TENSORSTRUKTUR DER ZELLMATRIZEN BEI FINITEN ELEMENTEN

Bachelorarbeit

eingereicht von

Enes Witwit

betreuut von

Prof. Dr. Kanschat

Fakultät für Mathematik und Informatik

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.

18. 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
$\mathbf{A}^{(n)}$	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 [MK12, 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 (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 erstmal 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 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 Definitionen 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

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

mit $\Omega = [\,0\,,1\,]^2$ und $f : \bar{\Omega} \to \mathbb{R}$ zwei mal stetig partiell differenzierbar. 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 partiell stetig differenzierbar sein sollte. Die Lösung sollte also in

dem Raum der zweimal partiell stetig differenzierbaren Funktionen liegen.

Nun kann es aber sein, dass eine Lösung für dieses Problem nicht in diesem Raum existiert. Wir können uns 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 Lösungen finden können, in Räumen die nicht alle notwendigen Regularitätsbedingungen erfüllen. Diese Lösungen nennen wir dann *schwache Lösungen*. Doch wie finden wir schwache Lösungen? Dafür nutzen wir eine funktionalanalytische 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

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

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 erster 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*. Nun wissen wir in welchem Raum die klassische Lösung liegt, aber welche Topologie ist für die schwache Lösung sinnvoll? Ein funktionalanalytischer Ansatz versucht nun die Räume zu definieren, in der die Lösung u für (2.5) liegt. In unserem Fall wäre folgender Raum ergiebig

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

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*

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

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

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

Das heißt Sobolev Räume sind eine Teilmenge von den L_p 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 [Joh08, 36]. Beide genannten Themen würden den Rahmen dieser Bachelorarbeit sprengen, daher verweise 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.

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 ellptischer Randwertaufaben zweiter bzw. 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

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

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 wieder zurück zu unserem ursprünglichen Problem.

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

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. 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

ursprüngichen 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 bringt uns den Vorteil, dass wir u_n als lineare Kombination der Basiselemente von V_n approximieren können mit endlich vielen Parametern.

Die diskrete schwache Formulierung sieht jetzt 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 nur die Basis zum Testen zu nutzen. Das obere 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) erhalten wir eine neue Darstellung des diskreten Problems.

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

Das 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:

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

Das heißt, es gilt 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 n die Approximation besser wird. Das Verhalten des Fehlers im Bezug zum Diskretierisierungsparameters n wird in [CG05, 154] ausgiebig untersucht.

Bemerkung 2.7. *(Galerkin Eigenschaften)*

1. *Galerking 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 Ritz-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) ist es zu erst notwendig

1. Das Grundgebiet in geometrisch einfache Teilgebiete $\Omega_h = \{ \Omega_k \}_{k=1,\dots,N}$ z.B. 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 globale 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* $V_h = lin \{ \phi_i \}_{i=0}^{N}$, *wobei die Ansatzfunktionen sind durch*

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

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

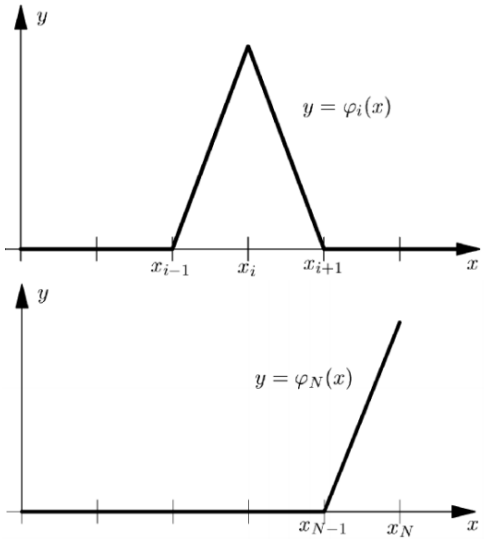

Abbildung 1: Ansatzfunktionen ϕ_i [CG05, 184]

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

Nun haben wir eine Idee davon, wie man ein Gebiet zerlegt und wie Ansatzfunktionen aussehen könnten und welche Eigenschaften sie zu erfüllen haben, doch wie genau sieht nun das diskrete Problem bei der *Finite Elemente Methode* nun 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* zu der *globalen Steifigkeitsmatrix* zusammen setzen. Wir werden hier nodale Basisfunktionen benutzten. Diese sind durch $\varphi_k(x_l) = \delta_{kl}$ definiert. Weiterhin sei $\hat{N}$ die Zahl 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 $\{\varphi_i\}_{i=1}^{\hat{N}}$ die Basis von V_n
3. Definiere $A_n = (a(\varphi_k, \varphi_i))_{i,k=1}^{\hat{N}}$ und $f_n = (f(\varphi_i))_{i=1}^{\hat{N}}$
4. Löse lineares Gleichungssystem $A_n u_n = f_n$ zur Bestimmung der Koeffizienten u_i der Darstellung $u_n(x) = \sum_{i=1}^{\hat{N}} u_i \varphi_i(x)$

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 wegen der Addivität des Integrals als Summen

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

und

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

I_j ist gerade die List 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 vom Integral zu nutze machen und mit Hilfe vom Transformationssatz für Integrale eine einheitlichere Form herleiten.

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

Um dann die 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 vom Finite Elemente Ansatz und auch der späteren Arbeit zur Herleitung der Pseudoinversen.

Wir rekapitulieren die erste Gleichung dieser Arbeit

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

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 genau diese Transformation von der wir gerade sprachen, nämlich die Transformation von den lokalen Freiheitsgraden in die globalen Freiheitsgrade. Es gilt nur noch zu klären, welche Aufgabe die Matrix C in unserer Gleichung besitzt. Diese Frage klären wir im Rahmen des nächsten Unterkapitels zum *Diskontinuierlichen Galerkin-Verfahren*.

Vorher definieren wir uns die Strukturen, die diese Arbeit untersucht. Das ist einerseits die Masse Matrix mit folgender 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. Desweiteren 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 brauchen 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 sogenannte *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. Intuitiv machen wir eine Polynominterpolation für die zu integrierende Funktion f und summieren über Stützstellen.

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

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

Doch wie genau ist diese 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.

Einer Quadraturformel wollen wir besondere Beachtung schenken, da diese für unsere Anwendung besondere praktikabilität gezeigt hat.

Definition 2.10. *(Gauss Lobatto)*
Es sei x_i die $(i-1)$ te Nullstelle des Legendre Polynoms $P_{n-1}^{\cdot}(x)$ 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 $2n-3$. Eine ausführliche Ausarbeitung der Thematik finden Sie in [Ran05, 79].

2.1.5 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. Das heißt eine Verfeinerung des Gitters ist lokal möglich, ohne direkt Probleme zu bekommen [CG05, 292].

$$v = A(u) = \sum_{k=1}^{n_{cells}} C^T P_k^T A_k (P_k C u) \tag{2.20}$$

Die Matrix C kommt aus der Diskontinuierlichen Galerkin Methode und kümmert sich um *hängende Knoten* und sorgt dafür, das wir keine Probleme mit der Stetigkeit der Lösung bekommen.

Nun können wir diese Gleichung besser fassen und haben eine Idee davon, was sie aussagt. In dieser Arbeit werden wir uns nur mit dem Term

$$v_k = A_k u \tag{2.21}$$

beschäftigen, für A_k Elementsteifigkeitsmatrix der Masse Matrix und der Laplace Bilinearform für Referenzzellen im zweidimensionalen. Die Verallgemeinerung auf die dritte Dimension ist mit wendig Mehraufwand verbunden und wird an gegebener Stelle erläutert. Außerdem wir machen uns keine Gedanken um P_k noch um C.

Das Ziel des nächsten Kapitels ist die Untersuchung der Transformation von (2.21) zu

$$A_k^+ v_k = u \tag{2.22}$$

Vorher sollten wir uns jedoch mit der Tensor Dekomposition beschäftigen und dafür eine theoretische Grundlage schaffen.

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 Masse Matrix und der Laplace Bilinearform als einen Tensor umzudefinieren. Nachdem wir dies gemacht haben, werden wir die Strukturen dieses Tensors untersuchen, mit Hinblick einen einfachen Weg zu finden, die Pseudoinverse zu bestimmen. Was genau ein Tensor ist, was wir unter der Pseudoinversen eines Tensors verstehen und wie wir einen Tensor analyisieren, wird in diesem Unterkapitel beantwortet.

Definition 2.11. *(Tensor)*
Ein Tensor ist eine multidimensionale Matrix $\mathfrak{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$. *Die Ordnung ist die Anzahl der Dimensionen, in diesem Fall* N.

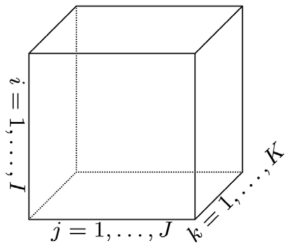

Abbildung 2: Tensor dritter Ordnung $\mathfrak{X} \in \mathbb{R}^{I \times J \times K}$ [TK09, 456]

Vorsicht zu haben, gilt es bei den Begriffen Ordnung und Rang bei Tensoren. Es sollte vermieden werden diese Begriffe zu verwechseln. Außerdem sollte man nicht den Begriff des Rangs einer Matrix, mit dem Begriff des Ranges eines Tensors verwechseln. Die Definition des Rangs eines Tensors werden wir hier auslassen, da dies ein überaus schwieriger Begriff ist und es vermieden werden kann in dieser Arbeit damit zu hantieren. Eine ausführliche Erklärung des Begriffs finden Sie in [TK09, 464].

Um unsere Tensoren zu klassifizieren und charakterisieren, brauchen 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 Matrixindizes (i_n, j)

$$j = 1 + \sum_{\substack{k=1 \\ k \neq n}}^{N} (i_k - 1) J_k \text{ mit } J_k = \prod_{\substack{m=1 \\ m \neq n}}^{k-1} I_m \tag{2.23}$$

Nun fehlt uns noch eine Tensor Multiplikation 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 $I_1 \times \ldots I_{n-1} \times J \times I_{n+1} \times \ldots I_N$

$$(\mathcal{X} \times_n \mathbf{U})_{i_1 \ldots i_{n-1} j i_{n+1} \ldots i_N} = \sum_{i_n=1}^{I_n} x_{i_1 \ldots i_N} u_{j i_n} \tag{2.24}$$

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

$$\mathcal{Y} = \mathcal{X} \times_n \mathbf{U} \Longleftrightarrow \mathbf{Y}_{(n)} = \mathbf{U}\mathbf{X}_{(n)} \tag{2.25}$$

Wir möchte noch eine Eigenschaft des n-mode Produkts hervorheben, die besonders wichtig ist für die Herleitung der Pseudoinversen.

Bemerkung 2.18.

2.2.2 Singulärwertzerlegung höherer Ordnung

Die *Singulärwertzerlegung höherer Ordnung* bzw. *Higher Order Singular Value Decomposition* (HOSVD) oder auch bekannt unter der Tucker Dekomposition ist eine uminterpretierte multidimensionale Hauptkomponentenanalyse. Man versucht durch Hauptachsentransformationen die Korrelation zwischen den verschiedenen Komponenten durch Überführung in eine neue Basis zu minimieren. Die HOSVD zerlegt den Tensor in einen sogenannten *Core Tensor* multipliziert mit einer Matrix in jeder Ordnung.

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

$$\mathcal{X} = \mathcal{G} \times_1 A^{(1)} \ldots \times_N A^{(N)} \tag{2.26}$$

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

$$\mathbf{X}_{(n)} = A^{(n)} \mathbf{G}_{(n)} \left(A^{(N)} \otimes \ldots \otimes A^{(n+1)} \otimes A^{(n-1)} \otimes \cdots \otimes A^{(1)} \right)^T \tag{2.27}$$

Beispiel 2.19. *(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

$$\mathcal{X} \approx \mathcal{G} \times_1 A \times_2 B \times_3 C \tag{2.28}$$

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 Core Tensor und zeigt wie hoch die Korrelation zwischen den verschiedenen Komponenten ist.

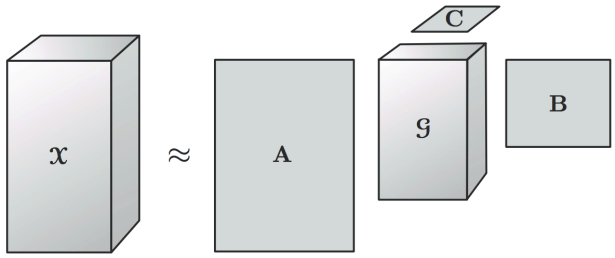

Abbildung 3: HOSVD eines Tensors dritter Ordnung [TK09, 475]

Bemerkung 2.20. *(Berechnung der HOSVD)*
Die Berechnung der HOSVD von $\mathfrak{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ *geht 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* $\mathfrak{G}$ *ergibt sich aus der Projektion des Tensors auf die Tensorbasis geformt von den Faktormatrizen* $\{U_k\}_{k=1}^N$ *also* $\mathfrak{G} = \mathfrak{X} \times_{n=1}^N U_n^T$

Die HOSVD existiert für alle Tensoren. Wie sieht es aber mit der Eindeutigkeit aus? Die HOSVD ist keine eindeutige Zerlegung. Dies führen wir an einem Tensor dritter Ordnung an.

Beispiel 2.21. *(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*

$$\mathfrak{X} = \mathfrak{G} \times_1 A \times_2 B \times_3 C = (\mathfrak{G} \times_1 U \times_2 V \times_3 W) \times_1 AU^{-1} \times_2 BV^{-1} \times_3 CW^{-1} \quad (2.29)$$

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

Mit dieser Kenntniss können wir nun zum Beispiel versuchen, soviele Elemente des Kerntensor wie möglich auf Null zu bekommen oder so klein wie möglich zu machen, damit wir bei der späteren Herleitung der Pseudoinversen weniger Probleme bekommen.

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

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

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

Lemma 2.24. *(Transponieren)*

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

Dieses Ergebnisse sind entscheidend für die Herleitung der Pseudoinversen. Die theoretische Grundlage ist nun geschaffen. Es ist Zeit sich dem Herz dieser Arbeit zu widmen, nämlich der Tensorstruktur der Elementarsteifigkeitsmatrizen und die Herleitung der Pseudoinversen.

3 Tensorstruktur und Pseudoinverse der Zellmatrizen

3.1 Summenfaktorisierung

In [Tea] 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 dieses Unterkapitels ist die Tensorstruktur für die Massematrix und der Laplace Bilinearform herzuleiten und für die Berechnung der Pseudoinversen zu nutzen.

3.1.1 Berechnung der Elementmassenmatrix

Es sei T die Referenzzelle für Rechtecke und $\varphi_i(\mathbf{x})$ Basisfunktion des diskreten Raumes V_n mit $\mathbf{x} = (x, y)$.

$$M_{ik} = \int_T \varphi_i(\mathbf{x})\, \varphi_j(\mathbf{x})\, d\mathbf{x} \tag{3.1}$$

Die Basisfunktionen haben eine Tensorstruktur, die wie folgt aussieht

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

Wir werden eine lexikographische Ordnung der Freiheitsgrade benutzen. Die Reichweite von i_1 und i_2 reicht von 0 bis N. Das heißt der Index i geht von 0 bis $N_p = (N + 1)^2 - 1$. In Abbildung (4) sehen Sie ein Beispiel für $N = 3$.

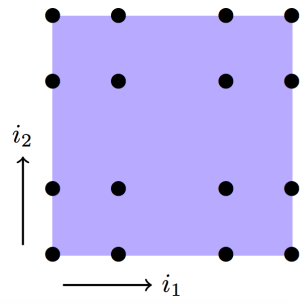

Abbildung 4: [Tea, 3]

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. Dann können wir (3.1) approximieren wie folgt

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

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

Wir definieren uns $\mathcal{N}_{iq} = \varphi_i(x_q)$ 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 Massematrix schreiben als

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

Dies ist die Form mit in $\mathcal{N}$ an Stützstellen evaluierte zweidimensionale Basisfunktionen. Wir können dies aber weiter aufspalten in eindimensionale Basisfunktionen und dadurch eine Effizienzsteigerung erzielen bei der Berechnung des Matrix-Vektor Produkts mit der Elementmassenmatrix.

3.1.2 Berechnung der Elementmassenmatrix-Vektor Produkt

Das Ziel dieses Unterkapitels ist die effiziente Berechnung des Matrix-Vektor Produkts $Mu = v$ mit M als Massematrix.

Wir fangen damit an, 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 $\mathcal{N}$ bloß mit eindimensionalen Basisfunktionen. Äquivalent 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 $Q_{1D} = N$ und $Q = (N+1)^2$, da wie wir oben bereits argumentiert haben, damit exakt integrieren können. Wir erinnern uns $v = Mu = \mathcal{N}\mathcal{W}\mathcal{N}^T$. 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 $\mathcal{W}_N$ sieht wie folgt aus

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

Genauer

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

Damit können wir folgende Umformulierung bereits vornehmen

$$v = Mu = \mathcal{N}\mathcal{W}\mathcal{N}^T = \mathcal{W}_N\mathcal{N}^T = [(\mathcal{W}_N^{1D} \otimes \mathcal{W}_N^{1D})(\mathcal{N}^{1D} \otimes \mathcal{N}^{1D})^T]u. \tag{3.7}$$

Mit Lemma (2.24) können wir das Transponieren, wie folgt, reinziehen

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

Dann nutzen wir Lemma (2.23) und erhalten

$$[(\mathcal{W}_N^{1D} \otimes \mathcal{W}_N^{1D})((\mathcal{N}^{1D})^T \otimes (\mathcal{N}_{1D})^T)]u = [(\mathcal{W}_N^{1D}(\mathcal{N}^{1D})^T) \otimes (\mathcal{W}_N^{1D}(\mathcal{N}_{1D})^T)]u. \tag{3.9}$$

3.1.3 Tensorstruktur der Laplace Bilinearform

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

$$V_{ij} = \int_T \nabla\varphi_i(\mathbf{x})\,\nabla\varphi_j(\mathbf{x})d\mathbf{x} = \int_T (\partial_{x_1}\varphi_i(\mathbf{x})\partial_{x_1}\varphi_j(\mathbf{x})) + (\partial_{x_2}\varphi_i(\mathbf{x})\partial_{x_2}\varphi_j(\mathbf{x}))\,d\mathbf{x}. \quad (3.10)$$

Wir führen elementare Umformungen durch und nutzen die Linearität des Integrals

$$\begin{aligned} V_{ij} &= \int_T (\partial_{x_1}\varphi_i(\mathbf{x})\partial_{x_1}\varphi_j(\mathbf{x})) + (\partial_{x_2}\varphi_i(\mathbf{x})\partial_{x_2}\varphi_j(\mathbf{x}))\,d\mathbf{x} \\ &= \int_T \partial_{x_1}\varphi_i(\mathbf{x})\partial_{x_1}\varphi_j(\mathbf{x})d\mathbf{x} + \int_T \partial_{x_2}\varphi_i(\mathbf{x})\partial_{x_2}\varphi_j(\mathbf{x})\,d\mathbf{x}. \end{aligned} \quad (3.11)$$

Nun nutzen wir eine Quadratur für das Integral. 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 obere Gleichung

$$V_{ij} = \underbrace{\sum_{q=1}^{(N+1)^2} \mathbf{w}_q\partial_{x_1}\varphi_i(\mathbf{x}_q)\partial_{x_1}\varphi_j(\mathbf{x}_q)}_{K_1} + \underbrace{\sum_{q=1}^{(N+1)^2} \mathbf{w}_q\partial_{x_2}\varphi_i(\mathbf{x}_q)\partial_{x_2}\varphi_j(\mathbf{x}_q)}_{K_2}. \quad (3.12)$$

Wir können eine große Ähnlichkeit mit der Struktur der Massematrix in (3.3), wenn wir uns jeweils nur K_1 und K_2 seperat anschauen. Jetzt gilt es die Tensorstruktur der Ansatzfunktionen auszunutzen. Dafür betrachten wir K_1.

$$\begin{aligned} K_1 &= \sum_{q=1}^{(N+1)^2} \mathbf{w}_q\partial_{x_1}\varphi_i(\mathbf{x}_q)\partial_{x_1}\varphi_j(\mathbf{x}_q) \\ &= \sum_{q_1=1}^{N}\sum_{q_2=1}^{N} w_{q1}w_{q2}\partial_{x_1}\varphi_{i1}(x_{q1})\varphi_{i2}(x_{q2})\partial_{x_1}\varphi_{j1}(x_{q1})\varphi_{j2}(x_{q2}) \\ &= \sum_{q_1=1}^{N}\sum_{q_2=1}^{N} w_{q1}w_{q2}\varphi'_{i1}(x_{q1})\varphi_{i2}(x_{q2})\varphi'_{j1}(x_{q1})\varphi_{j2}(x_{q2}) \\ &= \sum_{q_1=1}^{N} w_{q1}\varphi'_{i1}(x_{q1})\varphi'_{j1}(x_{q1}) \sum_{q_2=1}^{N} w_{q2}\varphi_{i2}(x_{q2})\varphi_{j2}(x_{q2}) \end{aligned} \quad (3.13)$$

Man kann die Ähnlichkeit mit der Gleichung (3.3) erkennen. Wir sehen, dass wir in einer Dimension nun aber die evaluierten Ableitungen der Basisfunktionen und in die andere Dimension die evaluierten Basisfunktionen 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.

$$\widehat{\mathcal{N}}^{1D} = \begin{bmatrix} \frac{\partial \varphi_1^{1D}(x_1)}{\partial x} & \cdots & \frac{\partial \varphi_n^{1D}(x_1)}{\partial x} \\ \vdots & \ddots & \vdots \\ \frac{\partial \varphi_1^{1D}(x_N)}{\partial x} & \cdots & \frac{\partial \varphi_n^{1D}(x_N)}{\partial x} \end{bmatrix} \tag{3.14}$$

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

$$K_1 = K_2 = (\widehat{\mathcal{W}}_N^{1D}(\widehat{\mathcal{N}}^{1D})^T) \otimes (\mathcal{W}_N^{1D}(\mathcal{N}^{1D})^T). \tag{3.15}$$

Insgesamt kann man also die Elemensteifigkeitsmatrix für die Laplace Bilinearform darstellen als

$$V = (\widehat{\mathcal{W}}_N^{1D}(\widehat{\mathcal{N}}^{1D})^T) \otimes (\mathcal{W}_N^{1D}(\mathcal{N}^{1D})^T) + (\widehat{\mathcal{W}}_N^{1D}(\widehat{\mathcal{N}}^{1D})^T) \otimes (\mathcal{W}_N^{1D}(\mathcal{N}^{1D})^T). \tag{3.16}$$

Nun können wir das zusammenfassen zu

$$V = 2((\widehat{\mathcal{W}}_N^{1D}(\widehat{\mathcal{N}}^{1D})^T) \otimes (\mathcal{W}_N^{1D}(\mathcal{N}^{1D})^T)). \tag{3.17}$$

3.1.4 Pseudoinverse

Nun wollen wir die Pseudoinversen für die Elementsteifigkeitsmatrix der Massematrix M und der Laplace Bilinearform V herleiten. Dank unserer Vorarbeit und unserem Vorwissen zum Kronecker Produkt, ist es möglich, dies sehr effizient zu machen.

Rekapituliere die Tensorstruktur für die Massematrix

$$M = (\mathcal{W}_N^{1D}(\mathcal{N}^{1D})^T) \otimes (\mathcal{W}_N^{1D}(\mathcal{N}_{1D})^T). \tag{3.18}$$

Nun wollen wir die Pseudoinverse herleiten

$$M^+ = ((\mathcal{W}_N^{1D}(\mathcal{N}^{1D})^T) \otimes (\mathcal{W}_N^{1D}(\mathcal{N}_{1D})^T))^+ \tag{3.19}$$

Wir nutzen Lemma (2.22) und erhalten

$$((\mathcal{W}_N^{1D}(\mathcal{N}^{1D})^T) \otimes (\mathcal{W}_N^{1D}(\mathcal{N}_{1D})^T))^+ = (\mathcal{W}_N^{1D}(\mathcal{N}^{1D})^T)^+ \otimes (\mathcal{W}_N^{1D}(\mathcal{N}_{1D})^T)^+ \tag{3.20}$$

Analog für die Laplace Bilinearform

$$\begin{aligned} V^+ &= (2((\widehat{\mathcal{W}}_N^{1D}(\widehat{\mathcal{N}}^{1D})^T) \otimes (\mathcal{W}_N^{1D}(\mathcal{N}^{1D})^T)))^+ \\ &= \frac{1}{2}((\widehat{\mathcal{W}}_N^{1D}(\widehat{\mathcal{N}}^{1D})^T)^+ \otimes (\mathcal{W}_N^{1D}(\mathcal{N}^{1D})^T)^+) \end{aligned} \tag{3.21}$$

Das heißt anstatt, dass wir die Pseudoinverse einer Matrix der Größe $N^2 \times N^2$ berechnen, berechnen wir nur die Pseudoinversen einer Matrix der Größe $N \times N$.

Was uns letztlich interessiert ist die Auswertung der Pseudoinversen an einem Vektor u also M^+u und V^+u. Wie wir dies effizient ausrechnen und welche Komplexität uns erwartet, sehen wir in Kapitel 4.

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

$$\mathfrak{X} = \mathfrak{G} \times_{n=1}^{N} U^{(n)} \tag{3.22}$$

Wir können einen Tensor $\mathfrak{X}$ in einen Kerntensor $\mathfrak{G}$ und zugehörige Faktormatrizen U 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 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.

Dafür sollten wir erstmal unsere Tensoren herleiten. Dies geschieht mit Hilfe der Tensorstruktur der Ansatzfunktionen. Wir definieren den *Massetensor* elementweise durch

$$M_{i1,i2,j1,j2} = \int_T \varphi_{i1}(x_1)\varphi_{i2}(x_2)\varphi_{j1}(x_1)\varphi_{j2}(x_2)\, d(x_1, x_2) \tag{3.23}$$

und unseren *lokalen Laplace Tensor*, welcher das pendant zu der Elementsteifigkeitsmatrix der Laplace Bilinearform ist, wie folgt

$$V_{i1,i2,j1,j2} = \int_T \varphi'_{i1}(x_1)\varphi_{i2}(x_2)\varphi'_{j1}(x_1)\varphi_{j2}(x_2) + \varphi_{i1}(x_1)\varphi'_{i2}(x_2)\varphi_{j1}(x_1)\varphi'_{j2}(x_2)\, d(x_1, x_2) \tag{3.24}$$

Diese Transformation von Matrix zu Tensor ist also eigentlich eine Abbildung die ein Indextupel (i, j) eines Matrixelements auf den Indextupel eines Tensorelements (i_1, i_2, j_1, j_2) abbildet. Damit wir uns eine Tensor-Tensor Multiplikation definieren, sollte uns diese Transformation klar sein. In dieser Transformation stecken implizit zwei mal 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

$$p^{-1}(i_1, i_2) = i_1 + (N+1)i_2 = i \tag{3.25}$$

Wie können wir nun aber gegeben i das korrespondiere Tupel (i_1, i_2) berechnen? Dazu nutzen wir die Modulo Rechnung. Wir nehmen einfach das Inverse der Transformation *modulo* $(N+1)$.

$$i \ (mod(N+1)) = p^{-1}(i_1, i_2) \ (mod(N+1)) = i_1 + \underbrace{(N+1)i_2}_{0} \ (mod(N+1)) \tag{3.26}$$

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

$$i \ (mod(N+1)) = i_1 \ \ (mod(N+1)) = i_1. \tag{3.27}$$

Nun wissen wir, wie wir aus der Information i unser korrespondierendes i_1 extrahieren können. Die Gleichung (3.25) können wir nun nach i_2 wie folgt umstellen

$$i_2 = \frac{i - i_1}{N+1} \tag{3.28}$$

Mit dem Wissen über i_1 können wir dies weiter umformen

$$i_2 = \frac{i - (i \ (mod(N+1)))}{N+1} \tag{3.29}$$

Damit haben wir unsere Transformation p gefunden

$$p(i) = \Big(i \ (mod(N+1)), \frac{i - (i \ (mod(N+1)))}{N+1}\Big) \tag{3.30}$$

und eindeutig festmachen, welches Element der Matrixform zu welchem Element der Tensorform gehört. Da wir nun die Transformationen kennen, können wir diese zu Hilfe nehmen für unser Tensor-Tensor Produkt. Vorher sollten wir uns Matrix-Matrix Produkt anschauen.

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

$$C_{ik} = \sum_{j=1}^{N^2} M_{ij} M_{jk} \tag{3.31}$$

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

$$C_{p(i),p(k)} = C_{i_1,i_2,j_1,j_2} = \sum_{j=1}^{N^2} M_{p(i),p(j)} M_{p(j),p(k)} = \sum_{j_1=1}^{N} \sum_{j_2=1}^{N} M_{i_1,i_2,j_1,j_2} M_{j_1,j_2,k_1,k_2} \tag{3.32}$$

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

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

das Produkt dieser beiden Tensoren wie folgt elementenweise

$$ttp(\mathfrak{X}^1, \mathfrak{X}^2)_{i_1,i_2,j_1,j_2} = \sum_{j_1=1}^{I_1} \sum_{j_2=1}^{I_2} \mathfrak{X}^1_{i_1,i_2,j_1,j_2} \mathfrak{X}^2_{j_1,j_2,k_1,k_2} \tag{3.33}$$

Von der Komplexität her, ist das Tensor-Tensor-Produkt der Masse-Tensoren bzw. der lokalen Laplace-Tensoren genau so komplex, wie das Produkt von Massematrizen bzw. das Produkt von den Elementsteifigkeitsmatrizen der Laplace Bilinearform. Dazu später mehr in Kapitel 4.

Nun brauchen wir noch den Operator des Transponierens für Tensoren. Analog wie für den Tensor-Tensor-Produkt, können wir uns den Operator des Transponierens erstmal für Matrizen anschauen. Sei $A \in \mathbb{R}^{N^2 \times N^2}$ beliebige Matrix. Dann ist die transponierte Matrix gegeben durch

$$A^T_{ij} = A_{ji}. \tag{3.34}$$

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

$$A^T_{p(i)p(j)} = A^T_{i_1\, i_2\, i_1\, j_2} = A_{j_1\, j_2\, i_1\, i_2} = A_{p(j)\, p(i)}. \tag{3.35}$$

Wir können nun die Moore Penrose Pseudoinverse Eigenschaften auch für Tensoren angeben. Vorher sollte aber das Problem mit der Maschinengenauigkeit angesprochen werden. Dazu gibt es ein Trick den wir nutzen können. Der Trick kann nur mit Vorsicht genossen werden. 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. $ttp(A^+, ttp(A, A^+)) - A^+ \ < \epsilon$
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, 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?

Aus der HOSVD ergibt sich die Zerlegung für einen Tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ mit

$$\mathcal{X} = \mathcal{G} \times_{n=1}^{N} U^{(n)}. \tag{3.36}$$

Nun nehmen wir die Pseudoinverse von beiden Seiten. Das können wir machen, da uns mittlerweile bekannt ist, was es bedeutet die Pseudoinverse von einem Tensor zu haben. Wir erhalten

$$\mathcal{X}^{+} = (\mathcal{G} \times_{n=1}^{N} U^{(n)})^{+}. \tag{3.37}$$

Wir hätten jetzt gerne ein Ergebnis, das uns sagt, dass wir den Pseudoinversen Operator einfach in die Klammer reinziehen können und so etwas erhalten wie

$$\mathcal{X}^{+} = \mathcal{G}^{+} \times_{n=1}^{N} U^{(n)^{+}} \tag{3.38}$$

Dies gilt es zu beweisen

Lemma 3.4. *(Verträglichkeit Pseudoinversen Operator mit n-mode Produkt)* *Es sei* $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ *und die Pseudoinversen der Zerlegung gegeben durch* $\mathcal{X}^{+} = (\mathcal{G} \times_{n=1}^{N} U^{(n)})^{+}$. *Dann gilt*

$$\mathcal{X}^{+} = (\mathcal{G} \times_{n=1}^{N} U^{(n)})^{+} = \mathcal{G}^{+} \times_{n=1}^{N} U^{(n)^{+}} \tag{3.39}$$

Beweis. Wir können die Tensoren entfalten und erhalten

$$\mathbf{X}_{(n)}^{+} = \Big(U^{(n)} \mathbf{G}_{(n)} \big(U^{(N)} \otimes \cdots \otimes U^{(n+1)} \otimes U^{(n-1)} \otimes \cdots \otimes U^{(1)} \big) \Big)^{+} \tag{3.40}$$

Wir substituieren wie folgt um $A := U^{(n)} \mathbf{G}_{(n)}$ und $B := \big(U^{(N)} \otimes \cdots \otimes U^{(n+1)} \otimes U^{(n-1)} \otimes \cdots \otimes U^{(1)} \big)$. Damit sieht die obere Gleichung wie folgt aus

$$\mathbf{X}_{(n)}^{+} = \Big(A\, B \Big)^{+} \tag{3.41}$$

Es sei wohlgemerkt, dass A und B Matrizen sind. Dementsprechend nutzen wir einfach bekannte Matrizeneigenschaften und erhalten.

$$\mathbf{X}_{(n)}^{+} = \Big(A\,B\Big)^{+} = B^{+}\,A^{+} \tag{3.42}$$

Wir resubstituieren

$$B^{+}\,A^{+} = \big(U^{(N)} \otimes \cdots \otimes U^{(n+1)} \otimes U^{(n-1)} \otimes \cdots \otimes U^{(1)}\big)^{+} (U^{(n)}\mathbf{G}_{(n)})^{+}. \tag{3.43}$$

Jetzt können wir Lemma (2.22) nutzen und erhalten

$$\begin{aligned} B^{+}\,A^{+} &= \big(U^{(N)} \otimes \cdots \otimes U^{(n+1)} \otimes U^{(n-1)} \otimes \cdots \otimes U^{(1)}\big)^{+} (U^{(n)}\mathbf{G}_{(n)})^{+} \\ &= \big((U^{(N)})^{+} \otimes \cdots \otimes (U^{(n+1)})^{+} \otimes (U^{(n-1)})^{+} \otimes \cdots \otimes (U^{(1)})^{+}\big)\mathbf{G}_{(n)}^{+} (U^{(n)})^{+}. \end{aligned} \tag{3.44}$$

Wir haben praktisch das B umgeformt, indem wir den Pseudoinversen Operator in Klammer gezogen haben und eben A mit ausgeklammert. Nun wollen wir das Ergebnis wieder zu einem Tensor falten. Dazu nutzen wir ein Argument von [Kol06, 11] *Proposition 3.7b* und erhalten das gewünschte Ergebnis.

□

Also können wir die Pseudoinverse wie folgt angeben

$$\mathfrak{X}^{+} = \mathfrak{G}^{+} \times_{n=1}^{N} U^{(n)^{+}} \tag{3.45}$$

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

$$\mathfrak{X}^{+} = \mathfrak{G}^{+} \times_{n=1}^{N} U^{(n)^{T}} \tag{3.46}$$

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 zwei 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.

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.

4.1 Effizientes Matrix-Vektor Produkt

In [Tea] wird eine Strategie vorgestellt 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, die sich wiederholende Struktur von $\mathcal{A}$. Genau diese wollen wir uns 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 in [Tea] vorgeschlagen z_i mit

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

Mit Hilfe diesen Algorithmus haben wir die Komplexität von des Auswertens der Gleichung von $2m^4$ auf $4m^3$ reduziert.

4.1.1 Erweiterung

Dieser Algorithmus ist optimal um für die erste Alternative mit der Pseudoinversen Berechnung durch Ausnutzung der Tensorstruktur, effektiv das Matrix-Vektor Produkt auszurechnen. Doch für die zweite Alternative mit der HOSVD, werden wir eine erweiterte Form des Algorithmuses brauchen, da wir mehrere Kronecker Produkte bekommen werden. Wir wollen den Algorithmus erweitern für die effektive Berechnung von $z = (\mathcal{C} \otimes \mathcal{B} \otimes \mathcal{A})v$ mit $\mathcal{A} \in \mathbb{R}^{n_1 \times m_1}$, $\mathcal{B} \in \mathbb{R}^{n_2 \times m_2}$,$\mathcal{C} \in \mathbb{R}^{n_3 \times m_3}$ und $v \in \mathbb{R}^{m_1 m_2 m_3}$.

$$z = \begin{pmatrix} c_{11}b_{11}\mathcal{A} & \dots & c_{11}b_{1m_2}\mathcal{A} & \dots & \dots & c_{1m_3}b_{11}\mathcal{A} & \dots & c_{1m_3}b_{1m_2}\mathcal{A} \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \vdots \\ c_{11}b_{n_21}\mathcal{A} & \dots & c_{11}b_{n_2m_2}\mathcal{A} & \dots & \dots & c_{1m_3}b_{n_21}\mathcal{A} & \dots & c_{1m_3}b_{n_2m_2}\mathcal{A} \\ c_{21}b_{n1}\mathcal{A} & \dots & c_{21}b_{nn}\mathcal{A} & \dots & \dots & c_{2n}b_{n1}\mathcal{A} & \dots & c_{2m_3}b_{n_2m_2}\mathcal{A} \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \vdots \\ c_{n_31}b_{n_21}\mathcal{A} & \dots & c_{n_31}b_{n_2m_3}\mathcal{A} & \dots & \dots & c_{n_3m_3}b_{n_21}\mathcal{A} & \dots & c_{n_3m_3}b_{n_2m_2}\mathcal{A} \end{pmatrix} v \tag{4.1}$$

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

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

Unser v können wir zu einem Tensor $\mathcal{V} \in \mathbb{R}^{m_1 \times m_2 \times m_3}$ umdefinieren, damit dieser

handlicher wird. Der erste Index repräsentiert in welcher Spalteneintrag von C, der Zweite in welchem Spalteneintrag von B und der Dritte in welchem Spalteneintrag von A wir uns befinden. Dies kann man wieder als eine Index Transformation ansehen, die bestimmte Einträge von v in bestimmte Tensorelemente abbildet. Wir schauen uns erstmal die einzelnen Einträge von z an.

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

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

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

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

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

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

Der komplette Algorithmus sieht wie folgt aus.

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

Wir haben in 4.2 eine Matrix-Vektor Multiplikation von einer Matrix der Größe $n_1n_2n_3 \times m_1m_2m_3$. Dementsprechend hätten wir $(n_1n_2n_3)^2$ Multiplikationen und $(m_1m_2m_3)^2$ Additionen. Sei $n_M = max(n_1, n_2, n_3)$ und $m_M = max(m_1, m_2, m_3)$. Die Komplexitätsklasse ist dann $O(n_M^6 + m_M^6)$. Die Komplexität des vorgeschlagenen Algorithmuses reduziert die Operationen auf $3n^4$ Multiplikationen und genau so viele Additionen. Ein enorme Reduktion, vor allem für großes n.

4.2 Anwendung

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

4.2.1 Summenfaktorisierung

Aus der Tensorsturktur aus Kapitel 3 für die lokale Massematrix und die Elementsteifigkeitsmatrix der Laplace Bilinearform, haben wir die Pseudoinversen hergeleitet. Nun schauen wir uns das Matrix-Vektor Produkt mit den Pseudoinversen als Matrix und einen beliebigen Vektor u.

Masse Matrix

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

Laplace Bilinearform

$$V^+u = [\frac{1}{2}((\widehat{\mathcal{W}}_N^{1D}(\widehat{\mathcal{N}}^{1D})^T)^+ \otimes (\mathcal{W}_N^{1D}(\mathcal{N}^{1D})^T)^+)]u$$

Wie komplex ist es die Pseudoinversen der Matrizen $(\mathcal{W}_N^{1D}(\mathcal{N}^{1D})^T)$ und $(\widehat{\mathcal{W}}_N^{1D}(\widehat{\mathcal{N}}^{1D})^T)$ zu berechnen?

Wir können als Ansatz eine Singulärwertzerlegung wählen. Aus der Singulärwertzerlegung müssen wir noch die Pseudoinverse erst herleiten. Allgemein gilt

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

für eine eine $m \times n$-Matrix M mit Rang r, wobei U eine orthogonale $m \times m$-Matrix ist, V^T die Transponierte einer orthogonalen $n \times n$-Matrix V und Σ eine reelle $m \times n$-Matrix mit den r Singulärwerten in der Diagonalen und sonst Nullen.

Die Pseudoinverse daraus hergeleitet 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 einfach nur invertieren müssen. Die Berechnung von U^T ist vernachlässigbar, da man bei der Berechnung der Singulärwertzerlegung direkt U^T speichern kann anstatt U. Also bekommen wir mit der

Singulärwertzerlegung einer Matrix $M \in \mathbb{R}^{m \times n}$ die Pseudoinverse mit einer Komplexität von $O(min(mn^2, m^2n))$ [AF09, 2]. Man könnte natürlich hier auch approximative Verfahren wählen und versuchen an Operationen zu sparen. Ich verweise hier auf [AF09] für die nähere Betrachtung von schnelleren Singulärwertzerlegungen.

Wir können stattdessen mit dem Gauß Algorithmus oder mit der Neumann Reihe arbeiten. Gauß Algorithmus gibt uns eine Komplexität von $O(n^3)$. Das Problem ist jedoch, dass die Matrizen allgemein nicht invertierbar sind. Neumann Reihe macht keinen Sinn, da die Komplexität mindestens genau so hoch ist wie bei Gauß.

- Um das Matrix-Vektor Produkt effizient zu berechnen nutzen wir einfach 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 also n^3.
- Berechnung der Pseudoinversen des Kronecker Produkts liegt in der Komplexitätsklasse $O(n^3)$

Insgesamt haben wir eine Komplexität von $O(4n^3) + O(n^3) + O(n^3) = O(6n^3)$. Wie wir noch weiter an Komplexität sparen können und wo wir anknüpfen können um effizienter zu werden, wird in Kapitel 5 diskutiert.

4.2.2 Singulärwertzerlegung höherer Ordnung

Nun haben wir uns mit Hilfe der Tucker Dekomposition eine Herleitung für die Pseudoinverse erarbeitet. Jetzt geht es um die effiziente Berechnung dieser Formel. Dazu wollen wir uns die Strategie zu nutze machen, die wir in Kapitel 4.1.1. hergeleitet haben für die effektive Berechnung von 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. Die Formel für die Pseudoinverse lautet

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

Wobei $\mathcal{G} \in \mathbb{R}^{I_1 \times \cdots \times I_4}$ und $U^{(n)} \in \mathbb{R}^{I_n \times I_n}$ Man kann (4.6) wie in (2.27) äquivalent umformen zu

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

Die Äquivalenz folgt mit Lemma (2.24). Im nächsten Schritt betrachten wir das Matrix-Vektor Produkt mit einem beliebigen Vektor $u_n \in \mathbb{R}^{I^N \cdots I^{n-1}\, I^{n+1} \cdots I^1}$ und überlegen wie wir uns die Strukturen dort zu nutze machen.

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

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

$$\mathcal{X}^+_{(n)} v = U^{(n)^T} \mathcal{G}^+_{(n)} (U^{(N_1)} \otimes U^{(N_2)} \otimes U^{(N_3)}) u_n \,. \tag{4.9}$$

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:

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

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:

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

5 Resultate

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.

[Kol06] Tamara Kolda. *Multilinear operators for higher-order decompositions.* 2006.

[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.