

TENSORSTRUKTUR DER ZELLMATRIZEN BEI FINITEN ELEMENTEN

Bachelorarbeit

eingereicht von

Enes Witwit

betreut von

Prof. Dr. Kanschat

Fakultät für Mathematik und Informatik

Universität Heidelberg

Hiermit versichere ich, Enes Witwit, dass ich die vorliegende Bachelorarbeit selbstständig 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 Prüfungsverfahrens gewesen ist.

25. 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 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 $O(N^6)$.

Abstract

The following work attempts to derive efficient algorithms and approaches to compute the matrix vector product with the pseudoinverse of specific cellmatrices. The cellmatrices that we take into account are from the mass matrix and the laplace bilinearform. We derived two approaches to compute the pseudoinverse. The one uses the tensor structure of the cellmatrices to derive 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 of flexibility. The second approach is very flexible but it has a high complexity order $O(N^6)$.

Inhaltsverzeichnis

Notation	I
Abbildungsverzeichnis	II
1 Einführung	1
2 Theorie	3
2.1 Numerische Behandlung partieller Differentialgleichungen	3
2.2 Tensor Dekomposition	15
3 Pseudoinversen der Zellmatrizen	20
3.1 Summenfaktorisierung	20
3.2 Singulärwertzerlegung höherer Ordnung	25
4 Effiziente Implementierung	32
4.1 Effizientes Matrix-Vektor Produkt	32
4.2 Anwendung	37
4.3 Singulärwertzerlegung höherer Ordnung	40
5 Resultate	44
Literatur	45

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
$\text{int}(\Omega)$	Inneres von Ω
$C^l(\Omega)$	Räume stetig-differenzierbarer Funktionen
$L_p(\Omega)$	Räume zur p -ten Potenz integrierbarer Funktionen ($1 \leq p \leq \infty$)
$W_p^l(\Omega)$	Sobolev-Raum
$H^l(\Omega), H_0^l(\Omega)$	Sobolev-Räume für $p=2$
$\text{supp}(v)$	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

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]	18
4	[Tea, 3]	20

1 Einführung

Hochleistungsrechnen ist eine neue Disziplin, die mit dem Drang entstand, immer komplexere Probleme zu lösen. Es ist eine neue Art an Probleme ranzugehen. 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 Probleme 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.

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

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 $i = 1, \dots, n$, 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, groß werden. Wenn sie so groß wird, dass sie nicht mehr im Cache liegt, bekommen wir das Problem, da der Zugriff auf Elemente der Matrix sehr teuer wird. Dementsprechend wollen wir uns vor allem 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

$$A_k^+ v_k \quad (1.2)$$

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 undefinieren 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_k zu nutze 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 undefinieren 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.

Dann kommen wir zu einem ganz anderen Thema, nämlich dem der Tensor Dekomposition. 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 durchführen. In Hoffnung, 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 $f : \bar{\Omega} \rightarrow \mathbb{R}$ zwei mal stetig partiell differenzierbar. Der funktionalanalytische Ansatz beschäftigt sich erst mit notwendigen Funktionenräumen um dann auf analytischer Ebene eine Umformulierung der Differentialgleichung durchzuführen, 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.

Es kann aber sein, dass eine Lösung für dieses Problem nicht in diesem Raum existiert. Wir können uns dafür als Beispiel die Betragsfunktion $b : \mathbb{R} \rightarrow \mathbb{R}$ anschau-

en.

$$b(x) = |x| \quad (2.2)$$

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

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

Dies zeigt uns, dass wir potentiell auch eine Art von Lösungen finden können, in Räumen, deren Elemente nicht alle notwendigen Regularitätsbedingungen erfüllen. Diese Lösungen nennen wir dann *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.

$$\int_{\Omega} -\Delta u v \, dx = \int_{\Omega} f v \, dx \quad (2.4)$$

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

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

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*. Es ist jetzt klar, in welchem Raum die klassische Lösung liegt, aber welche Topologie ist für die schwache Lösung sinnvoll? 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 Sobolev Räume definiert durch

Definition 2.2. (*Sobolev Raum*)

Es sei $\alpha = (\alpha_1, \dots, \alpha_n)$ mit $\alpha_i \geq 0$ ganzzahlig. Weiterhin sei $|\alpha| = \sum_i \alpha_i$ und

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

Für $k = 1, 2, \dots$ 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 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*)

$$\|v\|_{W^{k,p}(\Omega)} = \begin{cases} \left(\sum_{|\alpha| \leq k} \|D^\alpha v\|_{L^p(\Omega)}^{1/p} dx \right)^{1/2} \text{ für } p < \infty \\ \max_{|\alpha| \leq k} \|D^\alpha v\|_{L^\infty(\Omega)} dx \text{ für } p = \infty \end{cases}$$

$W^{k,p}(\Omega)$ 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 Randwertaufgaben 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

$$\int_{\Omega} \nabla u \cdot \nabla v \, dx = \int_{\partial\Omega} \frac{\partial u}{\partial n} v \, ds - \int_{\Omega} \Delta u \, \Delta v \, dx$$

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.

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

Wir wissen jede Lösung dieses Problems erfüllt die Variationsgleichung

$$\int_{\Omega} -\nabla u \cdot \nabla v \, dx = \int_{\Omega} f v \, dx \quad \text{für alle } v \in H_0^1(\Omega) \quad (2.6)$$

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

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

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 $a(\cdot, \cdot) : V \times V \rightarrow \mathbb{R}$ eine Bilinearform. Sei ferner $f \in V^*$. Falls die Bilinearform $a(\cdot, \cdot)$ symmetrisch ist, ist $u \in V$ ein Minimierer der Gleichung

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

genau dann wenn u die schwache Formulierung löst.

2.1.2 Galerkin Verfahren

Unser Ausgangspunkt ist nun die Lösung des Problems

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

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 ursprünglichen 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:

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

Sei nun e_1, \dots, 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:

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

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

$$u_n = \sum_{j=1}^n u_j e_j \quad (2.11)$$

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

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

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

$$Au = f \quad (2.13)$$

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 Diskretisierungsparameters 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 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 unserer 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 Übergangsbedingungen, 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. $\text{int}(\Omega_i) \cap \text{int}(\Omega_j) = \emptyset$, 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 $i = 1, \dots, N$. Ferner sei $h_i := x_i - x_{i-1}$, $i = 1, \dots, N$. Wir wählen lineare Ansatzfunktionen $V_h = \text{lin}\{\varphi_i\}_{i=0}^N$, wobei die Ansatzfunktionen sind durch

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

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

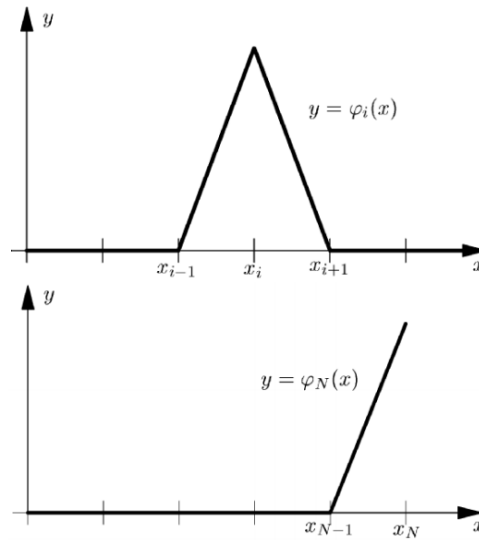


Abbildung 1: Ansatzfunktionen φ_i [CG05, 184]

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

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 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* zu der *globalen Steifigkeitsmatrix* zusammen setzen. Wir werden hier nodale Basisfunktionen benutzen. 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

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

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 Additivität 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 Dreiecken 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

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

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.

Die Strukturen, die wir betrachten, sollten definiert werden. Das ist einerseits die Masse Matrix mit der Bilinearform

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

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

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

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 *Quadraturformeln*.

2.1.4 Quadratur

Allgemein beschäftigt uns das Integrationsproblem

$$I(f) = \int_a^b f(x) \, dx, \quad (2.18)$$

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 $x_i \in \mathbb{R}$, $a \leq x_0 < \dots < x_n \leq b$ Stützstellen mit Gewichten $\alpha_i \in \mathbb{R}$. Dann können wir das Integral (2.18) approximieren durch

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

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 $P'_{n-1}(x)$ 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

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

Gauss Lobatto liefert uns eine exakte Approximation von Polynomen bis zu Grad $2n - 3$. Eine ausführliche Ausarbeitung der Thematik finden Sie in [Ran05, 79].

2.2 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 dieser Tensors untersuchen. Das Ziel dieser Untersuchungen wird sein 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 $\mathfrak{X} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_N}$. Die Ordnung ist die Anzahl der Dimensionen, in diesem Fall N .

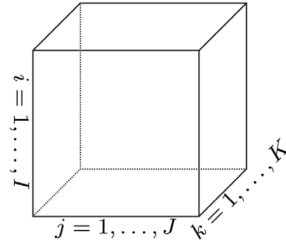


Abbildung 2: Tensor dritter Ordnung $\mathfrak{X} \in \mathbb{R}^{I \times J \times K}$ [TK09, 456]

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 $\mathfrak{X} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_N}$ ein Tensor.

Definition 2.12. (*Symmetrie*)

- a) Den Tensor \mathfrak{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 \mathfrak{X} nennt man *diagonal*, wenn $x_{i_1, \dots, i_N} \neq 0$ genau dann wenn $i_1 = \dots = i_N$.

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 \mathfrak{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, \dots, i_N) auf das Matrixindize-Tupel (i_n, j)

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

Es 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 \mathfrak{X} mit einer Matrix $\mathbf{U} \in \mathbb{R}^{J \times I_n}$ wird als $\mathfrak{X} \times_n \mathbf{U}$ notiert. Die Ergebnismatrix hat die Größe $I_1 \times \dots \times I_{n-1} \times J \times I_{n+1} \times \dots \times I_N$

$$(\mathfrak{X} \times_n \mathbf{U})_{i_1 \dots i_{n-1} j i_{n+1} \dots i_N} = \sum_{i_n=1}^{I_n} x_{i_1 \dots i_N} u_{j i_n} \quad (2.21)$$

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

$$\mathbf{Y} = \mathbf{X} \times_n \mathbf{U} \iff \mathbf{Y}_{(n)} = \mathbf{U} \mathbf{X}_{(n)} \quad (2.22)$$

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 bzw. mode.

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

$$\mathbf{X} = \mathbf{G} \times_1 A^{(1)} \dots \times_N A^{(N)}. \quad (2.23)$$

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

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

Beispiel 2.18. *(HOSVD Tensor dritter Ordnung)*

Es sei $\mathbf{X} \in \mathbb{R}^{I \times J \times K}$. Dann kann man den Tensor \mathbf{X} zerlegen in

$$\mathbf{X} \approx \mathbf{G} \times_1 A \times_2 B \times_3 C, \quad (2.25)$$

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 \mathbf{G} bezeichnet den Kerntensor und zeigt wie hoch die Korrelation zwischen den verschiedenen Komponenten ist.

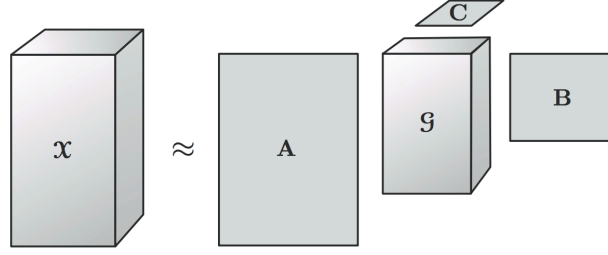


Abbildung 3: HOSVD eines Tensors dritter Ordnung [TK09, 475]

Bemerkung 2.19. (*Berechnung der HOSVD*)

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

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 2.20. (*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

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

In anderen Worten: Wir können den Kerntensor \mathbf{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, so viele 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 nutzen wollen.

Lemma 2.21. *(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 2.22. *(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 2.23. *(Transponieren)*

Es seien A, B beliebige Matrizen. Es gilt

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

Ausführliche Untersuchungen des Kronecker Produkts finden Sie in [Loa99]. 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 Pseudoinversen der Zellmatrizen

3.1 Summenfaktorisierung

In [Tea] wird die effektive Berechnung der Masse Matrix mit einem Vektor multipliziert, vorgestellt. Diese Methodik sollten wir uns kurz vor Augen führen und daraufaufbauend um einige eigene Gedanken erweitern um sie für unsere Anwendung zugänglich zu machen. Das Ziel dieses Unterkapitels ist es die Tensorstruktur für die Massematrix 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 $\varphi_i^{2D}(\mathbf{x})$ zweidimensionale reelle Basisfunktion des diskreten Raumes V_n mit $\mathbf{x} = (x, y)$.

$$M_{ik} = \int_T \varphi_i^{2D}(\mathbf{x}) \varphi_j^{2D}(\mathbf{x}) d\mathbf{x} \quad (3.1)$$

Die Basisfunktionen haben eine Tensorstruktur, die wie folgt aussieht

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

wobei φ^{1D} 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 $N_p = (N + 1)^2$. In Abbildung (4) sehen Sie ein Beispiel für $N = 3$.

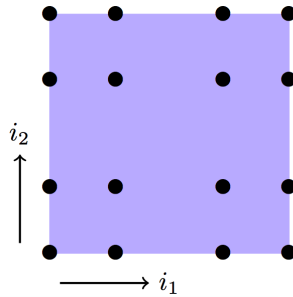


Abbildung 4: [Tea, 3]

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. 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. Wir wählen Q_{1D} gleich der Anzahl der Basisfunktionen N , 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 Massematrix 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 dies aber weiter aufspalten in eindimensionale Basisfunktionen, dank der Tensorstruktur der Basisfunktionen, und dadurch eine Effizienzsteigerung erzielen bei der Berechnung des Matrix-Vektor Produkts mit der Elementmassenmatrix.

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. Ausgeschrieben sieht die Gleichung (3.5) wie folgt aus:

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

Wir nutzen absofort $Q_{1D} = N + 1$ und $Q = (N + 1)^2$, da wie wir oben bereits argumentiert haben, damit exakt integrieren können. 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.

$$\mathcal{W}_N = \mathcal{W}_N^{1D} \otimes \mathcal{W}_N^{1D} \quad (3.6)$$

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

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

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 (2.22) und erhalten

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

3.1.2 Tensorstruktur der Elementsteifigkeitsmatrix 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

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

Wir nutzen die Linearität des Integrals.

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

Für die Integrale verwenden wir wieder 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 obere Gleichung

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

Wir können eine große Ähnlichkeit mit der Struktur der Massematrix in (3.3), wenn wir uns jeweils nur K^1 und K^2 separat anschauen. Jetzt gilt es die Tensor-

struktur der Ansatzfunktionen auszunutzen. Dafür betrachten wir K^1 .

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

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

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

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

$$\begin{aligned}
K_1 &= (\widehat{\mathcal{W}}_N^{1D} (\widehat{\mathcal{N}}^{1D})^T) \otimes (\mathcal{W}_N^{1D} (\mathcal{N}^{1D})^T) \\
K_2 &= (\mathcal{W}_N^{1D} (\mathcal{N}^{1D})^T) \otimes (\widehat{\mathcal{W}}_N^{1D} (\widehat{\mathcal{N}}^{1D})^T)
\end{aligned}$$

Insgesamt kann man die Elementsteifigkeitsmatrix 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 gilt es jetzt herzuleiten. Dank unserer Vorarbeit und unserem

Vorwissen zum Kronecker Produkt, ist es möglich, dies problemlos zu machen.

Rekapituliere die Tensorstruktur für die Massematrix

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

Nun wollen wir die Pseudoinverse herleiten

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

Wir nutzen Lemma (2.21) und erhalten

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

Hier können wir nun eine Singulärwertzerlegung zum berechnen der Pseudoinversen oder den Gauß Algorithmus zum berechnen der Inversen verwenden. Warum das Ganze? Wir hätten auch die Inverse von der Massematrix berechnen können. Der Unterschied ist, dass wir statt der Inversen einer Matrix der Größe $N^2 \times N^2$ berechnen, berechnen wir die Inverse einer Matrix der Größe $N \times N$.

Für die Laplace Bilinearform können wir dies leider nicht so einfach machen. Das Problem ist die Addition, die uns das ganze stark erschwert. Je nach Basis, kann man das Problem vereinfachen. Man könnte beispielsweise die Lagrange Basis nehmen mit geeigneten Stützstellen und würde für \mathcal{N}^{1D} und \mathcal{W}_N^{1D} Diagonalmatrizen bekommen. Wie wir später sehen werden, ist es sogar möglich die Einheitsmatrix für das Produkt dieser beiden Matrizen zu erhalten.

Was uns letztlich interessiert ist die Auswertung der Pseudoinversen an einem Vektor u also M^+u und V^+u . Wenn wir uns später diese Strukturen betrachten, werden wir ein Ergebnis aus der Stabilitätstheorie verwenden können, um das Problem mit der Laplace Bilinearform zu lösen. Zu dem 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 Masse Matrix und zur Steifigkeitsmatrix der Laplace Bilinearform effizient berechnen können. Es sei $\mathfrak{X} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_K}$ ein Tensor. Dann können wir mit der HOSVD diesen Tensor zer-

legen

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

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 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

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

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

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

Diese Transformation von Matrix zu Tensor ist also eigentlich eine Abbildung die einen 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

$$\begin{aligned} p : i &\rightarrow (i_1, i_2), \\ p : j &\rightarrow (j_1, j_2). \end{aligned}$$

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

Das Inverse der Transformation ist gegeben durch

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

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

$$i \pmod{(N + 1)} = p^{-1}(i_1, i_2) \pmod{(N + 1)} = i_1 + \underbrace{(N + 1)i_2}_0 \pmod{(N + 1)} \quad (3.22)$$

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

$$i \pmod{(N + 1)} = i_1 \pmod{(N + 1)} = i_1. \quad (3.23)$$

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

$$i_2 = \frac{i - i_1}{N + 1}. \quad (3.24)$$

Mit dem Wissen über i_1 können wir dies weiter umformen zu

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

Damit haben wir unsere Transformation p

$$p(i) = \left(i \pmod{(N + 1)}, \frac{i - (i \pmod{(N + 1)})}{N + 1} \right) \quad (3.26)$$

gefunden und eindeutig festmachen können, welches Element der Matrixform zu welchem Element der Tensorform gehört. Durch die Definition der Transformationen, können wir diese zu Hilfe nehmen für unser Tensor-Tensor Produkt. Vorher sollten

wir uns das Matrix-Matrix Produkt als Motivation 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

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

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

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

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

Definition 3.2. (*Tensor-Tensor Produkt*)

Es seien $\mathbf{X}^1 \in \mathbb{R}^{I_1 \times I_2 \times I_1 \times I_2}$ und $\mathbf{X}^2 \in \mathbb{R}^{I_1 \times I_2 \times I_1 \times I_2}$ Tensoren. Dann definieren wir das Produkt dieser beiden Tensoren elementweise wie folgt

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

Von der Komplexität her, ist das Tensor-Tensor-Produkt der Masse-Tensoren bzw. der lokalen Laplace-Tensoren genau so komplex, wie das Produkt der korrespondierenden Matrizen. Dazu später mehr in Kapitel 4.

Es wird noch der Operator des Transponierens für Tensoren gebraucht. Analog zum 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

$$A_{ij}^T = A_{ji}. \quad (3.30)$$

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

$$A_{p(i)p(j)}^T = A_{i_1 i_2 i_1 i_2}^T = A_{j_1 j_2 i_1 i_2} = A_{p(j)p(i)}. \quad (3.31)$$

Wir können nun die Moore Penrose Pseudoinverse Eigenschaften auch für Tensoren angeben. Vorher sollte aber das Problem mit der Maschinengenauigkeit ange-

sprochen 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. $ttp(A, ttp(A^+, A)) - A < \epsilon$
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 Erwägung 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 $\mathbf{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3 \times I_4}$ mit

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

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

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

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

Man könnte folgenden Beweisansatz nehmen. Es sei $\mathbf{X} \in \mathbb{R}^{I \times I \times I \times I}$ ein Tensor der Ordnung 4 und $U \in \mathbb{R}^{I \times I}$ eine orthogonale Matrix. Dann gilt es für $n \in \{1, \dots, 4\}$

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.

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

Den letzten Schritt können wir machen, weil U orthogonal ist. Es fehlt noch die Vertauschung von den beiden Matrizen, das heißt wir brauchen Kommutativität. Das dürfen wir aber nicht, denn $\mathcal{X}_{(n)}^+ \in \mathbb{R}^{I^3 \times I}$ und $U^+ \in \mathbb{R}^{I \times I}$. Also wäre die Matrixmultiplikation gar nicht wohldefiniert. Das liefert uns aber keinen Widerspruch zu unserem Ergebnis, da gilt:

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

Denn Links berechnen wir die Pseudoinverse des Tensors. Wie die Pseudoinverse davon definiert ist, haben wir uns für unsere Anwendung definiert und ist nur für unsere Anwendung sinnvoll. Rechts nehmen wir die Pseudoinverse einer Matrix, das heißt das sind zwei verschiedene Pseudoinversen Operatoren.

Das Problem ist, dass wenn man mit dem entfalteten Tensor rechnet, 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 wir nicht haben. Daher nehmen wir es einfach an, dass das Ergebnis stimmt und rechnen weiter. Wir bekommen nun folgende Darstellung der Pseudoinversen

$$\mathbf{x}^+ = \mathbf{g}^+ \times_{n=1}^4 U^{(n)+}. \quad (3.34)$$

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

$$\mathbf{x}^+ = \mathbf{g}^+ \times_{n=1}^4 U^{(n)T} \quad (3.35)$$

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 mittels 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 die Pseudoinversen herzuleiten. In beiden Möglichkeiten fanden wir das 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 sich wiederholende Strukturen 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

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

Nun müssen wir die Informationen der Matrix B noch mit reinbringen. Dazu berechnen wir wie in [Tea] vorgeschlagen z_i mit

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

Der komplette Algorithmus ist nachfolgend dargestellt.

```

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

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

Doch für den zweiten Ansatz über die HOSVD, werden wir eine erweiterte Form des Algorithmus brauchen, da wir mehrere Kronecker Produkte bekommen. Außerdem kann der folgende Algorithmus dann auch für dreidimensionale Ansatzfunktionen bei dem 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 $v \in \mathbb{R}^{N^3}$.

$$z = \begin{pmatrix} c_{11}b_{11}\mathcal{A} & \dots & c_{11}b_{1N}\mathcal{A} & \dots & \dots & c_{1N}b_{11}\mathcal{A} & \dots & c_{1N}b_{1N}\mathcal{A} \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \vdots \\ c_{11}b_{N1}\mathcal{A} & \dots & c_{11}b_{NN}\mathcal{A} & \dots & \dots & c_{1N}b_{N1}\mathcal{A} & \dots & c_{1N}b_{NN}\mathcal{A} \\ c_{21}b_{11}\mathcal{A} & \dots & c_{21}b_{1N}\mathcal{A} & \dots & \dots & c_{2N}b_{11}\mathcal{A} & \dots & c_{2N}b_{1N}\mathcal{A} \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \vdots \\ c_{N1}b_{N1}\mathcal{A} & \dots & c_{N1}b_{NN}\mathcal{A} & \dots & \dots & c_{NN}b_{N1}\mathcal{A} & \dots & c_{NN}b_{NN}\mathcal{A} \end{pmatrix} v \quad (4.1)$$

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

$$z = \begin{pmatrix} c_{11}\textcolor{green}{b}_{11}\textcolor{red}{A} & \dots & c_{11}\textcolor{green}{b}_{1N}\textcolor{red}{A} & \dots & \dots & c_{1N}\textcolor{green}{b}_{11}\textcolor{red}{A} & \dots & c_{1N}\textcolor{green}{b}_{1N}\textcolor{red}{A} \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \vdots \\ c_{11}\textcolor{green}{b}_{N1}\textcolor{red}{A} & \dots & c_{11}\textcolor{green}{b}_{NN}\textcolor{red}{A} & \dots & \dots & c_{1N}\textcolor{green}{b}_{N1}\textcolor{red}{A} & \dots & c_{1N}\textcolor{green}{b}_{NN}\textcolor{red}{A} \\ c_{21}\textcolor{green}{b}_{11}\textcolor{red}{A} & \dots & c_{21}\textcolor{green}{b}_{1N}\textcolor{red}{A} & \dots & \dots & c_{2N}\textcolor{green}{b}_{11}\textcolor{red}{A} & \dots & c_{2N}\textcolor{green}{b}_{1N}\textcolor{red}{A} \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \vdots \\ c_{N1}\textcolor{green}{b}_{N1}\textcolor{red}{A} & \dots & c_{N1}\textcolor{green}{b}_{NN}\textcolor{red}{A} & \dots & \dots & c_{NN}\textcolor{green}{b}_{N1}\textcolor{red}{A} & \dots & c_{NN}\textcolor{green}{b}_{NN}\textcolor{red}{A} \end{pmatrix} v \quad (4.2)$$

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 welchen Spalteneintrag von C , der Zweite in welchen Spalteneintrag von B und der Dritte in welchen Spalteneintrag von A wir uns befinden. Dies kann man wieder als eine Index-Transformation ansehen, die bestimmte Einträge von v eindeutig auf Tensorelemente abbildet. Wir schauen uns erstmal die einzelnen Einträge von z an.

$$z_1 = \mathcal{V}(1, 1, 1)c_{11}\textcolor{green}{b}_{11}\textcolor{red}{a}_{11} + \dots + \mathcal{V}(1, 1, N)c_{11}\textcolor{green}{b}_{11}\textcolor{red}{a}_{1N} + \dots + \mathcal{V}(1, N, 1)c_{11}\textcolor{green}{b}_{1N}\textcolor{red}{a}_{11} \\ + \dots + \mathcal{V}(N, 1, 1)c_{1N}\textcolor{green}{b}_{11}\textcolor{red}{a}_{11} + \dots + \mathcal{V}(N, N, N)c_{1N}\textcolor{green}{b}_{1N}\textcolor{red}{a}_{1N}$$

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

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

Damit haben wir uns die sich wiederholende Struktur von der Matrix $\textcolor{red}{A}$ zu nutze gemacht. Im nächsten Schritt machen wir uns die sich wiederholende Struktur von

b_{ij} zu nutze. Wir definieren hierfür $\mathcal{W}_{\mathbf{1},k}(i) := w_k(i, 1)b_{\mathbf{1}1} + \cdots + w_k(i, N)b_{\mathbf{1}N}$. Damit erhalten wir

$$z_1 = \mathcal{W}_{1,1}(1)c_{11} + \cdots + \mathcal{W}_{1,1}(N)c_{1N}. \quad (4.3)$$

Wir formen z genau so wie wir das auch für v gemacht haben und erhalten dafür den Tensor \mathcal{Z} . Damit erhalten wir für allgemeines z_i folgende Formel

$$\mathcal{Z}(i, j, k) = \mathcal{W}_{j,k}(1)c_{i1} + \cdots + \mathcal{W}_{j,k}(N)c_{iN}, \quad (4.4)$$

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
       $w_k(i, j) = \mathcal{V}(i, j, 1)a_{k1} + \dots + \mathcal{V}(i, j, N)a_{kN}$ 
    end for
  end for
end for
for k=1 < N do
  for i= 1 < N do
    for j= 1 < N do
       $\mathcal{W}_{i,j}(k) := w_j(k, 1)b_{i1} + \dots + w_j(k, N)b_{iN}$ 
    end for
  end for
end for
for k=1 < N do
  for i= 1 < N do
    for j= 1 < N do
       $\mathcal{Z}(i, j, k) = \mathcal{W}_{j,k}(1)c_{i1} + \dots + \mathcal{W}_{j,k}(N)c_{iN}$ 
    end for
  end for
end for

```

Wir haben in (4.2) eine Matrix-Vektor Multiplikation von einer Matrix der Größe $N^3 \times N^3$. Dementsprechend hätten wir N^6 Multiplikationen und N^6 Additionen. Dies entspricht $2N^6$ elementare Operationen. Die Komplexität des vorgeschlagenen Algorithmuses reduziert die Operationen auf $3N^4$ Multiplikationen und genau so viele Additionen. Somit haben wir insgesamt $6N^4$ Operationen. Das ist eine beträchtliche 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 Tensorstruktur aus Kapitel 3 für die lokale Massematrix hergeleitet. Nun schauen wir uns das Matrix-Vektor Produkt mit den Pseudoinversen als Matrix und einen beliebigen Vektor u .

Masse Matrix

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

Wie komplex ist es die Pseudoinversen der Matrizen $(\mathcal{W}_N^{1D}(\mathcal{N}^{1D})^T)$ 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 $m \times n$ -Matrix M mit Rang r , wobei U eine orthogonale $m \times m$ -Matrix, V^T die Transponierte einer orthogonalen $n \times n$ -Matrix V und Σ eine reelle $m \times n$ -Diagonalmatrix ist.

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, wenn $m = n$. Gauß Algorithmus gibt uns eine Komplexität von $O(n^3)$.

Neumann Reihe macht keinen Sinn, da die Komplexität mindestens genau so hoch ist wie bei Gauß.

Gauß Algorithmus macht natürlich nur 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 $x = \{x_1, \dots, x_n\}$ die Stützstellen und $w = \{w_1, \dots, w_n\}$ die Gewichte der Quadratur.

Wir wählen

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

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} eine Diagonalmatrix ist, mit den Diagonaleinträgen $\frac{1}{\sqrt{w_i}}$, gilt $\mathcal{N} = (\mathcal{N})^T$. Weiterhin ist \mathcal{W} auch eine Diagonalmatrix mit den Einträgen w_i in der Diagonalen. In der Ergebnismatrix steht dann in der Diagonalen

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

□

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

- 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.
- 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 wir noch weiter an Komplexität sparen können 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

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

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 geht, wie in Beweis für Bemerkung (4.1).

Wir erhalten folgende vereinfachte Form.

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

Mit der in Kapitel 4.1 besprochenen Strategie, können wir den Vektor u in eine Matrix umformen, sodass wir (4.2.1) äquivalent umformen können zu

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

wobei U und V definiert sind durch $U_{ij} = u(q_i, q_j)$ und $V_{ij} = v(q_i, q_j)$

Dies können wir weiter vereinfachen.

$$V = (A \otimes I_n)U + (I_n \otimes A)U = AU I_n + I_n U A^T = AU + U A^T$$

Dies ist gerade die Lyapunow Gleichung aus der Stabilitätstheorie. Zu der 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 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 $\mathcal{X} \in \mathbb{R}^{I_1 \times \dots \times I_4}$ ein Tensor. Der Tensor \mathcal{X} könnte der Massentensor sein oder der Laplace Bilinearform Tensor. Da die Massematrix und die Elementsteifigkeitsmatrix quadratisch sind, gilt $N := I_1 = \dots = I_4$. Die Formel für die Pseudoinverse lautet

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

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.

1. Für den ersten Schritt des Algorithmuses müssen wir alle *mode* – *k* Entfaltungen berechnen. Dies impliziert eine Ordnung jedes Element des Tensors in ein Element der Ergebnismatrix. Damit haben wir genau so viele Zuweisungen, wie 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. Für jede SVD haben wir die Komplexitätsklasse $O(N^4)$. Insgesamt $O(4N^4)$.
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 genau so viele Multiplikationen. Davon müssen wir vier berechnen. Insgesamt haben wir eine Komplexität von $8N^5$ Rechenoperationen.

Insgesamt ergibt sich eine Komplexität von $O(8N^4 + 8N^5)$. Die Berechnung der orthogonal Matrizen für die HOSVD zeigt, dass alle Matrizen gleich sind. Dementsprechend müssen wir im zweiten Schritt die SVD nur einmal berechnen. Damit reduziert sich die Komplexität zu $O(5N^4 + 8N^5)$.

Im nächsten Schritt betrachten wir das Matrix-Vektor Produkt mit einem beliebigen Vektor $u \in \mathbb{R}^{N^3}$ und überlegen wie wir uns die Strukturen dort zu nutze machen.

4.3.1 Naiver Ansatz

Wir schauen uns zu erst einen naiven Ansatz an. Dafür benötigen wir jedoch noch 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 uns das Ganze nun 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 $Y = \text{tvp}(\mathcal{A}, U)$ 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

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

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 . Mit der Berechnung der HOSVD ergibt dieser Ansatz eine Komplexität von $O(6N^4 + 8N^5)$.

4.3.2 Entfaltete HOSVD

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

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

Die Äquivalenz folgt mit Lemma (2.23).

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

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

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

$\underbrace{\hspace{15em}}_{K_3}$

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

Wie schaffen wir es den vollbesetzten Kerntensor zu invertieren bzw. die Pseudoinverse herzuleiten? Wie wir wissen, müssen wir dazu erstmal Zahlen deren Betrag kleiner ist als $\epsilon > 0$ auf Null setzen und die restlichen Elemente einzeln invertieren. Dazu müssen wir, da der Kerntensor vollbesetzt ist, durch alle Elemente des Tensors durch-iterieren. Das heißt, wir haben eine Komplexität von $O(N^4)$, da unser Tensor Ordnung 4 hat. Es folgt eine letzte Matrix-Matrix Multiplikation $U^{(n)^T} \mathcal{G}_{(n)}^+$. Wie groß sind diese Matrizen? Die Matrix $\mathcal{G}_{(n)}^+$ hat die Größe $N \times N^3$. Die Matrix $U^{(n)^T}$ hat die Größe $N \times N$. Wir haben bei der Multiplikation dieser beiden Matrizen N^6 Multiplikationen und genau so viele Additionen. Dann können wir die Komplexitätsklassen durch $O(N^6)$ angeben. Die Anzahl der Operationen steigt mit raketengeschwindigkeit durch diese Matrix-Matrix Multiplikation. 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 zusammen zu rechnen durch ein Matrix-Matrix Produkt. 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 ist N^{10} .

Insgesamt haben wir

- $O(N^4)$ für die Berechnung von K_1 ,
- $O(6N^4)$ für die Berechnung von K_2 .
- $O(N^{10})$ für die Berechnung von K_3 .

Zusammengefasst $O(7N^4 + N^{10})$.

Dies ist aber nicht annähernd so gut wie Option 1. Man bemerke, dass wir die

Komplexität von der Berechnung der HOSVD noch nicht dazu addiert haben. 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 einfach nur die Diagonalelemente für die Multiplikation in Erwägung ziehen. Damit sparen wir uns $O(N^4)$ Operationen.
2. Was wir tun könnten ist den Kerntensor in K_2 reinmultiplizieren bevor wir das Kronecker Produkt ausrechnen (siehe dazu die Bemerkung (4.3)). Selbst wenn wir das hinkriegen, würden wir es nicht hinkommen die Komplexität von Option 1 zu übertreffen, denn die Berechnung von K_3 würde uns immer noch einen gewaltigen Strich durch die Rechnung machen.
3. Eine letzte Alternative ist die trunkierte HOSVD.

Wir wollen für eine Effizienzsteigerung folgende Bemerkung nutzen.

Bemerkung 4.3. Sei A eine Diagonalmatrix mit $A \in \mathbb{R}^{n^2 \times n^2}$, $C, B \in \mathbb{R}^{n \times n}$ und $y \in \mathbb{R}^{n^2}$. Dann gilt:

$$A(B \otimes C)y = (B \otimes C) \begin{pmatrix} a_{11}y_1 \\ \vdots \\ a_{n^2 n^2}y_{n^2} \end{pmatrix}$$

Beweis. Folgt aus der Kommutativität der Matrixmultiplikation für Diagonalmatrizen. □

Das Problem ist, dass unser entfalteter Kerntensor keine Diagonalmatrix ist. Der Kerntensor ist an sich zwar super-diagonal, bloß hängt es von der Entfaltung ab, ob der entfaltete Tensor eine Diagonalmatrix ist. Der entfaltete Kerntensor ist genau dann eine Diagonalmatrix, wenn wir die natürliche Entfaltung wählen. Mit dieser Entfaltung ist die Entfaltung gemeint mit Hilfe der Index Transformation $p(\cdot)$.

Selbst wenn wir es schaffen dies effizient zu berechnen und sogar noch eine trunkierte HOSVD wählen. Ist das Ergebnis das was wir wollen? 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 Ausnutzung der Tensorstruktur.

5 Resultate

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

Es ist naheliegend, dass man den Ansatz mit 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 sich nicht rentiert.

Der erste Ansatz versucht die Elementsteifigkeitsmatrizen erst in eine Tensorstruktur zu überführen. Dieser Ansatz ist nicht universell und muss für jede Bilinearform extra hergeleitet werden. Mit Hilfe eines Algorithmuses 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 $O(N^{10})$, wobei N der Polynomgrad ist.

Zu dem 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 so groß ist, dass sie nicht mehr im Cache abgelegt werden kann, mit einem überproportionalen Zeitaufwand verbunden ist. Dies sollte gemieden werden.

Man kann auf die Ergebnisse dieser Arbeit aufbauend versuchen im ersten Ansatz die Komplexität noch weiter zu drücken. 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.