_3 W) \times_1 AU^{-1} \times_2 BV^{-1} \times_3 CW^{-1}. \quad (2.26)$$

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 dieser Kenntnis können wir nun zum Beispiel versuchen, so viele Elemente des Kerntensors wie möglich auf Null zu bringen oder so klein wie möglich zu machen, damit entstehen bei der späteren Herleitung der Pseudoinversen weniger Probleme.

Es bedarf noch einiger Eigenschaften des Kronecker Produkts, die wir später für die Berechnung der Pseudoinversen nutzen wollen.

Lemma 2.21. *(Matrixprodukt und Kronecker Produkt)*
Es seien beliebige A, B, C, D *Matrizen, deren Matrizenprodukte* AC *und* BD *definiert sind. Dann gilt*

$$AC \otimes BD = (A \otimes B)(C \otimes D).$$

Lemma 2.22. *(Invertieren des Kronecker Produkts)*
Es seien $A \in \mathbb{R}^{i\times i}$ *und* $B \in \mathbb{R}^{j\times j}$ *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 analog

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

Lemma 2.23. *(Transponieren)*
Es seien A, B *beliebige Matrizen. Es gilt*

$$(A \otimes B)^{T} = A^{T} \otimes B^{T}.$$

Ausführliche Untersuchungen des Kronecker Produkts finden sich in [Loa99]. Diese Ergebnisse sind entscheidend für die Herleitung der Pseudoinversen. Die theoretische Grundlage ist geschaffen, somit widmen wir uns im Folgenden Kapitel der Tensorstruktur der Elementarsteifigkeitsmatrizen und die Herleitung der Pseudoinversen.

3 Pseudoinversen der Zellmatrizen

3.1 Summenfaktorisierung

Das Ziel dieses Unterkapitels ist es die Tensorstruktur für die Massenmatrix und der Laplace Bilinearform herzuleiten und für die Berechnung der Pseudoinversen zu nutzen.

3.1.1 Tensorstruktur der Elementmassenmatrix

Es sei T die Referenzzelle für Rechtecke und die $\varphi_i^{2D}(\mathbf{x})$ zweidimensionalen Basisfunktionen, mit $i \in \{1, \ldots, n\}$, des diskreten Raumes V_n mit $\mathbf{x} = (x, y)$. Die Massenmatrix kann elementweise dargestellt werden durch:

$$M_{ij} = \int_T \varphi_i^{2D}(\mathbf{x})\, \varphi_j^{2D}(\mathbf{x})\, d\mathbf{x}. \tag{3.1}$$

Wir haben hier die Konstante für die Transformation weggelassen, da diese für unsere Arbeit keine Relevanz hat. Die Basisfunktionen haben eine Tensorstruktur, die wie folgt aussieht

$$\varphi_i^{2D}(\mathbf{x}) = \varphi_{i_1+(N+1)i_2}^{2D}(x, y) = \varphi_{i_1}^{1D}(x)\varphi_{i_2}^{1D}(y), \tag{3.2}$$

wobei φ^{2D} eine eindimensionale reelle Basisfunktion ist.

Wir werden eine lexikographische Ordnung der Freiheitsgrade benutzen. Die Reichweite von i_1 und i_2 reicht von 1 bis N. Das heißt der Index i geht von 1 bis $N_p = N^2$. In Abbildung (4) findet sich ein Beispiel für $N = 4$.

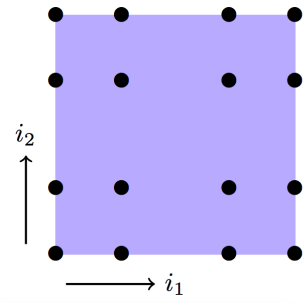

Abbildung 4: Lexikographische Ordnung [Tea, 3]

Wir wollen für die Berechnung der Integrale in (3.1) die Gauss Quadratur benut-

zen. Es seien $\mathbf{x}_q = (x_{q1}, x_{q2})$ die Stützstellen und $\mathbf{w}_q = w_{q1}w_{q2}$ die Gewichte. Die Gleichung (3.1) kann approximiert werden durch

$$\begin{aligned} M_{ij} &= \int_T \varphi_i(\mathbf{x})\,\varphi_j(\mathbf{x})\,d\mathbf{x} \\ &\approx \sum_{q=1}^{Q} \mathbf{w}_q\ \varphi_i(\mathbf{x}_q)\,\varphi_j(\mathbf{x}_q) \\ &= \sum_{q_1=1}^{Q_{1D}} \sum_{q_2=1}^{Q_{1D}} \varphi_{i_1}(x_{q1})\varphi_{i_2}(x_{q2})\varphi_{j_1}(x_{q1})\varphi_{j_2}(x_{q2})\,w_{q1}w_{q2} \\ &= \sum_{q_1=1}^{Q_{1D}} w_{q1}\varphi_{i_1}(x_{q1})\varphi_{j_1}(x_{q1}) \sum_{q_2=1}^{Q_{1D}} w_{q2}\varphi_{i_2}(x_{q2})\varphi_{j_2}(x_{q2})\,. \end{aligned} \tag{3.3}$$

Wir wählen die Anzahl der Quadraturpunkte Q_{1D} per Dimension so, dass wir exakt integrieren. Dazu ist die Wahl von Q_{1D} gleich $N+1$ geeignet, da wir mit der Gauss Quadratur mit $N+1$ Stützstellen bis $2N$ exakt integrieren und der höchste Grad bei uns $2N$ ist.

Wir definieren uns eine Matrix durch $\mathcal{N}_{iq} = \varphi_i(\mathbf{x}_q)$ und weiterhin die Matrix $\mathcal{W}_{ii} = \mathbf{w}_i$, die in der Diagonalen die Quadraturgewichte hat und sonst Nullen. Damit können wir nun die Massenmatrix schreiben als

$$M = \mathcal{N}\mathcal{W}\mathcal{N}^T\,. \tag{3.4}$$

In $\mathcal{N}$ sind die Elemente an Stützstellen evaluierte zweidimensionale Basisfunktionen. Wir können diese aber weiter aufspalten in eindimensionale Basisfunktionen. Dadurch erzielen wir eine Effizienzsteigerung bei der Berechnung des Matrix-Vektor Produkts mit der Elementsteifigkeitsmatrix der Massenmatrix.

Wir beginnen damit, die Matrix $\mathcal{N}$ in ein Tensorprodukt aufzuspalten.

$$\mathcal{N} = \mathcal{N}^{1D} \otimes \mathcal{N}^{1D} \tag{3.5}$$

Die Matrix $\mathcal{N}^{1D}$ ist nun äquivalent definiert wie $\mathcal{N}$, nur mit eindimensionalen Ansatzfunktionen. Ausgeschrieben sieht die Gleichung (3.5) wie folgt aus:

$$\begin{bmatrix} \varphi_1^{2D}(\boldsymbol{x}_1) & \dots & \varphi_N^{2D}(\boldsymbol{x}_1) \\ \vdots & \ddots & \vdots \\ \varphi_1^{2D}(\boldsymbol{x}_Q) & \dots & \varphi_N^{2D}(\boldsymbol{x}_Q) \end{bmatrix} = \begin{bmatrix} \varphi_1^{1D}(x_1) & \dots & \varphi_n^{1D}(x_1) \\ \vdots & \ddots & \vdots \\ \varphi_1^{1D}(x_{Q_{1D}}) & \dots & \varphi_n^{1D}(x_{Q_{1D}}) \end{bmatrix} \otimes \begin{bmatrix} \varphi_1^{1D}(x_1) & \dots & \varphi_n^{1D}(x_1) \\ \vdots & \ddots & \vdots \\ \varphi_1^{1D}(x_{Q_{1D}}) & \dots & \varphi_n^{1D}(x_{Q_{1D}}) \end{bmatrix}.$$

Wir nutzen absofort $Q_{1D} = N+1$ und $Q = (N+1)^2$, damit integrieren wir exakt. Wir erinnern uns $M = \mathcal{N}\mathcal{W}\mathcal{N}^T$. Da $\mathcal{W}$ eine Diagonalmatrix ist, ist es naheliegend diese Multiplikation bereits durchzuführen. Definiere $\mathcal{W}_N = \mathcal{N}\mathcal{W}$ und die Spaltung von $\mathcal{W}_N$ kann im nächsten Schritt hergeleitet werden.

$$\mathcal{W}_N = \mathcal{W}_N^{1D} \otimes \mathcal{W}_N^{1D} \tag{3.6}$$

Genauer

$$\begin{bmatrix} \mathcal{N}_{11}\mathbf{w}_1 & \dots & \mathcal{N}_{1N}\mathbf{w_N} \\ \vdots & \ddots & \vdots \\ \mathcal{N}_{N1}\mathbf{w}_1 & \dots & \mathcal{N}_{NN}\mathbf{w_N} \end{bmatrix} = \begin{bmatrix} \mathcal{N}_{11}^{1D}w_1 & \dots & \mathcal{N}_{1N}^{1D}w_N \\ \vdots & \ddots & \vdots \\ \mathcal{N}_{N1}^{1D}w_1 & \dots & \mathcal{N}_{NN}^{1D}w_N \end{bmatrix} \otimes \begin{bmatrix} \mathcal{N}_{11}^{1D}w_1 & \dots & \mathcal{N}_{1N}^{1D}w_N \\ \vdots & \ddots & \vdots \\ \mathcal{N}_{N1}^{1D}w_1 & \dots & \mathcal{N}_{NN}^{1D}w_N \end{bmatrix}.$$

Damit können wir folgende Umformulierung bereits vornehmen

$$M = \mathcal{N}\mathcal{W}\mathcal{N}^T = \mathcal{W}_N\mathcal{N}^T = [\mathcal{W}_N^{1D} \otimes \mathcal{W}_N^{1D}][\mathcal{N}^{1D} \otimes \mathcal{N}^{1D}]^T. \tag{3.7}$$

Mit Lemma (2.23) folgt

$$[\mathcal{W}_N^{1D} \otimes \mathcal{W}_N^{1D}][\mathcal{N}^{1D} \otimes \mathcal{N}^{1D}]^T = [\mathcal{W}_N^{1D} \otimes \mathcal{W}_N^{1D}][(\mathcal{N}^{1D})^T \otimes (\mathcal{N}^{1D})^T].$$

Dann nutzen wir Lemma (2.21) und erhalten

$$[\mathcal{W}_N^{1D} \otimes \mathcal{W}_N^{1D}][(\mathcal{N}^{1D})^T \otimes (\mathcal{N}_{1D})^T] = [\mathcal{W}_N^{1D}(\mathcal{N}^{1D})^T] \otimes [\mathcal{W}_N^{1D}(\mathcal{N}_{1D})^T]. \tag{3.8}$$

3.1.2 Tensorstruktur der Elementsteifigkeitsmatrix der Laplace Bilinearform

Der Ansatz kann leicht geändert, benutzt werden um andere Bilinearformen auszudrücken wie die Laplace Bilinearform. Es sei die Elementsteifigkeitsmatrix der Laplace Bilinearform elementweise gegeben durch

$$V_{ij} = \int_T \nabla\varphi_i(\mathbf{x})\,\nabla\varphi_j(\mathbf{x})d\mathbf{x} = \int_T (\partial_{x_1}\varphi_i(\mathbf{x})\partial_{x_1}\varphi_j(\mathbf{x})) + (\partial_{x_2}\varphi_i(\mathbf{x})\partial_{x_2}\varphi_j(\mathbf{x}))\,d\mathbf{x}. \tag{3.9}$$

Wir nutzen die Linearität des Integrals.

$$\begin{aligned} V_{ij} &= \int_T (\partial_{x_1}\varphi_i(\mathbf{x})\partial_{x_1}\varphi_j(\mathbf{x})) + (\partial_{x_2}\varphi_i(\mathbf{x})\partial_{x_2}\varphi_j(\mathbf{x}))\, d\mathbf{x} \\ &= \int_T \partial_{x_1}\varphi_i(\mathbf{x})\partial_{x_1}\varphi_j(\mathbf{x}) d\mathbf{x} + \int_T \partial_{x_2}\varphi_i(\mathbf{x})\partial_{x_2}\varphi_j(\mathbf{x})\, d\mathbf{x} \end{aligned} \tag{3.10}$$

Für die Integrale verwenden wir erneut die Gauss Quadratur. Es seien $\mathbf{x}_q = (x_{q1}, x_{q2})$ die Stützstellen und $\mathbf{w}_q = w_{q1}w_{q2}$ die Gewichte, dann folgt für die obige Gleichung

$$V_{ij} = \underbrace{\sum_{q=1}^{(N+1)^2} \mathbf{w}_q \partial_{x_1}\varphi_i(\mathbf{x}_q)\partial_{x_1}\varphi_j(\mathbf{x}_q)}_{K^1} + \underbrace{\sum_{q=1}^{(N+1)^2} \mathbf{w}_q \partial_{x_2}\varphi_i(\mathbf{x}_q)\partial_{x_2}\varphi_j(\mathbf{x}_q)}_{K^2}. \tag{3.11}$$

Wir können eine große Ähnlichkeit mit der Struktur der Massenmatrix in (3.3) feststellen, wenn wir uns jeweils K^1 und K^2 seperat anschauen. Jetzt gilt es die Tensorstruktur der Ansatzfunktionen auszunutzen. Dafür betrachten wir K^1:

$$\begin{aligned} K^1_{ij} &= \sum_{q=1}^{(N+1)^2} \mathbf{w}_q \partial_{x_1}\varphi_i(\mathbf{x}_q)\partial_{x_1}\varphi_j(\mathbf{x}_q) \\ &= \sum_{q_1=1}^{N}\sum_{q_2=1}^{N} w_{q1}w_{q2}\partial_{x_1}\varphi_{i1}(x_{q1})\varphi_{i2}(x_{q2})\partial_{x_1}\varphi_{j1}(x_{q1})\varphi_{j2}(x_{q2}) \\ &= \sum_{q_1=1}^{N}\sum_{q_2=1}^{N} w_{q1}w_{q2}\varphi'_{i1}(x_{q1})\varphi_{i2}(x_{q2})\varphi'_{j1}(x_{q1})\varphi_{j2}(x_{q2}) \\ &= \sum_{q_1=1}^{N} w_{q1}\varphi'_{i1}(x_{q1})\varphi'_{j1}(x_{q1}) \sum_{q_2=1}^{N} w_{q2}\varphi_{i2}(x_{q2})\varphi_{j2}(x_{q2}). \end{aligned} \tag{3.12}$$

Man kann die Ähnlichkeit mit der Gleichung (3.3) erkennen. Wir sehen, dass wir in einer Dimension nun aber die evaluierten Ableitungen der Ansatzfunktionen und in der anderen Dimension die evaluierten Ansatzfunktionen haben. Wir definieren uns zwei Matrizen $\widehat{\mathcal{W}}_N^{1D}$ und $\widehat{\mathcal{N}}^{1D}$ die ähnlich sind zu $\mathcal{W}^{1D}$ und $\mathcal{N}^{1D}$ mit dem Unterschied, dass diese Matrizen nicht die Ansatzfunktionen evaluieren, sondern deren Ableitung.

$$\widehat{\mathcal{N}}^{1D} = \begin{bmatrix} \varphi_1'^{1D}(x_1) & \dots & \varphi_n'^{1D}(x_1) \\ \vdots & \ddots & \vdots \\ \varphi_1'^{1D}(x_N) & \dots & \varphi_n'^{1D}(x_N) \end{bmatrix} \tag{3.13}$$

Analog definieren uns $\widehat{\mathcal{W}}_N^{1D}$. Dann folgt mit analoger Umformung wie bei der Massenmatrix

$$K_1 = (\widehat{\mathcal{W}}_N^{1D}(\widehat{\mathcal{N}}^{1D})^T) \otimes (\mathcal{W}_N^{1D}(\mathcal{N}^{1D})^T)$$
$$K_2 = (\mathcal{W}_N^{1D}(\mathcal{N}^{1D})^T) \otimes (\widehat{\mathcal{W}}_N^{1D}(\widehat{\mathcal{N}}^{1D})^T).$$

Insgesamt kann man die Elemensteifigkeitsmatrix für die Laplace Bilinearform darstellen als

$$V = [\widehat{\mathcal{W}}_N^{1D}(\widehat{\mathcal{N}}^{1D})^T] \otimes [\mathcal{W}_N^{1D}(\mathcal{N}^{1D})^T] + [\mathcal{W}_N^{1D}(\mathcal{N}^{1D})^T] \otimes [\widehat{\mathcal{W}}_N^{1D}(\widehat{\mathcal{N}}^{1D})^T]. \tag{3.14}$$

3.1.3 Pseudoinverse

Die Pseudoinversen für die Elementsteifigkeitsmatrix der Massematrix M und der Laplace Bilinearform V sollen im Folgenden hergeleitet werden. Dank der Vorarbeit und dem Vorwissen zum Kronecker Produkt ist diese Herleitung problemlos möglich.

Rekapituliere die Tensorstruktur für die Massematrix

$$M = ([\mathcal{W}_N^{1D}(\mathcal{N}^{1D})^T] \otimes [\mathcal{W}_N^{1D}(\mathcal{N}_{1D})^T]. \tag{3.15}$$

Nun wollen wir die Pseudoinverse herleiten

$$M^+ = [(\mathcal{W}_N^{1D}(\mathcal{N}^{1D})^T] \otimes [\mathcal{W}_N^{1D}(\mathcal{N}_{1D})^T)]^+. \tag{3.16}$$

Wir nutzen Lemma (2.22) und erhalten

$$[(\mathcal{W}_N^{1D}(\mathcal{N}^{1D})^T] \otimes [\mathcal{W}_N^{1D}(\mathcal{N}_{1D})^T])^+ = [\mathcal{W}_N^{1D}(\mathcal{N}^{1D})^T]^+ \otimes [\mathcal{W}_N^{1D}(\mathcal{N}_{1D})^T]^+. \tag{3.17}$$

An dieser Stelle kann entweder eine Singulärwertzerlegung zum Berechnen der Pseudoinversen oder der Gauss Algorithmus zum berechnen der Inversen verwendet werden. Warum das Ganze? Wir hätten auch die Inverse von der Massematrix berechnen können. Der Unterschied ist, dass wir anstatt der Inversen einer Matrix der

Größe $N^2 \times N^2$ berechnen, wir die Inverse einer Matrix der Größe $N \times N$ berechnen.

Für die Laplace Bilinearform können wir dies leider nicht so einfach machen. Das Problem hierbei ist die Addition, die uns das Ganze stark erschwert. Je nach Basis jedoch, kann man das Problem vereinfachen. Man könnte beispielsweise die Lagrange Basis nehmen und geeigneten Stützstellen verwenden und würde für $\mathcal{N}^{1D}$ und $\mathcal{W}_N^{1D}$ Diagonalmatrizen bekommen. Auf die Möglichkeit, die Einheitsmatrix für das Produkt dieser beiden Matrizen zu erhalten, wird im weiteren Verlauf dieser Arbeit eingegangen.

Hauptsächlich von Interesse ist die Auswertung der Pseudoinversen an einem Vektor u also M^+u und V^+u. Wenn wir später diese Strukturen betrachten, werden wir ein Ergebnis aus der Stabilitätstheorie verwenden können, um das Problem mit der Laplace Bilinearform zu lösen. Zudem werden wir uns effiziente Algorithmen anschauen, um diese Gleichungen optimal zu berechnen.

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 Massenmatrix und zur Steifigkeitsmatrix der Laplace Bilinearform effiizient berechnen können. Es sei $\mathfrak{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_K}$ ein Tensor. Dann können wir mit der HOSVD diesen Tensor zerlegen

$$\mathfrak{X} = \mathfrak{G} \times_{n=1}^{K} U^{(n)}. \tag{3.18}$$

In dieser Arbeit werden wir uns auf $K = 4$ konzentrieren, da unsere zu betrachtenden Tensoren von Ordnung 4 sind.

Wir können einen Tensor $\mathfrak{X}$ in einen Kerntensor $\mathfrak{G}$ und zugehörige Faktormatrizen $U^{(n)}$ zerlegen. Wie bekommen wir nun die Pseudoinverse zu $\mathfrak{X}$? Was bedeutet in dem Kontext eines Tensors überhaupt Pseudoinverse?

Die Eigenschaften der Moore Penrose Pseudoinverse für Matrizen lautet

Lemma 3.1. *(Moore Penrose Pseudoinverse)*

1. $AA^+A = A$
2. $A^+AA^+ = A^+$
3. $(AA^+)^T = AA^+$
4. $(A^+A)^T = A^+A$

Jetzt gilt es diese Eigenschaften auf Tensoren zu übertragen. Da wir vorerst keine Tensor-Tensor Multiplikation haben, gilt es diese zu definieren. Diese Tensor-Tensor Multiplikation macht dann nur für unsere Anwendung einen Sinn.

Dafür müssen wir zuerst unsere Tensoren herleiten. Dies geschieht mit Hilfe der Tensorstruktur der Ansatzfunktionen. Wir definieren den *Massentensor* elementweise durch

$$M_{i1,i2,j1,j2} = \int_T \varphi_{i1}(x_1)\varphi_{i2}(x_2)\varphi_{j1}(x_1)\varphi_{j2}(x_2)\, d(x_1, x_2) \tag{3.19}$$

und unseren *lokalen Laplace Tensor*, welcher das Pendant zur Elementsteifigkeitsmatrix der Laplace Bilinearform ist, wie folgt

$$V_{i1,i2,j1,j2} = \int_T \varphi'_{i1}(x_1)\varphi_{i2}(x_2)\varphi'_{j1}(x_1)\varphi_{j2}(x_2) + \varphi_{i1}(x_1)\varphi'_{i2}(x_2)\varphi_{j1}(x_1)\varphi'_{j2}(x_2)\, d(x_1, x_2). \tag{3.20}$$

Diese Transformation von Matrix zu Tensor ist somit eine Abbildung die einen Indextupel (i, j) eines Matrixelements auf den Indextupel eines Tensorelements (i_1, i_2, j_1, j_2) abbildet. Um eine Tensor-Tensor Multiplikation definieren zu können, muss uns diese Transformation klar sein. In dieser Transformation steckt implizit zweimal die gleiche Transformation. Nämlich

$$p : i \to (i_1, i_2),$$
$$p : j \to (j_1, j_2).$$

Diese Transformation zu definieren erfolgt durch intuitives Umformen und dem Hintergrundwissen zur lexikograpischen Ordnung der Freiheitsgrade.

Das Inverse der Transformation ist gegeben durch

$$p^{-1}(i_1, i_2) = i_1 + (N+1)i_2 = i\,, \tag{3.21}$$

wobei $N+1$ die lokalen Freiheitsgrade pro Dimension sind. Wie können wir aber gegeben i das korrespondiere Tupel (i_1, i_2) berechnen? Dazu nutzen wir die Modulo Rechnung. Hierzu nehmen wir das Inverse der Transformation $modulo\,(N+1)$.

$$i \ (mod(N+1)) = p^{-1}(i_1, i_2) \ (mod(N+1)) = i_1 + \underbrace{(N+1)i_2}_{0} \ (mod(N+1)) \tag{3.22}$$

Da $(N+1)i_2$ ein vielfaches von $(N+1)$ ist, ergibt dies 0. Da $i_1 < (N+1)$ folgt

$$i \ (mod(N+1)) = i_1 \ (mod(N+1)) = i_1. \tag{3.23}$$

Nun wissen wir, wie aus der Information i unser korrespondierendes i_1 extrahiert werden kann. Die Gleichung (3.21) können wir nach i_2 wie folgt umstellen

$$i_2 = \frac{i - i_1}{N+1}. \tag{3.24}$$

Mit dem Wissen über i_1 können wir dies weiter umformen zu

$$i_2 = \frac{i - (i \ (mod(N+1)))}{N+1}. \tag{3.25}$$

Damit haben wir unsere Transformation p

$$p(i) = \Big(i \ (mod(N+1)), \frac{i - (i \ (mod(N+1)))}{N+1}\Big) \tag{3.26}$$

gefunden und eindeutig festgelegt, welches Element der Matrixform zu welchem Element der Tensorform gehört. Durch die Definition der Transformationen können wir diese für unser Tensor-Tensor Produkt Definition zu Hilfe nehmen. Zuvor schauen wir uns das Matrix-Matrix Produkt als Motivation an.

Es sei $M \in \mathbb{R}^{N^2 \times N^2}$ die lokale Massenmatrix. Dann folgt für $MM = C \in \mathbb{R}^{N^2 \times N^2}$ die elementenweise Definition

$$C_{ik} = \sum_{j=1}^{N^2} M_{ij} M_{jk} \tag{3.27}$$

Nun nutzen wir unsere Index-Transformation, um die Matrixelemente als Tensorelemente umzudefinieren. Es sei weiterhin $p(i) = (i_1, i_2)$ und $p(k) = (k_1, k_2)$.

$$C_{p(i),p(k)} = C_{i_1,i_2,j_1,j_2} = \sum_{j=1}^{N^2} M_{p(i),p(j)} M_{p(j),p(k)} = \sum_{j_1=1}^{N} \sum_{j_2=1}^{N} M_{i_1,i_2,j_1,j_2} M_{j_1,j_2,k_1,k_2} \tag{3.28}$$

Damit haben wir eine Motivation für die Definition unseres Tensor-Tensor Produkts.

Definition 3.2. *(Tensor-Tensor Produkt)*
Es seien $\mathfrak{X}^1 \in \mathbb{R}^{I_1 \times I_2 \times I_1 \times I_2}$ *und* $\mathfrak{X}^2 \in \mathbb{R}^{I_1 \times I_2 \times I_1 \times I_2}$ *Tensoren. Dann definieren wir*

das Produkt dieser beiden Tensoren elemebtweise wie folgt

$$ttp(\mathfrak{X}^1, \mathfrak{X}^2)_{i_1,i_2,j_1,j_2} = \sum_{j_1=1}^{I_1} \sum_{j_2=1}^{I_2} \mathfrak{X}^1_{i_1,i_2,j_1,j_2} \mathfrak{X}^2_{j_1,j_2,k_1,k_2} \tag{3.29}$$

Die Komplexität betrachtend, ist das Tensor-Tensor-Produkt der *Massentensoren* beziehungsweise der *lokalen Laplace-Tensoren* genau so komplex wie das Produkt der korrespondieren Matrizen. Darauf wird in Kapitel 4 ausführlich eingegangen.

Es wird noch der Operator des Transponierens für Tensoren benötigt. Analog zum Tensor-Tensor-Produkt, schauen wir uns den Operator des Transponierens zuerst für Matrizen an. Sei $A \in \mathbb{R}^{N^2 \times N^2}$ eine beliebige Matrix, dann ist die transponierte Matrix gegeben durch

$$A^T_{ij} = A_{ji}. \tag{3.30}$$

Wir können die Index-Transformation nutzen, um den äquivalenten Operator für Tensoren zu erhalten. Dies bringt uns folgendes Ergebnis

$$A^T_{p(i)p(j)} = A^T_{i_1\, i_2\, i_1\, j_2} = A_{j_1\, j_2\, i_1\, i_2} = A_{p(j)\, p(i)}. \tag{3.31}$$

Wir können nun die Moore Penrose Pseudoinverse Eigenschaften auch für Tensoren angeben. Jedoch sollte zuvor das Problem der Maschinengenauigkeit angesprochen werden. Dazu gibt es einen Trick, der mit Bedacht eingesetzt werden muss. Die Gleichheit wie in Lemma (3.1) ist mit einem Rechner nicht zu erzielen, daher wird das Lemma abgeschwächt und für Tensoren angegeben.

Lemma 3.3. *(Moore Penrose Pseudoinverse für Tensoren)*

1. $ttp(A, ttp(A^+, A)) - A \quad < \epsilon$
2. $ttp(A^+, ttp(A, A^+)) - A^+ \quad < \epsilon$
3. $(ttp(A, A^+))^T - ttp(A, A^+) < \epsilon$
4. $(ttp(A^+, A))^T - ttp(A^+, A) < \epsilon$

Die Wahl des Epsilons ist hier entscheidend. Man könnte Maschinengenauigkeit wählen, was jedoch in für unseren Zweck sinnlos ist. 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. An diesem Punkt wissen wir, wie wir einen Tensor als Pseudoinverse klassifizieren können. Wie errechnet sich jedoch die Pseudoinverse?

Aus der HOSVD ergibt sich die Zerlegung für einen Tensor $\mathfrak{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3 \times I_4}$ mit

$$\mathfrak{X} = \mathfrak{G} \times_{n=1}^{4} U^{(n)}. \tag{3.32}$$

Nun nehmen wir die Pseudoinverse von beiden Seiten. Dies ist möglich, da mittlerweile bekannt ist, was es bedeutet die Pseudoinverse von einem Tensor zu haben. Wir erhalten

$$\mathfrak{X}^{+} = (\mathfrak{G} \times_{n=1}^{4} U^{(n)})^{+}. \tag{3.33}$$

Sei $\mathcal{G}$ super-diagonal und $U^{(n)} = U^{(i)}$ für alle $i, n \in \{1, \ldots, 4\}$. Den Pseudoinversen Operator können wir hineinziehen. Ob das Hineinziehen des Operators erlaubt ist, wurde formal nicht bewiesen. In meiner Arbeit habe ich dies experimentell nachgewiesen. Die Probleme mit dem Beweis dieser Aussage sind fehlenden Resultate für das Kommunikationsverhalten verschiedener Operatoren. Wir arbeiten hierbei mit dem $n-mode$ Produkt, Entfaltungen und Matrix Produkten.

Man könnte folgenden Beweisansatz nutzen:
Es sei $\mathcal{X} \in \mathbb{R}^{I \times I \times I \times I}$ ein Tensor der Ordnung 4 und $U \in \mathbb{R}^{I \times I}$ eine orthogonale Matrix. Dann gilt es für $n \in \{1, \ldots, 4\}$ zu zeigen:

$$(\mathcal{X} \times_n U)^{+} = \mathcal{X}^{+} \times_n U^{T}$$

Wir nutzen die Äquivalenz des $n-mode$ Produktes mit dem Matrix Produkt mit dem entfalteten Tensor.

$$(\mathcal{X} \times_n U)^{+}_{(n)} = (U\mathcal{X}_{(n)})^{+} = \mathcal{X}^{+}_{(n)} U^{+}$$

Der letzte Schritt können ist möglich, da U orthogonal ist. Es fehlt noch der Tausch von beiden Matrizen, das heißt wir brauchen Kommutativität. Dies ist allerdings nicht möglich, denn $\mathcal{X}^{+}_{(n)} \in \mathbb{R}^{I^3 \times I}$ und $U^{+} \in \mathbb{R}^{I \times I}$. Also wäre die Matrixmultiplikation nicht wohldefiniert. Das liefert uns jedoch keinen Widerspruch zu unserem Ergebnis, da gilt:

$$(\mathcal{X}^{+})_{(n)} \neq (\mathcal{X}_{(n)})^{+},$$

denn links berechnen wir die Pseudoinverse des Tensors. Wie dessen Pseudoinverse davon definiert ist, haben wir für unsere Anwendung definiert und dies ist auch nur für unsere Anwendung sinnvoll. Rechts nehmen wir die Pseudoinverse einer Matrix.

Wir haben also zwei verschiedene Pseudoinversen Operatoren.

Das Problem ist, sollte man mit dem entfalteten Tensor rechnen, dies eher eine veranschaulichte Darstellung ist und wenig Nützlichkeit birgt.

Doch wenn wir nicht mit den entfalteten Tensoren rechnen würden, bräuchten wir Resultate über die Invertierung von $n-mode$ Produkten, die nicht vorhanden sind. Daher nehmen wir an, dass das Ergebnis stimmt und rechnen weiter. Wir bekommen nun folgende Darstellung der Pseudoinversen

$$\mathcal{X}^{+} = \mathcal{G}^{+} \times_{n=1}^{4} U^{(n)^{+}}. \tag{3.34}$$

Da die Faktormatrizen $U^{(n)}$ orthogonal sind, reicht es, die Transponierte zu verwenden.

$$\mathcal{X}^{+} = \mathcal{G}^{+} \times_{n=1}^{4} U^{(n)^{T}} \tag{3.35}$$

Das Invertieren des Kerntensors erweist sich alllerdings als problematisch. Hierfür ist es nützlich, die Struktur des Kerntensors zu kennen. Der Kerntensor ist leider in den meisten Fällen voll besetzt. Genaueres Hinsehen zeigt zwei Arten von Zahlen. Relativ große Zahlen von größer als 1 und relativ kleine Zahlen von kleiner als 10^{-10}. Die kleinen Zahlen sind in diesem Fall unbrauchbar und beinhalten wenig Informationen. Doch das Eliminieren vieler kleiner Zahlen nimmt uns in der Summe möglicherweise relevante Informationen. Daher sollte man vorsichtig bei diesem Verfahren sein. Die Grenze, die entscheidet, welche Zahl klein genug ist um eliminiert zu werden, sollte mit Bedacht gewählt werden. Je größer die Grenze ist umso größer muss das ϵ bei der Definition der Tensor-Pseudoinversen, in Lemma (3.3), gewählt werden.

Wir können folglich kleine Zahlen einfach ausradieren und erhalten dadurch einen super-diagonalen Tensor. Die Invertierung des Tensors beschränkt sich darauf, jedes Diagonalelement zu invertieren.

Wir wissen nun wie wir unsere Tensoren berechnen können und wie sich die Pseudoinverse mittels Singulärwertzerlegung höherer Ordnung gewinnen lässt .
Im folgenden Kapitel wird die effiziente Berechnung der Pseudoinversen dargestellt.

4 Effiziente Implementierung und Komplexitätsanalyse

In Kapitel 3 haben wir uns zwei Möglichkeiten angeschaut die Pseudoinversen herzuleiten. In beiden Möglichkeiten fand sich das Kronecker-Produkt. Um dieses effizient implementieren zu können, sollten wir uns Gedanken darüber machen, wie diese Struktur nutzbar ist.

In [Tea] wird die effektive Berechnung der Massenmatrix mit einem Vektor multipliziert, vorgestellt. Dafür wird die Tensorstruktur der Massenmatrix ausgenutzt. Diese Methodik sollten wir uns kurz vor Augen führen und daraufaufbauend, um einige eigene Gedanken erweitern, um sie für unsere Anwendung zugänglich zu machen.

4.1 Effizientes Matrix-Vektor Produkt

In [Tea] wird eine Strategie vorgestellt, um ein Matrix-Vektor Produkt mit Kronecker Produkt Matrizen $z = (\mathcal{B} \otimes \mathcal{A})y$ effektiv zu berechnen.

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

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

Wir sehen sich wiederholende Strukturen von $\mathcal{A}$. Genau diese wollen wir uns zunutze 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 die Faktoren b_{ij}, die mit $\mathcal{A}$ multipliziert werden, 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 $y^{(2)} = (y_{n+1}, \dots, y_{2n})^T$ definieren und

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

Wir führen dies weiter und erhalten

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

Es müssen die Informationen der Matrix $\mathcal{B}$ noch eingebracht werden. Dazu berechnen wir wie in [Tea] vorgeschlagen z_i mit

$$z_j^{(k)} = \sum_{i=1}^{q} b_{ki} w_j^{(i)}$$

Der komplette Algorithmus ist nachfolgend dargestellt.

for i=1 < q **do**
  for j= 1 < m **do**
    $w_j^{(i)} = y_1^{(i)} a_{j1} + \cdots + y_n^{(i)} a_{jn}$
  end for
end for
for k=1 < n **do**
  for j= 1 < p **do**
    $z_j^{(k)} := w_j^{(1)} b_{k1} + \cdots + w_j^{(q)} b_{kq}$
  end for
end for

Nehmen wir für die triviale Berechnung der Komplexität an $n = m = p = q$. Mit Hilfe dieses Algorithmus haben wir die Komplexität des Auswertens der Gleichung von $2m^4$ Operationen auf $4m^3$ Operationen reduziert.

Dieser Algorithmus ist optimal für das effektive Berechnen von

$$v = M^+ u = [\mathcal{W}_N^{1D} (\mathcal{N}^{1D})^T]^+ \otimes [\mathcal{W}_N^{1D} (\mathcal{N}_{1D})^T]^+ u \,. \tag{4.1}$$

Den Algorithmus kann man ebenfalls als Matrix-Produkt darstellen. Dazu ist es notwendig, die Vektoren u und v in Matrizen zu überführen. Dies erfolgt über unsere Index-Transformation $p(\cdot)$. Seien V die zu v, und U die zu u korrespondierenden Matrizen. Dann kann man (4.1) darstellen als

$$V = [\mathcal{W}_N^{1D} (\mathcal{N}^{1D})^T]^+ U ([\mathcal{W}_N^{1D} (\mathcal{N}_{1D})^T]^+)^T. \tag{4.2}$$

Diese Überführung wird später für die Berechnung der Laplace Bilinearform benötigt.

Für den zweiten Ansatz über die HOSVD werden wir eine erweiterte Form des Algorithmus benötigen, da wir mehrere Kronecker Produkte erhalten. Außerdem kann der erweiterte Algorithmus dann auch für dreidimensionale Ansatzfunktionen beim ersten Ansatz verwendet werden.

4.1.1 Erweiterung

Wir wollen den Algorithmus erweitern für die Berechnung von $z = (\mathcal{C} \otimes \mathcal{B} \otimes \mathcal{A})v$ mit $\mathcal{A} \in \mathbb{R}^{N \times N}$, $\mathcal{B} \in \mathbb{R}^{N \times N}$, $\mathcal{C} \in \mathbb{R}^{N \times N}$ und $v \in \mathbb{R}^{N^3}$.

$$z = \begin{pmatrix} c_{11}b_{11}\mathcal{A} & \dots & c_{11}b_{1N}\mathcal{A} & \dots & \dots & c_{1N}b_{11}\mathcal{A} & \dots & c_{1N}b_{1N}\mathcal{A} \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \vdots \\ c_{11}b_{N1}\mathcal{A} & \dots & c_{11}b_{NN}\mathcal{A} & \dots & \dots & c_{1N}b_{N1}\mathcal{A} & \dots & c_{1N}b_{NN}\mathcal{A} \\ c_{21}b_{11}\mathcal{A} & \dots & c_{21}b_{1N}\mathcal{A} & \dots & \dots & c_{2N}b_{11}\mathcal{A} & \dots & c_{2N}b_{1N}\mathcal{A} \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \vdots \\ c_{N1}b_{N1}\mathcal{A} & \dots & c_{N1}b_{NN}\mathcal{A} & \dots & \dots & c_{NN}b_{N1}\mathcal{A} & \dots & c_{NN}b_{NN}\mathcal{A} \end{pmatrix} v \quad (4.3)$$

Wir sehen hier zwei sich wiederholende Strukturen, die wir ausnutzen wollen um Operationen zu sparen.

$$z = \begin{pmatrix} c_{11}b_{11}\mathcal{A} & \dots & c_{11}b_{1N}\mathcal{A} & \dots & \dots & c_{1N}b_{11}\mathcal{A} & \dots & c_{1N}b_{1N}\mathcal{A} \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \vdots \\ c_{11}b_{N1}\mathcal{A} & \dots & c_{11}b_{NN}\mathcal{A} & \dots & \dots & c_{1N}b_{N1}\mathcal{A} & \dots & c_{1N}b_{NN}\mathcal{A} \\ c_{21}b_{11}\mathcal{A} & \dots & c_{21}b_{1N}\mathcal{A} & \dots & \dots & c_{2N}b_{11}\mathcal{A} & \dots & c_{2N}b_{1N}\mathcal{A} \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \vdots \\ c_{N1}b_{N1}\mathcal{A} & \dots & c_{N1}b_{NN}\mathcal{A} & \dots & \dots & c_{NN}b_{N1}\mathcal{A} & \dots & c_{NN}b_{NN}\mathcal{A} \end{pmatrix} v \quad (4.4)$$

Den Vektor v können wir zu einem Tensor $\mathcal{V} \in \mathbb{R}^{N \times N \times N}$ umdefinieren, damit dieser handlicher für unsere Anwendung wird. Der erste Index repräsentiert, in welchem Spalteneintrag von $\mathcal{C}$, der zweite Index, in welchem Spalteneintrag von $\mathcal{B}$ und der dritte Index, in welchem Spalteneintrag von $\mathcal{A}$ wir uns befinden. Dies kann man als Index-Transformation ansehen, die bestimmte Einträge von v eindeutig auf Tensorelemente abbildet. Wir schauen uns zuerst die einzelnen Einträge von z an.

Dadurch können wir den ersten Eintrag von z darstellen durch

$$z_1 = \mathcal{V}(1,1,1)c_{11}b_{11}a_{11} + \cdots + \mathcal{V}(1,1,N)c_{11}b_{11}a_{1N} + \cdots + \mathcal{V}(1,N,1)c_{11}b_{1N}a_{11} \\ + \cdots + \mathcal{V}(N,1,1)c_{1N}b_{11}a_{11} + \cdots + \mathcal{V}(N,N,N)c_{1N}b_{1N}a_{1N}.$$

Definiere

$$w_1(i,j) := \mathcal{V}(i,j,1)a_{11} + \cdots + \mathcal{V}(i,j,N)a_{1N}.$$

Dann erhalten wir

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

Damit haben wir uns die sich wiederholende Struktur von der Matrix $\mathcal{A}$ zunutze gemacht. Im nächsten Schritt machen wir uns die sich wiederholende Struktur von $\mathcal{B}$ zunutze. Wir definieren hierfür

$$\mathcal{W}_{1,k}(i) := w_k(i,1)b_{11} + \cdots + w_k(i,N)b_{1N}.$$

Damit erhalten wir

$$z_1 = \mathcal{W}_{1,1}(1)c_{11} + \cdots + \mathcal{W}_{1,1}(N)c_{1N}.$$

Wir formen z genau so wie wir auch für v verfahren sind und erhalten dafür den Tensor $\mathcal{Z}$. Damit erhalten wir für einen beliebigen Eintrag von z folgende Formel

$$\mathcal{Z}(i,j,k) = \mathcal{W}_{j,k}(1)c_{i1} + \cdots + \mathcal{W}_{j,k}(N)c_{iN}\,,$$

wobei j und k den Zeilen jeweils in den Matrizen $\mathcal{B}$ und $\mathcal{C}$ entsprechen.
Der komplette Algorithmus ist nachfolgend dargestellt.

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**
      $\mathcal{W}_{i,j}(k) := w_j(k,1)b_{i1} + \cdots + w_j(k,N)b_{iN}$
    end for
  end for
end for
for k=1 < N **do**
  for i= 1 < N **do**
    for j= 1 < N **do**
      $\mathcal{Z}(i,j,k) = \mathcal{W}_{j,k}(1)c_{i1} + \cdots + \mathcal{W}_{j,k}(N)c_{iN}$
    end for
  end for
end for

Wir haben in (4.4) eine Matrix-Vektor Multiplikation von einer Matrix der Größe $N^3 \times N^3$. Dementsprechend hätten wir N^6 Multiplikationen und N^6 Additionen. Dies entspricht $2N^6$ elementaren Operationen. Die Komplexität des vorgeschlagenen Algorithmus reduziert die Operationen auf $3N^4$ Multiplikationen und genauso viele Additionen. Somit haben wir insgesamt $6N^4$ Operationen. Das ist, vor allem für großes N, eine beträchtliche Reduktion.

4.2 Tensorstruktur

Wie können wir diese Algorithmen für die in Kapitel 3 angesprochenen Strategien verwenden?

In diesem Unterkapitel schauen wir uns die Anwendung der hergeleiteten Algorithmen auf die Tensorstruktur unserer Bilinearformen an. Zudem wollen wir eine Komplexitätsanalyse für jeden der Ansätze angeben. Zuerst schauen wir uns das Matrix-Vektor Produkt mit der Pseudoinversen der Massenmatrix und einem beliebigen Vektor u an. Danach kommen wir zur Lösung des Problems mit der Laplace Bilinearform.

Massenmatrix

$$M^+u = [(\mathcal{W}_N^{1D}(\mathcal{N}^{1D})^T)^+ \otimes (\mathcal{W}_N^{1D}(\mathcal{N}_{1D})^T)^+]u.$$

Wie komplex ist es, die Pseudoinverse der Matrix $(\mathcal{W}_N^{1D}(\mathcal{N}^{1D})^T) \in \mathbb{R}^{N\times N}$ zu berechnen, wobei N die Anzahl der Freiheitsgrade in einer Dimension ist.

Wir können als Ansatz eine Singulärwertzerlegung wählen, aus der zuvor Pseudoinverse hergeleitet werden muss. Allgemein gilt

$$M \;=\; U\Sigma V^T$$

für eine eine $N \times N$-Matrix M mit Rang r, wobei U eine orthogonale $N \times N$-Matrix, V^T die Transponierte einer orthogonalen $N \times N$-Matrix V und Σ eine reelle $N \times N$-Diagonalmatrix ist.

Die daraus hergeleitete Pseudoinverse ergibt

$$M^+ = V\Sigma^+U^T.$$

Die Komplexität für die Herleitung der Pseudoinversen aus der Singulärwertzerlegung ist vernachlässigbar. Für die Berechnung von Σ^+ haben wir r Operationen, da in Σ in der Diagonalen r Einträge stehen, die wir nur invertieren müssen. Ebenfalls ist die Berechnung von U^T ist vernachlässigbar, da man bei der Berechnung der Singulärwertzerlegung direkt U^T speichern kann anstatt U. Somit bekommen wir mit der Singulärwertzerlegung einer Matrix $M \in \mathbb{R}^{N\times N}$ die Pseudoinverse mit einer Komplexität von $O(N^3)$ [AF09, 2]. Man könnte an dieser Stelle auch approximative Verfahren wählen und versuchen an Operationen zu sparen. Ich verweise hier auf [AF09] für die nähere Betrachtung von schnelleren Singulärwertzerlegungen.

Für die Berechnung der Inversen kann man mit dem Gauss Algorithmus. Der Gauss Algorithmus gibt uns eine Komplexität von $O(N^3)$. Der Gauss Algorithmus ergibt entsprechend einen Sinn, wenn $(\mathcal{W}_N^{1D}(\mathcal{N}^{1D})^T)$ invertierbar ist.

Bemerkung 4.1. *(Invertierbarkeit)*
Es seien $\mathcal{W}_N^{1D} \in \mathbb{R}^{N\times N}$ und $(\mathcal{N}^{1D})^T \in \mathbb{R}^{N\times N}$ Matrizen, die wie gewohnt definiert sind. Dann gilt, dass $(\mathcal{W}_N^{1D}(\mathcal{N}^{1D})^T) \in \mathbb{R}^{N\times N}$ invertierbar ist.

Beweis. Man kann durch einfache Basistransformation erreichen,
dass $(\mathcal{W}_N^{1D}(\mathcal{N}^{1D})^T) \in \mathbb{R}^{n\times n}$ die Einheitsmatrix ist. Es seien $x = \{x_1, \ldots, x_N\}$ die Stützstellen und $w = \{w_1, \ldots, w_N\}$ die Gewichte der Quadratur.

Wir wählen

$$\varphi_i^{1D}(x_k) = \frac{1}{\sqrt{w_i}} l_i(x_k) = \begin{cases} \frac{1}{\sqrt{w_i}} & \text{, wenn } i = k \\ 0 & \text{, sonst} \end{cases}$$

als neue Basis.

Die Funktion $l_i(x_k)$ bezeichnet das $i-te$ Lagrange Polynom ist. Es folgt

$$\mathcal{W}_N^{1D}(\mathcal{N}^{1D})^T = \mathcal{N}^{1D}\mathcal{W}^{1D}(\mathcal{N}^{1D})^T = I_N.$$

Da $\mathcal{N}^{1D}$ eine Diagonalmatrix ist, mit den Diagonaleinträgen $\frac{1}{\sqrt{w_i}}$, gilt

$$\mathcal{N}^{1D} = (\mathcal{N}^{1D})^T.$$

Weiterhin ist $\mathcal{W}^{1D}$ auch eine Diagonalmatrix mit den Einträgen w_i in der Diagonalen. In der Ergebnismatrix steht folglich in der Diagonalen

$$(\frac{1}{\sqrt{w_i}})^2 w_i = 1\,.$$

□

Insgesamt erhalten wir für die Berechnung der Inversen beziehungsweise der Pseudoinversen folgende Komplexität:

- Um das Matrix-Vektor Produkt effizient zu berechnen nutzen wir den Algorithmus für die Berechnung von $z = (\mathcal{B} \otimes \mathcal{A})y$ aus Kapitel 4.1. und erhalten eine Komplexität von $4N^3$.
- Matrizenmultiplikation ist kubisch.
- Berechnung der Pseudoinversen/Inversen liegt in $O(N^3)$.

Insgesamt haben wir eine Komplexität von $O(4N^3) + O(N^3) + O(N^3) = O(6N^3)$. Wie weitergehend an Komplexität gespart werden kann und wo wir anknüpfen können, um effizienter zu werden, wird in Kapitel 5 diskutiert. Wir kommen zu dem Problem mit der Laplace Bilinearform. **Laplace Bilinearform**

$$y = V^+u = \left([(\widehat{\mathcal{W}}_N^{1D}(\widehat{\mathcal{N}}^{1D})^T] \otimes [\mathcal{W}_N^{1D}(\mathcal{N}^{1D})^T] + [\mathcal{W}_N^{1D}(\mathcal{N}^{1D})^T] \otimes [\widehat{\mathcal{W}}_N^{1D}(\widehat{\mathcal{N}}^{1D})^T]\right)^+ u$$

Wir wollen dies weiter vereinfachen. Definiere $A = [(\widehat{\mathcal{W}}_N^{1D}(\widehat{\mathcal{N}}^{1D})^T]$. Wähle Basis so, dass $\mathcal{W}_N^{1D}(\mathcal{N}^{1D})^T = I_N$. Dies ist, wie in Beweis für Bemerkung (4.1) möglich.

Wir erhalten folgende vereinfachte Form.

$$Vu = [(A \otimes I_) + (I_n \otimes A)]u.$$

Mit der in Kapital 4.1 besprochenen Strategie, können wir den Vektoren u und y in Matrizen überführen, sodass wir (4.2) äquivalent umformen können zu

$$Y = V^+U = [(A \otimes I_N) + (I_N \otimes A)]U,$$

wobei U und Y definiert sind durch $U_{ij} = u(q_i, q_j)$ und $Y_{ij} = Y(q_i, q_j)$.

Dies können wir weiter vereinfachen.

$$V = (A \otimes I_N)U + (I_N \otimes A)U = AUI_n + I_nUA^T = AU + UA^T$$

Dies ist die Lyapunow Gleichung aus der Stabilitätstheorie. Zur Lösung der Lyapunow Gleichung gibt es zahlreiche Untersuchungen. Die Behandlung dieser Gleichung würde den Rahmen dieser Bachelorarbeit sprengen. Ich empfehle als Literatur für diese Untersuchungen [Mik09].

Die Lösung der Gleichung ist gegeben durch

$$U = \int_0^\infty e^{A\tau}(-V)e^{A^T\tau}d\tau.$$

Die naive Berechnung der Lösung dieser Gleichung erfolgt mit einer Komplexität von $O(N^6)$. Der Bartels-Stewart Algorithmus [uYZ03] liefert uns eine Komplexität von $O(N^3)$.

4.3 Singulärwertzerlegung höherer Ordnung

Nun haben wir uns mit Hilfe der HOSVD eine Herleitung für die Pseudoinverse erarbeitet. Folgend geht es um die effiziente Berechnung dieser Formel. Wir können uns zwei verschiedene Ansätze mit der HOSVD anschauen. Den ersten Ansatz nennen wir den naiven Ansatz und dieser erfolgt über die einfache Form der HOSVD mit dem $n-mode$ Produkt. Im zweiten Ansatz schauen wir uns die entfaltete HOSVD an. Dort erkennen wir eine Kronecker Produkt Struktur. Dazu wollen wir uns die in Kapitel 4.1.1. hergeleitete Strategie zunutze machen, für die effektive Berechnung von den zwei Kronecker Produkten mit einem Vektor. Es sei $\mathcal{X} \in \mathbb{R}^{I_1\times\cdots\times I_4}$ ein Tensor. Der Tensor $\mathcal{X}$ könnte der Massetensor sein oder der Laplace Bilinearform Tensor. Da die Massematrix und die Elementsteifigkeitsmatrix quadratisch sind, gilt $N := I_1 = \cdots = I_4$. Die Formel für die Pseudoinverse lautet

$$\mathcal{X}^+ = \mathcal{G}^+ \times_{n=1}^4 U^{(n)^T}, \tag{4.5}$$

wobei $\mathcal{G} \in \mathbb{R}^{N\times N\times N\times N}$ und $U^{(n)} \in \mathbb{R}^{N\times N}$.

Wie aufwändig ist es, die HOSVD zu berechnen? In Bemerkung (2.19) haben wir den Algorithmus für die Berechnung der HOSVD angegeben.

1. Für den ersten Schritt des Algorithmus müssen wir alle $mode-k$ Entfaltungen berechnen. Dies impliziert eine Ordnung jedes Elements des Tensors in ein Element der Ergebnismatrix. Damit haben wir genau so viele Zuweisungen, wie die Anzahl der Elemente des Tensors. In unserem Fall wären das $4N^4$ Operationen für alle modes.

2. Die Singulärwertzerlegung für die Entfaltungen gilt es im zweiten Schritt zu berechnen. Eine Entfaltung ist eine Matrix der Größe $N \times N^3$. Dementsprechend würde eine Singulärwertzerlegung für diese Matrizen in der Komplexitätsklas-

se $O(N^5)$ liegen. Wir müssen dies für vier Matrizen durchführen und erhalten $O(4N^5)$.

3. Im dritten Schritt wollen wir den Kerntensor berechnen. Dazu müssen wir $n-mode$ Produkte berechnen. Ein $n-mode$ Produkt hat N^5 Additionen und genauso viele Multiplikationen. Davon müssen wir vier berechnen. Insgesamt haben wir eine Komplexität von $8N^5$ Rechenoperationen.

Insgesamt ergibt sich eine Komplexität von $O(4N^4 + 12N^5)$. Die Berechnung der orthogonal Matrizen für die HOSVD zeigt, dass alle Matrizen gleich sind. Dementsprechend müssen wir im zweiten Schritt die Singulärwertzerlegung nur einmal berechnen. Außerdem müssen wir nur eine Entfaltung im ersten Schritt rechnen. Damit reduziert sich die Komplexität zu $O(N^4 + 9N^5)$.

Wir wollen aus der HOSVD nun die Pseudoinverse errechnen. Dies geschieht, indem wir die Transponierte der Faktormatrizen ausrechnen und den Kerntensor invertieren. Die Faktormatrizen können wir direkt in transponierter Form speichern. Um den Kerntensor zu invertieren müssen wir durch den Tensor iterieren und alle Elemente kleiner ϵ auf 0 setzen und alle Elemente größer oder gleich ϵ invertieren. Dies ergibt eine Komplexität von $O(N^4)$, da der Tensor von Ordnung 4 ist.

Im nächsten Schritt wollen wir einen beliebigen Vektor $u \in \mathbb{R}^{N^3}$ an die HOSVD multiplizieren.

4.3.1 Naiver Ansatz

Wir schauen uns zuerst den naiven Ansatz an. Dafür benötigen wir jedoch zusätzlich ein geeignetes Tensor-Vektor Produkt. Dazu schauen wir uns das Matrix-Vektor Produkt an. Sei $A \in \mathbb{R}^{m \times n}$ und $x \in \mathbb{R}^n$. Dann ist $y = Ax \in \mathbb{R}^m$ definiert durch

$$y_i = \sum_{j=1}^{n} a_{ij} x_j.$$

Durch die Index-Transformation $p(\cdot)$ können wir dies für Tensoren herleiten.

Definition 4.2. *(Tensor-Vektor Produkt)*
Es sei $\mathcal{A} \in \mathbb{R}^{N \times N \times N \times N}$ und $U \in \mathbb{R}^{N \times N}$. Dann ist $Y = tvp(\mathcal{A}, U)$ gegeben durch

$$Y_{i_1\, i_2} = \sum_{j_1=1}^{N} \sum_{j_2=1}^{N} \mathcal{A}_{i_1 i_2 j_1 j_2} U_{j_1 j_2}$$

Nun können wir uns den naiven Ansatz anschauen, der wie folgt gegeben ist

$$tvp(\mathcal{X}^+, U) = tvp(\mathcal{G}^+ \times_{n=1}^{4} U^{(n)^T}, U), \tag{4.6}$$

mit $U \in \mathbb{R}^{N \times N}$. Das ist das Matrix-Vektor Produkt in Tensorform.

Die Komplexität für die Berechnung ist gegeben durch N^4. Denn wir haben für die Berechnung eines einzelnen Elementes $Y_{i_1 i_2}$ zwei Schleifen die bis N durchlaufen. Insgesamt haben wir N^2 Elemente. Dies ergibt N^4 Operationen.
Mit der Berechnung der HOSVD, das Invertieren des Kerntensors und die Berechnung der $n-mode$ Produkte ergibt dieser Ansatz eine Komplexität von $O(3N^4 + 17N^5)$.

4.3.2 Entfaltete HOSVD

Wir wollen nun einen anderen Ansatz wählen und nutzen, dass wir die HOSVD in entfalteter Form betrachten können. Man kann (4.6) wie in (2.24) äquivalent umformen zu

$$\begin{aligned} & \mathcal{X}^+_{(n)} = U^{(n)^T} \mathcal{G}^+_{(n)} (U^{(4)^T} \otimes \cdots \otimes U^{(n+1)^T} \otimes U^{(n-1)^T} \otimes \cdots \otimes U^{(1)^T})^T \\ \iff & \mathcal{X}^+_{(n)} = U^{(n)} \mathcal{G}^+_{(n)} (U^{(4)} \otimes \cdots \otimes U^{(n+1)} \otimes U^{(n-1)} \otimes \cdots \otimes U^{(1)}). \end{aligned} \tag{4.7}$$

Die Äquivalenz folgt mit Lemma (2.23)

$$\mathcal{X}^+_{(n)} v = U^{(n)} \mathcal{G}^+_{(n)} (U^{(4)} \otimes \cdots \otimes U^{(n+1)} \otimes U^{(n-1)} \otimes \cdots \otimes U^{(1)}) u. \tag{4.8}$$

Wir führen die Variablen N_i ein mit $N_i \neq n$. Damit können wir (4.6) reduzieren auf

$$\mathcal{X}^+_{(n)} u = \underbrace{\underbrace{U^{(n)} \mathcal{G}^+_{(n)}}_{K_1} \underbrace{(U^{(N_1)} \otimes U^{(N_2)} \otimes U^{(N_3)}) u}_{K_2}}_{K_3}. \tag{4.9}$$

Wir erkennen, dass wir die Methodik nutzen können, die wir entwickelt haben um K_2 effizient zu berechnen. Dies ist möglich mit einer Komplexität von $O(6N^4)$.

Das Invertieren des Kerntensors ist wie bereits veranschaulicht von der Komplexität $O(N^4)$. Es folgt eine letzte Matrix-Matrix Multiplikation $U^{(n)^T} \mathcal{G}^+_{(n)}$. Hierbei hat die Matrix $\mathcal{G}^+_{(n)}$ die Größe $N \times N^3$ und die Matrix $U^{(n)^T}$ die Größe $N \times N$. Wir ha-

ben bei der Multiplikation dieser beiden Matrizen N^5 Multiplikationen und genauso viele Additionen. Dann können wir die Komplexitätsklassen durch $O(N^5)$ angeben, doch dank der Struktur des Kerntensors reduziert sich die Komplexität auf $O(N^4)$. Das liegt daran, dass der Tensor $\mathcal{G}^+$ super-diagonal ist. Damit müssen wir also pro Element der Ergebnismatrix nicht die Spalte mal Zeile rechnen, sondern einfach nur das Diagonalelement mal ein Element der Matrix.

Nun gilt es das Ergebnis von K_1 und das Ergebnis von K_2 durch ein Matrix-Matrix Produkt zusammenzurechnen. Die Matrix K_1 hat die Größe $N \times N^3$ und K_2 ist $N^3 \times N^3$ groß. Die Anzahl der Operationen für die Berechnung von K_3 liegt also in $O(2N^5)$.

Insgesamt haben wir

- $O(N^4)$ für die Berechnung von K_1
- $O(6N^4)$ für die Berechnung von K_2
- $O(2N^5)$ für die Berechnung von K_3

Insgesamt ergibt sich eine Belastung von $O(9N^4 + 11N^5) \in O(N^5)$.
Dieser Ansatz ist zwar effizienter als der Naive, aber nicht annähernd so gut wie der Tensorstruktur Ansatz. Wie können wir unsere Effizienz weiter steigern?

Wir können uns eine trunkierte Singulärwertzerlegung höherer Ordnung wagen oder eine andere Alternative finden. Folgende Möglichkeiten zeigen sich.

1. Anstatt den Kerntensor zu modifizieren, können wir bei der Matrix-Matrix Multiplikation nur die Diagonalelemente für die Multiplikation in Erwägung ziehen. Damit sparen wir $O(N^4)$ Operationen.

2. Man kann den Kerntensor in K_2 vor dem Ausrechnen des Kronecker-Produkts verwerten. Selbst wenn wir dies erreichen, können wir die Komplexität von Option 1 nicht übertreffen, denn die Berechnung von K_3 würde dies nicht zulassen.

3. Eine letzte Alternative ist die trunkierte HOSVD.

Wenn wir es schaffen dies effizient zu berechnen und sogar eine trunkierte HOSVD wählen, ist das Ergebnis nicht zielführend. Durch die trunkierte HOSVD geben wir wertvolle Informationen auf, bei der Entfaltung der Tensoren treten Schwierigkeiten auf und die Komplexität ist nicht annähernd so gut wie mit der Nutzung der Tensorstruktur. Bei der HOSVD ist die alleinige Berechnung der Zerlegung mit einem

hohen Aufwand verbunden. Das $n-mode$ Produkt liegt in $O(N^5)$. Es ist außerdem nicht möglich, diesen zu umgehen oder zu optimieren, da wir in unserem Fall voll besetzte Tensoren haben.

Aus diesen Gründen würde ich diesen Ansatz nicht empfehlen.

5 Resultate

Wir haben uns für die Berechnung der Pseudoinverse und das Produkt der Pseudoinversen mit einem beliebigen Vektor zwei Alternativen angeschaut. Die Erste nutzt die Tensorstruktur der gegebenen Bilinearform, während die Zweite die Matrizen der Bilinearform in Tensoren umdefiniert und dann eine Singulärwertzerlegung höherer Ordnung durchführt.

Es ist naheliegend, dass man den Ansatz der Singulärwertzerlegung wählt, da dieser universell ist. Wir brauchen kein Vorwissen über unsere Bilinearform, insbesondere keine strukturspezifischen Informationen. Doch die Arbeit hat gezeigt, dass dieser Ansatz mit einer so hohen Komplexität verbunden ist, dass er nicht lohnenswert ist.

Der erste Ansatz versucht, die Elementsteifigkeitsmatrizen zuerst in eine Tensorstruktur zu überführen. Dieser Ansatz ist nicht universell und muss für jede Bilinearform extra hergeleitet werden. Mit Hilfe eines Algorithmus für die effiziente Berechnung eines Matrix-Vektor Produkts mit einer, Matrix die eine Tensorstruktur hat, ist es möglich eine sehr niedrige Komplexität zu erzielen. Wir sprechen hier von einer kubischen Ordnung abhängig vom Polynomgrad. Im Vergleich dazu hat der zweite Ansatz die Komplexitätsklasse $O(N^5)$, wobei N der Polynomgrad ist.

Zudem braucht der Ansatz der HOSVD die Definition und Speicherung der Elementsteifigkeitsmatrix der korrespondierenden Bilinearform, was wir versuchen sollten zu vermeiden. Der Grund liegt nicht in der Ersparnis von Speicherplatz, sondern darin, dass das Abrufen von Elementen dieser Elementsteifigkeitsmatrix, wenn diese zu groß für Ablage im Cache ist, mit einem überproportionalen Zeitaufwand verbunden ist.

Auf die Ergebnisse dieser Arbeit aufbauend, kann man versuchen im ersten Ansatz die Komplexität noch weiter zu senken. Dies kann man über eine *trunkierte* oder andere Formen der Singulärwertzerlegung probieren. Man kann eine effiziente Matrix-Matrix Multiplikation, wie zum Beispiel mit dem *Strassen-Algorithmus* nutzen. Für die Laplace Bilinearform kann man sich effizientere Algorithmen für die Berechnung der Lyapunow Gleichung anschauen.

Literatur

[AF09] Santosh Vempala Alan Frieze, Ravi Kannan. *Fast Monte-Carlo Algorithms for finding Low-Rank Approximations.* 2009.

[CG05] Hans-Görg Roos Christian Großmann. *Numerische Behandlung partieller Differentialgleichungen.* Teubner, 2005.

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

[Loa99] Charles F. Van Loan. *The ubiquitous Kronecker product.* 1999.

[Mik09] Carl Christian Kjelgaard Mikkelsen. *Numerical Methods for large Lyapunov Equations.* 2009.

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

[Ran05] Prof. Dr. Rannacher. *Einführung in die Numerische Mathematik Vorlesungsskriptum.* 2005.

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

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

[uYZ03] Danny Sorensen und Yunkai Zhou. *Direct Methods for Matrix Sylvester and Lyapunov Gleichungen.* 2003.