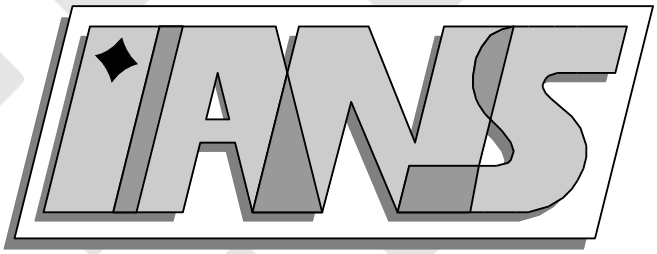
Universität Stuttgart



Hauptseminar Hierarchische Matrizen

Olaf Steinbach (ed.)

Berichte aus dem Institut für Angewandte Analysis und Numerische Simulation

Seminarbericht 2004/013

Universität Stuttgart

Hauptseminar Hierarchische Matrizen

Olaf Steinbach (ed.)

Berichte aus dem Institut für Angewandte Analysis und Numerische Simulation

Seminarbericht 2004/013

Institut für Angewandte Analysis und Numerische Simulation (IANS)
Fakultät Mathematik und Physik
Fachbereich Mathematik
Pfaffenwaldring 57
D-70 569 Stuttgart

E-Mail: ians-preprints@mathematik.uni-stuttgart.de
WWW: http://preprints.ians.uni-stuttgart.de

ISSN **1611-4176**

© Alle Rechte vorbehalten. Nachdruck nur mit Genehmigung des Autors.
IANS-Logo: Andreas Klimke. LaTeX-Style: Winfried Geis, Thomas Merkle.

Vorwort

Der vorliegende Seminarbericht entstand im Rahmen des Hauptseminars *Hierarchische Matrizen*, welches im Sommersemester 2004 am Institut für Angewandte Analysis und Numerische Simulation der Universität Stuttgart stattfand. Dieser Bericht enthält die schriftlichen Zusammenfassungen der neun teilnehmenden Studenten zu ihren Vorträgen.

Thema dieses Hauptseminars sind die von W. Hackbusch in einer Arbeit von 1999 eingeführten Hierarchischen Matrizen zur Approximation vollbesetzter Matrizen. Diese ermöglichen neben einer Datenkompression und somit einer effizienten Abspeicherung der Matrizen auch eine optimale Effizienz bei verschiedenen Matrix–Operationen, insbesondere bei der Matrix–Vektor–Multiplikation und der Invertierung. Hierarchische Matrizen basieren auf einer hierarchischen Partitionierung der Matrix in Blöcke kleinerer Dimension, welche durch Niedrig–Rang–Matrizen approximiert werden.

Ausgehend von einer rein algebraischen Darstellung werden im ersten Teil des Hauptseminars Hierarchische Matrizen eingeführt (Kapitel 1) und die wesentlichen Matrix–Operationen untersucht (Kapitel 3). Das entscheidende Hilfsmittel zur Definition von Niedrig–Rang–Approximationen ist dabei die Singulärwertzerlegung von Matrizen (Kapitel 2). Hierarchische Matrizen können aufgrund ihrer Struktur effizient abgespeichert werden und die auftretenden Matrix–Operationen können entsprechend effizient implementiert werden (Kapitel 4).
Aus algebraischer Sichtweise kann die Approximierbarkeit durch Niedrig–Rang–Matrizen als eine abstrakte Voraussetzung formuliert werden. Im zweiten Teil des Hauptseminars stehen Anwendungen im Vordergrund, die eine Approximation durch Hierarchische Matrizen a priori gewährleisten. Im Blickpunkt stehen dabei numerische Näherungsverfahren zur Lösung elliptischer Randwertprobleme. Schnelle Randelementmethoden beruhen auf einer geeigneten Aufspaltung der Fundamentallösung zur Trennung der Integrationsvariablen. Eine Möglichkeit ist dabei die Taylorentwicklung (Kapitel 5). Hierarchische Matrizen finden verstärkt auch Anwendungen bei finiten Elementen, z.B. bei Eigenwertproblemen (Kapitel 6) oder der Approximation der inversen FEM Steifigkeitsmatrix (Kapitel 7). Alternativ zur Berechnung der Niedrig–Rang–Approximation durch die Singulärwertzerlegung kann die Adaptive Cross Approximation verwendet werden (Kapitel 8). Abschliessend werden neuere Varianten hierarchischer Matrizen betrachtet (Kapitel 9).

Grundlage für die Vorträge und die schriftlichen Ausarbeitungen waren neben dem Skript *Hierarchische Matrizen* von S. Börm, L. Grasedyck und W. Hackbusch vor allem Originalarbeiten zu diesen Themen.

Ich hoffe, daß dieses Hauptseminar allen Beteiligten Spaß und Freude gemacht hat und daß dieser Seminarbericht gerne und oft daran erinnert. Mein Dank gilt allen Teilnehmern für

Ihren Einsatz bei der Vorbereitung der Vorträge und den schriftlichen Ausarbeitungen. Besonderer Dank gebührt jedoch Dipl.–Math. Jens Breuer und Dipl.–Math. Günther Of für die tatkräftige Unterstützung bei der Vorbereitung des Hauptseminars und dem Erstellen dieser schriftlichen Ausarbeitungen.

Stuttgart, Juli 2004 Olaf Steinbach

Inhaltsverzeichnis

Kapitel 1

Einführung in Hierarchische Matrizen

Michael Speck

1.1 Einleitung

Im Allgemeinen sind für eine reelle vollbesetzte $n \times n$-Matrix A der Speicherbedarf und die Kosten einer Matrix-Vektor-Multiplikation durch $\mathcal{O}(n^2)$ gegeben. Hierarchische Matrizen erlauben die Approximation von vollbesetzten Matrizen mit einem weit geringeren Speicherbedarf und optimaler Effizienz bei der Matrix-Vektor-Multiplikation sowie der Invertierung, nämlich mit einem fast linearen Aufwand von $\mathcal{O}(n \log^q n)$.
Abhängig von der Anwendung kann zu einer gegebenen Matrix A eine Hierarchische Matrix $A_{\mathcal{H}}$ so aufgebaut werden, daß die Ordnung des Approximationsfehlers von A nach $A_{\mathcal{H}}$ nicht schlechter ist, als die Ordnung des in der Anwendung gemachten Diskretisierungsfehlers.

1.1.1 Faktorisierung

Die Grundidee für die komprimierte Darstellung ist, daß sich jede $m \times n$-Matrix M mit Rang $r \leq \min(m, n)$ als Produkt zweier Matrizen $A \in \mathbb{R}^{m \times r}$ und $B \in \mathbb{R}^{r \times n}$ schreiben läßt, also $M = AB$ (Abb. 1.1). Durch diese Faktorisierung liegt der Speicherbedarf bei

$$\text{mem}(M) = r(m + n)$$

und im quadratischen Fall $m = n$ bei

$$\text{mem}(M) = 2rn.$$

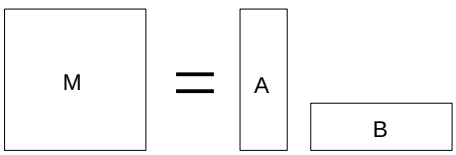


Abbildung 1.1: Faktorisierung einer $m \times n$ Matrix mit Rang r.

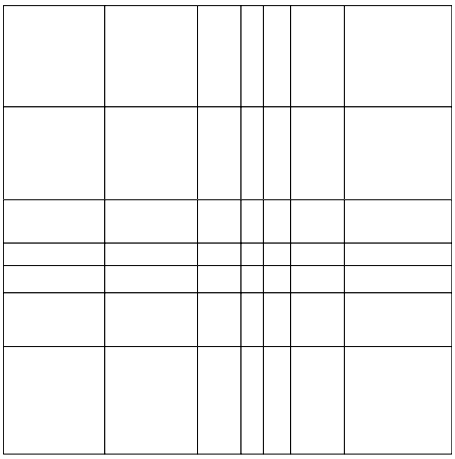


Abbildung 1.2: Partitionierung einer Matrix durch ein Tensorprodukt.

Das ist für $r \ll n$ deutlich besser als der allgemeine Aufwand n^2.
Da bei der Matrix-Vektor-Multiplikation die Multiplikationen gegenüber den Additionen dominieren, ergibt sich der gleiche Aufwand von

$$\mathrm{op}(M) = \mathrm{mem}(M) = r(m+n),$$

beziehungsweise für $m = n$

$$\mathrm{op}(M) = \mathrm{mem}(M) = 2rn.$$

Um eine komprimierte Darstellung einer beliebig gegebenen Matrix M zu erhalten, versucht man jetzt, die Matrix M in Teilmatrizen, auch Blöcke genannt, zu zerlegen, die entweder von vollem Rang, aber kleiner Dimension sind, oder die sich durch eine Faktorisierung *data-sparse*, also mit weniger Platzbedarf abspeichern lassen.

1.1.2 Beispiel 1 (Tensorprodukt)

Sei A eine $n \times n$-Matrix und $I = \{1, \ldots, n\}$ die dazugehörige Indexmenge. Sei $P_I = \{I_j : 1 \leq j \leq k\}$ eine beliebige Partitionierung von I in nichtleere, disjunkte, zusammenhängende Mengen mit der Eigenschaft

$$I = \bigcup_{j=1}^{k} I_j.$$

n_j bezeichne die Anzahl der Elemente in I_j und k die Anzahl der Partitionen in Abhängigkeit von n.
Dann erhält man durch das Tensorprodukt

$$P_T = P_I \times P_I = \{I_i \times I_j : I_i, I_j \in P_I\}$$

eine Partitionierung der Matrix A (Abb. 1.2).
Approximiert man jetzt die Matrix A, indem man für die Diagonalblöcke A_{ii} vollen Rang und für die Nichtdiagonalblöcke A_{ij} $(i \neq j)$ einen niedrigen Rang r annimmt, dann gelangt man zu einer komprimierten Darstellung, da sich alle Nichtdiagonalblöcke faktorisiert abspeichern lassen.

Ungeachtet des Approximationsfehlers, der von der konkreten Matrix und der konkreten Partitionierung abhinge, soll zunächst einmal nur überprüft werden, welche Verbesserung sich im Speicheraufwand ergäbe.
Wegen der Abschätzung

$$\sum_{i=1}^{k} n_i^2 = \Big(\sum_{i=1}^{k} (\frac{1}{\sqrt{k}})^2\Big)\Big(\sum_{i=1}^{k} n_i^2\Big) \geq \Big(\sum_{i=1}^{k} \frac{1}{\sqrt{k}} n_i\Big)^2 = \frac{1}{k}\Big(\sum_{i=1}^{k} n_i\Big)^2 = \frac{1}{k} n^2$$

gilt für den Speicherbedarf

$$\begin{aligned}
\mathrm{mem}(A) &= \sum_{i=1}^{k} \mathrm{mem}(A_{ii}) + \sum_{\substack{i,j=1 \\ i \neq j}}^{k} \mathrm{mem}(A_{ij}) \\
&= \sum_{i=1}^{k} n_i^2 + \sum_{\substack{i,j=1 \\ i \neq j}}^{k} (n_i + n_j) r \\
&= \sum_{i=1}^{k} n_i^2 + 2r \sum_{\substack{i=2 \\ j<i}}^{k} (n_i + n_j) \\
&= \sum_{i=1}^{k} n_i^2 + 2r(k-1)n \\
&\geq \frac{1}{k} n^2 + 2r(k-1)n \\
&= \frac{1}{k} n^2 + 2rkn - 2rn.
\end{aligned}$$

Die ersten beiden Terme bestimmen die Ordnung und man sieht, daß diese minimal wird, wenn man $k(n)$ proportional zu $\sqrt{n}$ wählt. Das ergibt dann aber immer noch $\mathcal{O}(n^{3/2})$.
Wählt man also für die Partitionierung der Matrix ein Tensorprodukt, dann ist der Speicherbedarf nicht optimal, das heißt nicht $\mathcal{O}(n \log^q n)$.

1.1.3 Beispiel 2 (Hierarchische Partitionierung 1)

Sei A eine $n \times n$-Matrix mit $n = 2^p$ für $p \in \mathbb{N}_0$ und $I = \{1, \ldots, n\}$ wieder die dazugehörige Indexmenge. Mit der Zerlegung

$$I = \{1, \ldots, n\} = \{1, \ldots, \frac{n}{2}\} \cup \{\frac{n}{2} + 1, \ldots, n\} =: I_1 \cup I_2$$

sei jetzt die Partitionierung von A rekursiv gegeben durch die Vorschrift

$$P_H(I \times I) := P_H(I_1 \times I_1) \cup \{I_1 \times I_2\} \cup \{I_2 \times I_1\} \cup P_H(I_2 \times I_2),$$

mit der Abbruchsbedingung $P_H(J \times J) := J \times J$, falls $|J| = 2^l$ für ein $l \in \mathbb{N}_0$ ist.
Die Matrix A wird also in vier Blöcke der Größe $\frac{n}{2} \times \frac{n}{2}$ zerlegt. Die Diagonalblöcke werden analog zerlegt und die neuen Diagonalblöcke ebenfalls und so weiter, bis die Größe der Diagonalblöcke nur noch $2^l \times 2^l$ beträgt, für ein kleines, festes $l \in \mathbb{N}_0$ (Abb. 1.3).
Bei $l = 0$ zum Beispiel wird die Partitionierung rekursiv fortgesetzt, bis die Diagonalblöcke 1×1-Matrizen sind.

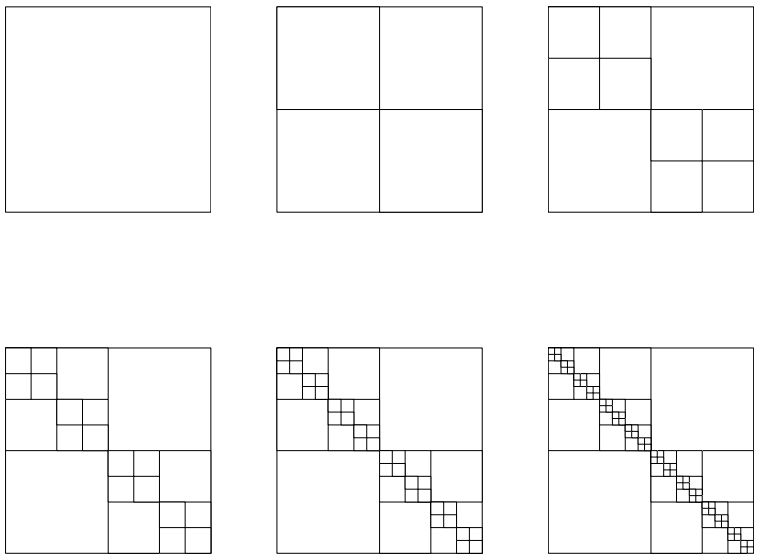


Abbildung 1.3: Die hierarchische Partitionierung P_H einer Matrix.

Jetzt wird die Matrix A analog dem Tensorproduktbeispiel approximiert, indem man für die Diagonalblöcke vollen Rang und für die Nichtdiagonalblöcke einen niedrigen Rang r annimmt. Zur einfacheren Darstellung bezeichne D_k einen der 2^k Diagonalblöcke nach dem k-ten Rekursionsschritt. Es reicht, einen Vertreter zu wählen, da alle $2^{-k}n \times 2^{-k}n$-Matrizen vollen Ranges sind, also den gleichen Speicherbedarf von

$$\text{mem}(D_k) = (2^{-k}n)^2$$

haben. Analog sei N_k einer der Nichtdiagonalblöcke, die aus k Verfeinerungsschritten resultieren. Diese haben jeweils den Speicherbedarf

$$\text{mem}(N_k) = 2r(2^{-k}n).$$

Für den Speicherbedarf nach dem ersten Rekursionsschritt gilt dann

$$\text{mem}(A) = 2\ \text{mem}(D_1) + 2\ \text{mem}(N_1).$$

Im zweiten Rekursionsschritt werden die Diagonalblöcke erneut zerlegt, so daß sich

$$\text{mem}(A) = 4\ \text{mem}(D_2) + 4\ \text{mem}(N_2) + 2\ \text{mem}(N_1)$$

ergibt. Nach dem dritten Schritt gilt

$$\text{mem}(A) = 8\ \text{mem}(D_3) + 8\ \text{mem}(N_3) + 4\ \text{mem}(N_2) + 2\ \text{mem}(N_1).$$

Der Speicheraufwand für die Nichtdiagonalblöcke aus vorherigen Rekursionsschritten bleibt also immer gleich und für jeden neuen Schritt k ergeben sich aus der Aufspaltung der Diagonalblöcke 2^k Blöcke der Größe $2^{-k}n \times 2^{-k}n$ mit einem niedrigen Rang r. Aus dem letzten Schritt, wenn die Rekursion terminiert, resultieren immer 2^{p-l} Blöcke der Größe $2^l \times 2^l$ mit vollem Rang.

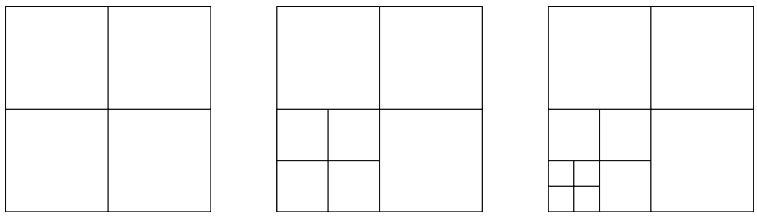


Abbildung 1.4: Die hierarchische Partitionierung P_{NO}.

Daraus ergibt sich insgesamt der Speicherbedarf (und analog der Aufwand der Matrix-Vektor-Multiplikation) als

$$\begin{aligned}
\mathrm{mem}(A) &= 2^{p-l}\mathrm{mem}(D_{p-l}) + \sum_{k=1}^{p-l} 2^k \mathrm{mem}(N_k) \\
&= 2^{p-l}2^{2l} + \sum_{k=1}^{p-l} 2^k (2r2^{-k}n) \\
&= 2^p 2^l + \sum_{k=1}^{p-l} 2rn \\
&= 2^l n + (p-l)2rn \\
&= 2^l n + 2rnp - 2lrn \\
&= 2rn\log_2 n + n(2^l - 2lr) \\
&= \mathcal{O}(rn\log_2 n).
\end{aligned}$$

Der Speicheraufwand und die Kosten der Matrix-Vektor-Multiplikation für diese hierarchische Partitionierung liegen also bei $\mathcal{O}(rn\log_2 n)$.

1.1.4 Beispiel 3 (Hierarchische Partitionierung 2)

Das folgende Beispiel hat eine feinere Partitionierung bei gleicher Ordnung der Komplexität. Seien A eine $n \times n$-Matrix mit $n = 2^p$ für $p \in \mathbb{N}_0$, $I = \{1, \ldots, n\}$ und die Zerlegung

$$I = \{1, \ldots, n\} = \{1, \ldots, \frac{n}{2}\} \cup \{\frac{n}{2} + 1, \ldots, n\} =: I_1 \cup I_2,$$

wie aus dem zweiten Beispiel.
Zusätzlich betrachtet man jetzt jeweils eine Partitionierung für die oberen und unteren Nebendiagonalblöcke: (Abb. 1.4 und 1.5)

$$P_{NO}(I \times J) := \{I_1 \times J_1\} \cup \{I_1 \times J_2\} \cup P_{NO}(I_2 \times J_1) \cup \{I_2 \times J_2\},$$

und

$$P_{NU}(I \times J) := \{I_1 \times J_1\} \cup P_{NU}(I_1 \times J_2) \cup \{I_2 \times J_1\} \cup \{I_2 \times J_2\}.$$

Dann sei die Partitionierung von A rekursiv gegeben durch die Vorschrift

$$P_{H'}(I \times I) := P_{H'}(I_1 \times I_1) \cup P_{NO}(I_1 \times I_2) \cup P_{NU}(I_2 \times I_1) \cup P_{H'}(I_2 \times I_2).$$

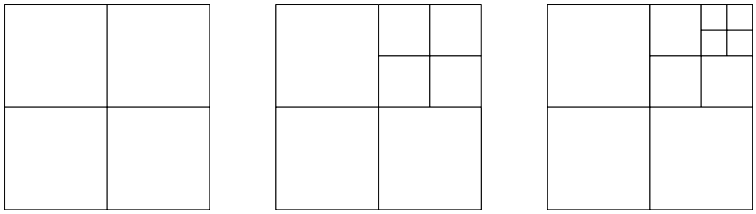


Abbildung 1.5: Die hierarchische Partitionierung P_{NU}.

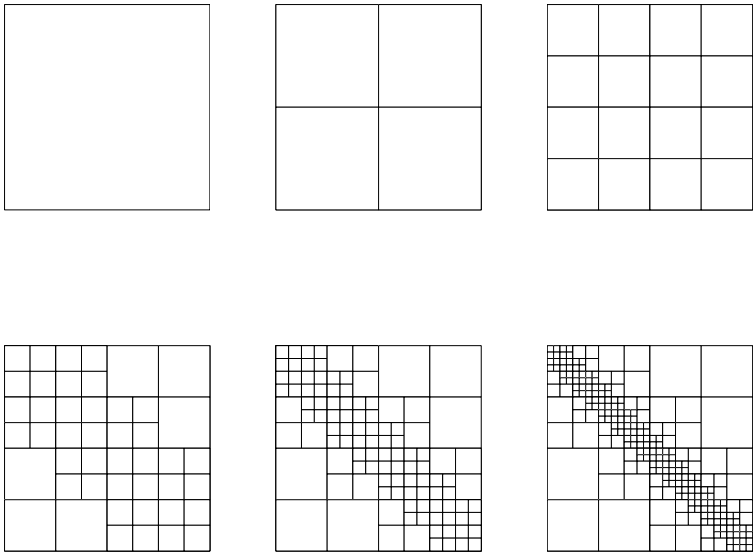


Abbildung 1.6: Die feinere hierarchische Partitionierung $P_{H'}$ einer Matrix.

mit den Abbruchsbedingungen $P_{H'}(J \times J) := J \times J$, $P_{NO}(J \times K) := J \times K$, $P_{NU}(J \times K) := J \times K$, falls $|J| = 2^l$ für ein $l \in \mathbb{N}_0$ ist[1] (Abb. 1.6).
In jedem Rekursionsschritt werden also die Diagonal- und die Nebendiagonalblöcke in vier gleichgroße Blöcke zerteilt, bis deren Blockgröße nur noch $2^l \times 2^l$ beträgt.

1.2 Hierarchische Partitionierungen

1.2.1 Partitionierung der Indexmenge

Wie das Beispiel des Tensorprodukts gezeigt hat, reicht eine einfache Partitionierung der Indexmenge $I = \{1, \ldots, n\}$ in disjunkte Blöcke nicht aus. Stattdessen benötigt man eine hierarchische Unterteilung, die in der Form eines Baumes verwaltet wird [7]. Die Struktur der Unterteilung hängt dabei von dem zugrundeliegenden Problem ab.

Definition 1.1 (Baum). Sei V eine Menge von Punkten und $E \subseteq V \times V$ eine Menge von gerichteten Kanten zwischen den Punkten. Dann heißt ein Tupel $T = (r, V, E)$ Baum mit der Wurzel $r \in V$, falls es für jeden Punkt $v \in V(T)$ genau einen Pfad $(v_0, v_1, \ldots, v_l) \in V^{l+1}$ mit $l \in \mathbb{N}_0$, $v_0 = r$, $v_l = v$ und $(v_{i-1}, v_i) \in E$ für $i = 1, 2, \ldots, l$ gibt.

[1] Es gilt immer $|J| = |K|$, da die betrachteten Matrizen quadratisch sind.

Die Pfadlänge l wird als Level bezeichnet und es folgt, daß auf Level 0 nur die Wurzel r liegt. Die Punkte t der Menge V heißen Knoten des Baumes T. Statt $t \in V(T)$ wird abkürzend die Schreibweise $t \in T$ verwendet. Die Menge der Söhne eines Knoten t ist definiert durch

$$\mathcal{S}(t) := \{s \in T : (t, s) \in E\}.$$

Falls $\mathcal{S}(t) = \emptyset$, dann ist t ein Blatt von T.

Definition 1.2 ($\mathcal{H}$-Baum). Sei I eine Indexmenge. Ein Baum T heißt $\mathcal{H}$-Baum von I, falls gilt

- $\forall t \in T : \emptyset \neq t \subseteq I$
- $I \in T$
- $\forall t \in T : |\mathcal{S}(t)| \neq 1$
- Wenn $t \in T$ kein Blatt ist, dann sind dessen Söhne disjunkte Teilmengen von I mit

$$t = \bigcup_{s \in \mathcal{S}(t)} s.$$

Es folgt, daß I die Wurzel von T ist.

Beispiel 1.1. Wie in dem zweiten und dritten Beispiel der Einleitung bereits verwendet, erhält man für eine Indexmenge $I = \{1, \ldots, n\}$ mit $n = 2^p$ einen $\mathcal{H}$-Baum, wenn man die Zerlegung

$$I = \{1, \ldots, n\} = \{1, \ldots, \frac{n}{2}\} \cup \{\frac{n}{2} + 1, \ldots, n\} =: I_1 \cup I_2$$

rekursiv anwendet. Mit der Schreibweise I_k^d, in der d für die Tiefe, das heißt die Anzahl der Rekursionsschritte, steht und k ein einfacher Laufindex ist, lassen sich die Knoten in diesem Baum schreiben als

$$\begin{aligned}
&I_1^0 = \{1, \ldots, n\} \\
&I_1^1 = \{1, \ldots, \frac{n}{2}\},\ I_2^1 = \{\frac{n}{2} + 1, \ldots, n\} \\
&\ldots \\
&I_1^{p-1} = \{1, 2\},\ \ldots,\ I_{n/2}^{p-1} = \{n-1, n\} \\
&I_1^p = \{1\},\ I_2^p = \{2\},\ \ldots,\ I_n^p = \{n\}.
\end{aligned}$$

Die Verzweigung des Baumes ist aus Abbildung 1.7 ersichtlich.

1.2.2 Zulässige Blöcke

Sei A eine $n \times n$-Matrix, $I = \{1, \ldots, n\}$ die dazugehörige Indexmenge und seien $I_i, I_j \subseteq I$ nicht leer. Dann wird die Teilmatrix $A_{|I_i \times I_j}$ von A als Block bezeichnet.
Allgemein heißt ein Block genau dann (r, ε)-zulässig, wenn er durch eine Matrix mit niedrigem Rang r approximiert werden kann, so daß der Approximationsfehler kleiner ist als ε. Ob ein Block diese Eigenschaft erfüllt, hängt von dem zugrundeliegenden Problem und von der Partitionierung der Indexmenge I ab.

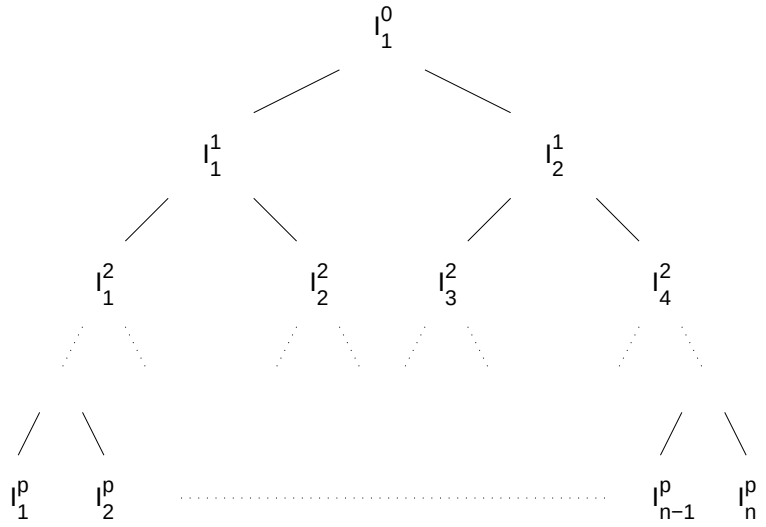


Abbildung 1.7: Binäre, hierarchische Partitionierung der Indexmenge I.

1.2.3 Partitionierung der Matrix

Sei A eine $n \times n$-Matrix, $I = \{1, \ldots, n\}$ die zugehörige Indexmenge und T ein $\mathcal{H}$-Baum von I.
Ein Algorithmus zur hierarchischen Partitionierung der Matrix A ist gegeben durch [5, S. 14]:

1.	$P := \{I \times I\}$
2.	*Wiederhole:* Ersetze jeden nicht (r, ε)-zulässigen Block $J \times K \in P$ mit $\mathcal{S}(J) \times \mathcal{S}(K) = \{J' \times K' \mid J' \in \mathcal{S}(J) \wedge K' \in \mathcal{S}(K)\}$.
3.	*Bis:* Alle Blöcke $J \times K \in P$ sind entweder (r, ε)-zulässig oder $\mathcal{S}(J) = \emptyset$ oder $\mathcal{S}(K) = \emptyset$.

Ausgehend von der vollen Matrix A werden sukzessive alle nicht (r, ε)-zulässigen Blöcke $J \times K$ mit dem Tensorprodukt der Söhne $\mathcal{S}(J)$ und $\mathcal{S}(K)$ aus T ersetzt. Das heißt, es werden immer nur J, K-Paare der gleichen Stufe auf (r, ε)-Zulässigkeit geprüft. Da der Baum T endlich ist, terminiert der Algorithmus. Die nicht (r, ε)-zulässigen Blöcke, das heißt die vollbesetzen, jedoch hinreichend kleinen Matrizen, werden jeweils direkt ausgewertet. Jeder (r, ε)-zulässige Block kann nach Definition durch eine Niedrigrangmatrix hinreichend genau approximiert werden, das heißt diese Blöcke können durch die Auswertung der Niedrigrangmatrix effizient berechnet werden.
Eine Matrix mit dieser Struktur wird als Hierarchische Matrix oder auch als $\mathcal{H}$-Matrix bezeichnet.

1.3 Der Bisektionsalgorithmus

Wie bereits erwähnt ist die Partitionierung der Indexmenge $I = \{1, \ldots, n\}$ problembezogen. Sind die Indizes jedoch mit n Punkten einer m-dimensionalen Punktwolke assoziiert, dann läßt sich eine allgemeine Strategie zur Partitionierung der Indexmenge angeben.

Ausgehend von der Menge aller Punkte $\Omega_1^0 := \{\underline{x}_i \mid i \in I\}$ und der Menge aller Indizes $I_1^0 := I$, die die Wurzel des zu konstruierenden $\mathcal{H}$-Baumes T ist, wird rekursiv jeder Knoten von T in jeweils zwei Söhne zerlegt, bis seine Mächtigkeit, das heißt die Anzahl der Punkte in der assoziierten Punktmenge, eine gewünschte Mindestmächtigkeit erreicht hat. Das Verfahren terminiert, da die Mächtigkeit eines Sohnes echt kleiner als die des Vaters ist.
Um jetzt einen Knoten I_l^d der Mächtigkeit q in seine zwei Söhne I_{2l-1}^{d+1} und I_{2l}^{d+1} zu zerteilen, bestimmt man zunächst den Schwerpunkt

$$\underline{s} := \frac{1}{q} \sum_{k=1}^{q} \underline{x}_k.$$

Dann löst man das Maximierungsproblem

$$\underline{w} := \arg \max_{\substack{\underline{v} \in \mathbb{R}^m \\ \|\underline{v}\|_2 = 1}} \sum_{k=1}^{q} \left(\underline{v}^T (\underline{x}_k - \underline{s})\right)^2,$$

und zerlegt die Mengen I_l^d und Ω_l^d in

$$I_{2l-1}^{d+1} := \{i \in I_l^d \mid \left(\underline{w}^T (\underline{x}_i - \underline{s})\right) \geq 0\}, \quad \Omega_{2l-1}^{d+1} := \{\underline{x}_i \mid i \in I_{2l-1}^{d+1}\}$$

und

$$I_{2l}^{d+1} := \{i \in I_l^d \mid \left(\underline{w}^T (\underline{x}_i - \underline{s})\right) < 0\}, \quad \Omega_{2l}^{d+1} := \{\underline{x}_i \mid i \in I_{2l}^{d+1}\}.$$

Das heißt, der Hauptvektor $\underline{w}$ definiert als Normalenvektor eine Ebene durch den Schwerpunkt $\underline{s}$, die die Punktwolke zerteilt. Die Punkte mit $\left(\underline{w}^T (\underline{x}_i - \underline{s})\right) \geq 0$ liegen oberhalb (relativ zur Richtung des Normalenvektors $\underline{w}$) oder in der Ebene und die Punkte mit $\left(\underline{w}^T (\underline{x}_i - \underline{s})\right) < 0$ liegen unterhalb der Ebene.
Um das Maximierungsproblem zu lösen, wird es zuerst einmal umformuliert. Mit $\underline{a}_k := \underline{x}_k - \underline{s}$ gilt zunächst

$$\begin{aligned}
\sum_{k=1}^{q} (\underline{v}^T \underline{a}_k)^2 &= \sum_{k=1}^{q} (\underline{v}^T \underline{a}_k)(\underline{a}_k^T \underline{v}) \\
&= \sum_{k=1}^{q} \left(\underline{v}^T (\underline{a}_k \underline{a}_k^T) \underline{v}\right) \\
&= \sum_{k=1}^{q} \left(\left((\underline{a}_k \underline{a}_k^T) \underline{v}\right)^T \underline{v}\right) \\
&= \left(\left(\sum_{k=1}^{q} \underline{a}_k \underline{a}_k^T\right) \underline{v}\right)^T \underline{v} = (K\underline{v}, \underline{v})
\end{aligned}$$

mit der $m \times m$-Kovarianzmatrix $K := \sum_{k=1}^{q} (\underline{x}_k - \underline{s})(\underline{x}_k - \underline{s})^T$. Daraus folgt, daß

$$\begin{aligned}
\underline{w} &= \arg \max_{\substack{\underline{v} \in \mathbb{R}^m \\ \|\underline{v}\|_2 = 1}} \sum_{k=1}^{q} \left(\underline{v}^T (\underline{x}_k - \underline{s})\right)^2 \\
&= \arg \max_{\substack{\underline{v} \in \mathbb{R}^m \\ \|\underline{v}\|_2 = 1}} (K\underline{v}, \underline{v}) = \arg \max_{\underline{0} \neq \underline{v} \in \mathbb{R}^m} \frac{(K\underline{v}, \underline{v})}{(\underline{v}, \underline{v})}
\end{aligned}$$

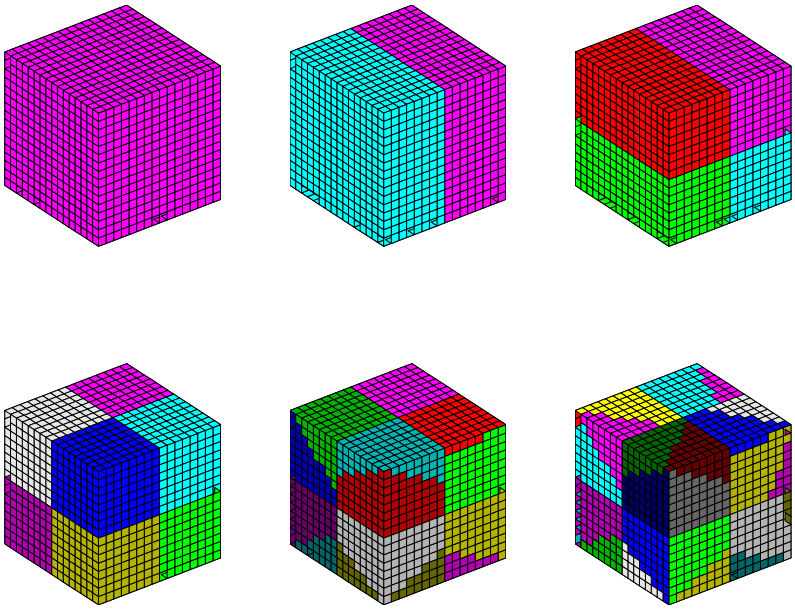


Abbildung 1.8: Bisektion einer Würfeloberfläche.

ist, sich also aus der Maximierung des Rayleigh-Quotienten ergibt. Dieser wird maximal, wenn $\underline{v}$ ein Eigenvektor zum maximalen Eigenwert λ_{max} von K ist. Man erhält also $\underline{w}$ durch Lösen des m-dimensionalen Eigenwertproblems

$$K\underline{w} = \lambda_{max}\underline{w}.$$

Die Zerteilung eines Knotens I_l^d der Mächtigkeit q hat einen Aufwand von $\mathcal{O}(q)$, der sich aus dem Aufwand der Teilschritte ergibt. Die Bestimmung des Schwerpunktes, die Assemblierung der Kovarianzmatrix und die Bildung von I_{2l-1}^{d+1} und I_{2l}^{d+1} ist jeweils $\mathcal{O}(q)$. Da die Dimension der Kovarianzmatrix nicht von n abhängt, ist das Lösen des Eigenwertproblems sogar $\mathcal{O}(1)$. Für die Zerteilung aller Knoten auf einem Level von T beträgt der Aufwand $\mathcal{O}(n)$, da die Knoten disjunkte Teilmengen von I sind. Es gibt aufgrund der Bisektion nur $\mathcal{O}(\log n)$ Level im Baum T. Damit ergibt sich der Gesamtaufwand des Bisektionsalgorithmus mit $\mathcal{O}(n \log n)$.

Beispiel 1.2. Die Oberfläche eines Würfels wird in gleichgroße Quadrate zerlegt, indem man jede Seite mit den Seitenhalbierenden vierteilt und dieses Prinzip auf die neuen Oberflächenelemente rekursiv anwendet, bis eine bestimmte Tiefe erreicht ist. Dann wendet man das Bisektionsverfahren auf die Menge der Mittelpunkte aller Quadrate an. Die Elemente eines Clusters Ω_l^d benutzen die gleiche Farbe, verschiedene Cluster haben verschiedene Farben (Abb. 1.8).

1.4 Zusammenfassung

Wie eingangs erwähnt erlauben Hierarchische Matrizen die hinreichend genaue Approximation von vollbesetzen Matrizen. Dabei werden der Speicherbedarf und der Aufwand für bestimmte

Operationen von $\mathcal{O}(n^2)$ oder mehr auf fast lineare $\mathcal{O}(n \log^q n)$ gedrückt. Dies ist an einem Beispiel für den Speicherbedarf und die Matrix-Vektor-Multiplikation nachgerechnet worden. Ein weiteres Beispiel ohne Rechnung hat zwar gezeigt, daß Hierarchische Matrizen verschieden konstruiert sein können und die Bauart von dem zugrundeliegenden Problem abhängt, dennoch hat sich eine allgemeine Vorgehensweise angeben lassen, mit der man zu einer Hierarchischen Matrix gelangt. Dabei haben sich die zwei anwendungsbezogenen Fragen

- Wie partioniert man die Indexmenge I *hierarchisch*?
- Wann ist ein Block (r, ε)-zulässig?

gestellt.
Für die erste Frage kann zum Beispiel der zuletzt beschriebene Bisektionsalgorithmus eine Antwort liefern, falls die Indexmenge mit einer Punktmenge assoziiert ist.
Für das zweite Problem gilt, daß man zwar prinzipiell jede beliebige Matrix unabhängig von einer Anwendung auf ihre $\mathcal{H}$-Matrix-Approximierbarkeit untersuchen kann, aber erst die spezielle Anwendung die Frage beantwortet, inwiefern die getroffene Partionierung sinnvoll ist.

Kapitel 2

Singulärwertzerlegung und Niedrigrang–Approximation

Anika Reimann

Ziel dieses Kapitels ist es, eine Niedrig - Rang Matrix B zu einer gegebenen Matrix $A \in \mathbb{R}^{m\times n}$ zu konstruieren, die das Minimierungsproblem

$$\min_{B\in\mathbb{R}^{m\times n}, rgB=k} \|A-B\|_2, \text{ bzw. } \min_{B\in\mathbb{R}^{m\times n}, rgB=k} \|A-B\|_F$$

löst. Außerdem wollen wir den Wert des Fehlers in der jeweiligen Norm berechnen. Mit Hilfe der in Abschnitt 2.2 eingeführten Singulärwertzerlegung kann der Minimierer B mit dem zugehörigen Fehler explizit bestimmt werden.

2.1 Grundlagen

Satz 2.1. *Sei $A \in \mathbb{R}^{m\times n}$. Die Matrix $A^TA \in \mathbb{R}^{n\times n}$ ist symmetrisch und positiv semidefinit. Sie besitzt den Rang $r \leq \min\{m,n\}$.*

Beweis. Aus der Matrizenmultiplikation sehen wir, dass $A^TA \in \mathbb{R}^{n\times n}$. Die Symmetrie für $B = A^TA$ zeigen wir mit Hilfe folgender Umformung

$$B^T = (A^TA)^T = A^T(A^T)^T = A^TA = B.$$

Die positive Semidefinitheit sehen wir sofort, da $\underline{x}^TA^TA\underline{x} = \|A\underline{x}\|_2^2 \geq 0$.
Da $rgA = rgA^T \leq \min\{m,n\}$ ist auch $rgA^TA \leq \min\{m,n\}$. □

2.2 Singulärwertzerlegung

Definition 2.1. Sei $A \in \mathbb{R}^{m\times n}$, so besitzt $A^TA \in \mathbb{R}^{n\times n}$ als symmetrische positiv semidefinite Matrix nichtnegative Eigenwerte $\lambda_1, ..., \lambda_n$. Die nichtnegativen reellen Zahlen $\sigma_i := \sqrt{\lambda_i}$ für $i \in \{1, ..., n\}$ werden als Singulärwerte von A bezeichnet.

Korollar 2.2. *Sei $A \in \mathbb{R}^{n\times n}$ eine symmetrische Matrix mit Eigenwerten λ_i für $i = 1, ..., n$, dann sind die Singulärwerte von A durch $\sigma_i = \sqrt{\lambda_i^2} = |\lambda_i|$ für $i = 1, ..., n$ gegeben.*

Bemerkung 2.1. Bei nichtsymmetrischen Matrizen können große Abweichungen zwischen (komplexen) Eigenwerten und Singulärwerten auftreten.

Als Resultat der Eigenschaften der Matrix A^TA erhalten wir die im folgenden Satz geschilderte Zerlegung einer Matrix A.

Satz 2.3. *Sei $A \in \mathbb{R}^{m\times n}$. Dann gibt es orthogonale Matrizen $U \in \mathbb{R}^{m\times m}$ und $V \in \mathbb{R}^{n\times n}$, so dass $\Sigma := U^TAV$ eine $m \times n$-Matrix der Gestalt*

$$\Sigma = \begin{pmatrix} \Sigma_r & 0 \\ 0 & 0 \end{pmatrix} \tag{2.1}$$

mit $\Sigma_r = diag(\sigma_1, ..., \sigma_r)$ und $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_r > 0$, und Nullblöcken entsprechender Größe ist. Dabei sind $\sigma_1, ..., \sigma_r$ die von Null verschiedenen Singulärwerte von A.

Beweis. Seien $\lambda_1, ..., \lambda_n$ mit $\lambda_1 \geq ... \geq \lambda_n \geq 0$ die Eigenwerte von A^TA. $\sigma_i := \sqrt{\lambda_i}$ für $i = 1, ..., r$ seien die positiven Singulärwerte von A. $V \in \mathbb{R}^{n\times n}$ sei eine orthogonale Matrix, deren Spalten $\underline{v}_1, ..., \underline{v}_n$ ein Orthonormalsystem von Eigenvektoren von A^TA zu den Eigenwerten $\lambda_1, ..., \lambda_n$ bilden. Sei $\Sigma_r := diag(\sigma_1, ..., \sigma_r)$ mit

$$\Sigma := \begin{pmatrix} \Sigma_r & 0 \\ 0 & 0 \end{pmatrix} \in \mathbb{R}^{m\times n}.$$

Dann ist

$$V^TA^TAV = \Sigma^2.$$

Wir definieren nun $V_1 = \left(\underline{v}_1, ..., \underline{v}_r\right)$. Sodann wählen wir $U_1 := AV_1\Sigma_r^{-1}$, dann gilt

$$U_1^TU_1 = \Sigma_r^{-1}V_1^TA^TAV_1\Sigma_r^{-1} = I_r.$$

Die Spalten von U_1 bilden also ein Orthonormalsystem. Wir ergänzen dieses durch $m - r$ weitere Vektoren zu einer Orthonormalbasis in $\mathbb{R}^m$ und fassen die $m - r$ Vektoren zu einer weiteren Matrix U_2 zusammen. Die Matrix $U := (U_1, U_2)$ ist also orthogonal. Es gilt daher $U_1\Sigma_r = AV_1$. Daher erhält man schließlich

$$\begin{aligned} U^TAV &= \begin{pmatrix} U_1^T \\ U_2^T \end{pmatrix} A(V_1, V_2) \\ &= \begin{pmatrix} U_1^TAV_1 & U_1^TAV_2 \\ U_2^TAV_1 & U_2^TAV_2 \end{pmatrix} \\ &= \begin{pmatrix} \Sigma_r^{-1}V_1^TA^TAV_1 & 0 \\ U_2^TU_1\Sigma_r & 0 \end{pmatrix} \\ &= \begin{pmatrix} \Sigma_r & 0 \\ 0 & 0 \end{pmatrix}. \end{aligned}$$

□

Nun können wir die für unsere Approximation benötigte Singulärwertzerlegung definieren.

Definition 2.2. Sei $A \in \mathbb{R}^{m\times n}$ eine Matrix mit $rgA = r$. Eine Zerlegung der Matrix A in der Art, wie in Satz 2.3 geschildert in ein Produkt $A = U\Sigma V^T$ mit orthogonalen Matrizen $U \in \mathbb{R}^{m\times m}$, $V \in \mathbb{R}^{n\times n}$ und einer Matrix

$$\Sigma = \begin{pmatrix} \Sigma_r & 0 \\ 0 & 0 \end{pmatrix} \in \mathbb{R}^{m\times n}, \quad \Sigma_r = diag(\sigma_1, ..., \sigma_r), \quad \sigma_1 \geq ... \geq \sigma_r > 0$$

heißt Singulärwertzerlegung von A.

Korollar 2.4. *Für Matrizen mit der Singulärwertzerlegung $A = U\Sigma V^T$ gilt*

$$A^T A = V\Sigma^T \Sigma V^T.$$

Da A und Σ duch Multiplikation mit nichtsingulären Matrizen auseinander hervorgehen, ist $r = rgA = rg\Sigma$, das heißt der Rang einer Matrix ist gleich der Anzahl der nichtverschwindenden Singulärwerte.
Die Singulärwerte einer Matrix sind eindeutig bestimmt. Seien $\sigma_1 \geq ... \geq \sigma_r > 0$, dann ist der Unterraum, den die Singulärvektoren $\underline{u}_1, ..., \underline{u}_r$ und $\underline{v}_1, ..., \underline{v}_r$ aufspannen, eindeutig bestimmt.

Korollar 2.5. *Wir können die Matrix A auch darstellen durch $A = \sum_{i=1}^{r} \sigma_i \underline{u}_i \underline{v}_i^T$.*

Beweis. Die Aussage ergibt sich sofort aus der Singulärwertzerlegung

$$A = U\Sigma V^T,$$

wegen

$$U\Sigma = (u_{ki}\sigma_i)_{ki}$$

und

$$U\Sigma V^T = \sum_{i=1}^{r} u_{ki}\sigma_i v_{li} = \sum_{i=1}^{r} \underline{u}_i \sigma_i \underline{v}_i^T.$$

□

Korollar 2.6. *Für eine Matrix A mit der Singulärwertzerlegung $U^T AV$ und den Singulärvektoren $\underline{u}_1, ..., \underline{u}_m$ und $\underline{v}_1, ..., \underline{v}_n$ gilt*

$$A\underline{v}_i = \begin{cases} \sigma_i \underline{u}_i & 1 \leq i \leq r, \\ 0 & r+1 \leq i \leq n, \end{cases} \tag{2.2}$$

$$A^T \underline{u}_i = \begin{cases} \sigma_i \underline{v}_i & 1 \leq i \leq r, \\ 0 & r+1 \leq i \leq m. \end{cases} \tag{2.3}$$

2.3 Die Approximation

Bevor wir den Satz von Mirsky mit der Approximation einführen können, müssen wir zunächst noch die Euklidische Matrixnorm und die Frobeniusnorm definieren.

Definition 2.3. Die Euklidische Matrixnorm ist für die Matrix $A \in \mathbb{R}^{m\times n}$ definiert durch

$$\|A\|_2 = \sup_{\underline{x}\in\mathbb{R}^n, \|\underline{x}\|_2 \neq 0} \frac{\|A\underline{x}\|_2}{\|\underline{x}\|_2}.$$

Definition 2.4. Für die Matrix $A \in \mathbb{R}^{m\times n}$ wird die Frobeniusnorm folgendermaßen definiert

$$\|A\|_F = \left(\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2\right)^{\frac{1}{2}}.$$

Es besteht der im folgenden Satz dargestellt Zusammenhang zwischen der Euklidischen Matrixnorm und dem größten Singulärwert.

Satz 2.7. *Für die Euklidische Matrixnorm gilt*

$$\|A\|_2 = \sqrt{\lambda_1} = \sigma_1,$$

wobei λ_1 der größte Eigenwert der Matrix A^TA ist. Sie wird daher auch als Spektralnorm bezeichnet.

Beweis. Es gilt

$$\|A\underline{x}\|_2^2 = \underline{x}^TA^TA\underline{x} = \underline{x}^TV\Sigma^T\Sigma V^T\underline{x} \leq \sigma_1^2\underline{x}^TVV^T\underline{x} = \sigma_1^2\underline{x}^T\underline{x}.$$

Wir erhalten also

$$\frac{\|A\underline{x}\|_2}{\|\underline{x}\|_2} \leq \sigma_1 = \sqrt{\lambda_1}.$$

Sei nun $\underline{v}_1$ der zu λ_1 gehörige Eigenvektor, dann gilt

$$\|A\underline{v}_1\|_2^2 = \underline{v}_1^TA^TA\underline{v}_1 = \underline{v}_1^T\underline{v}_1\lambda_1 = \sigma_1^2.$$

Daher ist

$$\|A\|_2 = \sup_{\underline{x}\in\mathbb{R}^n,\|\underline{x}\|_2\neq 0}\frac{\|A\underline{x}\|_2}{\|\underline{x}\|_2} = \sqrt{\lambda_1} = \sigma_1.$$

□

Satz 2.8. *(Satz von Mirsky) Sei $k < rgA$ und $A_k = \sum_{i=1}^{k}\sigma_i\underline{u}_i\underline{v}_i^T$, mit $\|\underline{v}_i\|_2 = 1$ und $\|\underline{u}_i\|_2 = 1$. Dann gelten die folgenden Beziehungen*

$$\min_{B\in\mathbb{R}^{m\times n},rgB=k}\|A-B\|_2 = \|A-A_k\|_2 = \sigma_{k+1}$$

$$\min_{B\in\mathbb{R}^{m\times n},rgB=k}\|A-B\|_F = \|A-A_k\|_F = \left(\sum_{i=k+1}^{n}\sigma_i^2\right)^{\frac{1}{2}}.$$

Beweis. Zunächst führen wir den Beweis für die Euklidische Matrixnorm.
Da $rgB = k$ ist $\dim\ker B = n-k$. Wir nehmen an, dass

$$\ker B = span\{\underline{x}_{k+1},...,\underline{x}_n\}.$$

Es existiert also ein $0 \neq \underline{z} \in span\{\underline{v}_1,...,\underline{v}_{k+1}\}\cap span\{\underline{x}_{k+1},...,\underline{x}_n\}$, da die $n+1$ Vektoren im $\mathbb{R}^n$ linear abhängig sind. Für dieses $\underline{z}$ gilt

$$\|A\underline{z}\|_2 = \|(A-B)\underline{z}\|_2. \tag{2.4}$$

Ohne Beschränkung der Allgemeinheit sei nun $\|\underline{z}\|_2 = 1$. Dann gilt mit $\underline{z} = \sum_{i=1}^{k+1} z_i \underline{v}_i$ wegen der Zerlegung $A^T A = V \Sigma^T \Sigma V^T$ und weil σ_{k+1} der kleinste Singulärwert ist

$$\begin{aligned}\|A - B\|_2^2 \geq \|(A-B)\underline{z}\|_2^2 &\overset{(2.4)}{=} \|A\underline{z}\|_2^2 = \sum_{i=1}^{k+1} \underline{z}^T \underline{v}_i \sigma_i^2 \underline{v}_i^T \underline{z} \\ &= \sum_{i=1}^{k+1}\sum_{l=1}^{k+1} z_l \underline{v}_l^T \underline{v}_i \sigma_i^2 z_l \underline{v}_i^T \underline{v}_l = \sum_{i=1}^{k+1} z_i^2 \sigma_i^2 \geq \sum_{i=1}^{k+1} z_i^2 \sigma_{k+1}^2 \\ &= \sigma_{k+1}^2 \sum_{i=1}^{k+1} z_i^2 = \sigma_{k+1}^2.\end{aligned} \tag{2.5}$$

Desweiteren gilt mit $\underline{w} \in \mathbb{R}^n$ beliebig, $\underline{w} = \sum_{l=1}^{n} w_l \underline{v}_l$

$$\begin{aligned}\|(A - A_k)\underline{w}\|_2^2 &= \left(\sum_{i=k+1}^{r} \sigma_i \underline{u}_i \underline{v}_i^T \underline{w}, \sum_{j=k+1}^{r} \sigma_j \underline{u}_j \underline{v}_j^T \underline{w} \right) \\ &= \left(\sum_{i=k+1}^{r} \sigma_i \underline{u}_i \underline{v}_i^T \left(\sum_{l=1}^{n} w_l \underline{v}_l \right), \sum_{j=k+1}^{r} \sigma_j \underline{u}_j \underline{v}_j^T \left(\sum_{o=1}^{n} w_o \underline{v}_o \right) \right) \\ &= \left(\sum_{i=k+1}^{r} \sigma_i \underline{u}_i w_i, \sum_{j=k+1}^{r} \sigma_j \underline{u}_j w_j \right) \\ &= \sum_{i=k+1}^{r} w_i^2 \sigma_i^2 \leq \sigma_{k+1}^2 \|\underline{w}\|_2^2.\end{aligned}$$

Wir erhalten also

$$\|A - A_k\|_2^2 = \sup_{\underline{w} \in \mathbb{R}^n, \|\underline{w}\| \neq 0} \frac{\|(A - A_k)\underline{w}\|_2^2}{\|\underline{w}\|_2^2} = \sigma_{k+1}^2.$$

Da

$$\|A - B\|_2^2 \geq \sigma_{k+1}^2 \text{ und } \|A - A_k\|_2^2 = \sigma_{k+1}^2$$

ist

$$\min_{B \in \mathbb{R}^{m \times n}, rgB = k} \|A - B\|_2 = \|A - A_k\|_2 = \sigma_{k+1}.$$

Wir wollen nun den Beweis für die Frobeniusnorm führen.
Zunächst zeigen wir wie oben

$$\|A - B\|_F^2 \geq \sum_{i=k+1}^{n} \sigma_i^2.$$

Wir nehmen wieder an, dass

$$\ker B = span\{\underline{x}_{k+1}, ..., \underline{x}_n\}.$$

Es existieren also $0 \neq \underline{z}_i \in span\{\underline{v}_1, ..., \underline{v}_k, \underline{v}_{k+i}\} \cap span\{\underline{x}_{k+1}, ..., \underline{x}_n\}$ für $i = k+1, ..., n$, da die $n+k$ Vektoren im $\mathbb{R}^n$ linear abhängig sind. Diese $\underline{z}_i$ Vektoren orthonormieren wir und ergänzen sie um $n-k$ Vektoren zu einer $n \times n$ – Matrix Z. Da

$$\begin{aligned}\|(A-B)Z\|_F^2 &= \sum_{j=1}^{m}\sum_{i=1}^{n}\left(\sum_{l=1}^{n}(a_{jl}-b_{jl})z_{li}\right)^2 \\ &= \sum_{j=1}^{m}\sum_{i=1}^{n}\sum_{l=1}^{n}(a_{jl}-b_{jl})^2 z_{li}^2 \\ &= \sum_{j=1}^{m}\sum_{l=1}^{n}(a_{jl}-b_{jl})^2 \sum_{i=1}^{n} z_{li}^2 \\ &= \|A-B\|_F^2\end{aligned}$$

und

$$\|(A-B)Z\|_F^2 = \sum_{j=1}^{m}\sum_{i=1}^{n}\left(\sum_{l=1}^{n}(a_{jl}-b_{jl})z_{li}\right)^2 = \sum_{i=1}^{n}\|(A-B)\underline{z}_i\|_2^2$$

gilt

$$\|A-B\|_F^2 = \|(A-B)Z\|_F^2 = \sum_{i=1}^{n}\|(A-B)\underline{z}_i\|_2^2 \geq \sum_{i=k+1}^{n}\|(A-B)\underline{z}_i\|_2^2 \geq \sum_{i=k+1}^{n}\sigma_i^2.$$

Wir erhalten unter Ausnutzung von $\|\underline{u}_i\|_2 = 1$ und $\|\underline{v}_i\|_2 = 1$

$$\begin{aligned}\|A-A_k\|_F^2 &= \left\|\sum_{i=k+1}^{n}\sigma_i \underline{u}_i \underline{v}_i^T\right\|^2 = \sum_{l=1}^{m}\sum_{j=1}^{n}\left|\sum_{i=k+1}^{n}\sigma_i u_{li} v_{ji}\right|^2 \\ &= \sum_{i=k+1}^{n}\sum_{l=1}^{m}\sum_{j=1}^{n}(\sigma_i u_{li} v_{ji})^2 = \sum_{i=k+1}^{n}\sigma_i^2\left(\sum_{l=1}^{m}\sum_{j=1}^{n}u_{li}^2 v_{ji}^2\right) \\ &= \sum_{i=k+1}^{n}\sigma_i^2.\end{aligned}$$

Da

$$\|A-B\|_F^2 \geq \sum_{i=k+1}^{n}\sigma_i^2 \text{ und } \|A-A_k\|_F^2 = \sum_{i=k+1}^{n}\sigma_i^2$$

ist

$$\min_{B\in\mathbb{R}^{m\times n}, rgB=k}\|A-B\|_F = \|A-A_k\|_F = \left(\sum_{i=k+1}^{n}\sigma_i^2\right)^{\frac{1}{2}}.$$

□

Kapitel 3

Arithmetik von $\mathcal{H}$–Matrizen

Kersten Hof

3.1 Einleitung

Für diesen gesamten Abschnitt sei auf [7] verwiesen. Thema dieses Kapitels sollen verschiedene arithmetische Operationen von $\mathcal{H}$–Matrizen und ihre Aufwandsabschätzungen sein, wobei hier nur die in der Praxis teuren Multiplikationen gezählt werden. Die Arithmetik wird genauer untersucht im Spezialfall der unten beschriebenen $\mathcal{H}$–Partitionierung T einer Indexmenge $\mathcal{I} = \{1, \ldots, n\}$ für $n = 2^p$. Durch die spezielle Form der Matrizen erhält man für alle interessanten Matrixoperationen annähernd linearen Aufwand. Zunächst werden die hierarchische Partitionierung einer Indexmenge $\mathcal{I} = \{1, \ldots, n\}$ und die hierarchische Block–Partitionierung sowie daraus dann eine Menge von $\mathcal{H}$–Matrizen eingeführt, um die Form dieser Matrizen vorzustellen.

3.1.1 Hierarchische Partitionierung einer Indexmenge

Definition 3.1. Als hierarchische Partitionierung einer Indexmenge $\mathcal{I} = \{1, \ldots, n\}$ wird die Menge

$$\{\mathcal{I}_i^\lambda\}_{\lambda=0,\ldots,L, i=1,\ldots,N(\lambda)} \tag{3.1}$$

bezeichnet, wobei die $\mathcal{I}_i^\lambda$ für $i = 1, \ldots, N(\lambda)$ disjunkte Teilmengen der Indexmenge $\mathcal{I}$ auf dem Level λ sind und für jedes Level λ gilt:

$$\bigcup_{i=1,\ldots,N(\lambda)} \mathcal{I}_i^\lambda = \mathcal{I}. \tag{3.2}$$

Dies führt dann auf eine Baumstruktur T, bei der ein Knoten die disjunkte Vereinigung seiner Sohnknoten ist.
Ein Beispiel stellt folgende $\mathcal{H}$–Partitionierung von $\mathcal{I} = \{1, \ldots, n\}$ im Spezialfall $n = 2^p$ dar: Die feinste Partitionierung auf dem Level $L = p$ besteht aus den einelementigen Teilmengen

$$\mathcal{I}_1^p = \{1\}, \mathcal{I}_2^p = \{2\}, \ldots, \mathcal{I}_n^p = \{n\}. \tag{3.3}$$

Auf Level $p - 1$ werden jeweils 2 Teilmengen von Level p kombiniert zu

$$\mathcal{I}_1^{p-1} = \{1, 2\}, \mathcal{I}_2^{p-1} = \{3, 4\}, \ldots, \mathcal{I}_{\frac{n}{2}}^{p-1} = \{n-1, n\}. \tag{3.4}$$

Es folgen 4–elementige Teilmengen auf Level $p-2$ und entsprechend für kleinere Level. Auf dem gröbsten Level 0 ist schließlich die ganze Indexmenge $\mathcal{I}_1^0 = \mathcal{I} = \{1, \ldots, n\}$ der einzige Block. Auf diese Art und Weise erhält man einen Binärbaum T mit Wurzel $\mathcal{I}$ und Knoten $\left\{\mathcal{I}_i^\lambda : 0 \leq \lambda \leq p, 1 \leq i \leq 2^\lambda\right\}$.

3.1.2 Hierarchische Block–Partitionierung

Definition 3.2. Die durch den Baum T von Definition 3.1 induzierte hierarchische $\mathcal{H}$–Partitionierung $P_2(\mathcal{I}, T)$ von $\mathcal{I} \times \mathcal{I}$ ist definiert als

$$P_2(\mathcal{I}, T) = \{(\mathcal{I}_1, \mathcal{I}_2) \in \mathcal{I}_i^\lambda \times \mathcal{I}_j^\lambda, \lambda = 1, \ldots, L, i, j = 1, \ldots, n(\lambda)\}. \tag{3.5}$$

Im oben bereits vorgestellten Spezialfall kann man eine hierarchische Block–Partitionierung $P_2 = P_2(\mathcal{I}, T)$ von $\mathcal{I} \times \mathcal{I}$ rekursiv über die Tiefe des Baumes T definieren:

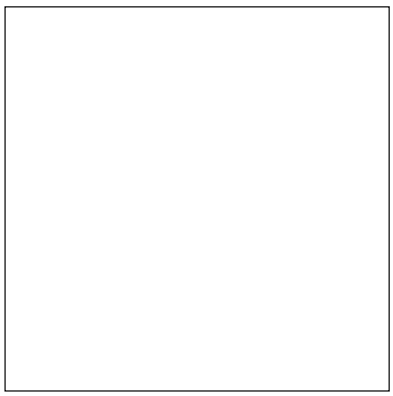
1. $p = 0$:
$$P_2(\mathcal{I}, T) := \{\mathcal{I} \times \mathcal{I}\}. \tag{3.6}$$

2. $p = 1$:
$$P_2(\mathcal{I}, T) := \{\mathcal{I}_1 \times \mathcal{I}_1, \mathcal{I}_1 \times \mathcal{I}_2, \mathcal{I}_2 \times \mathcal{I}_1, \mathcal{I}_2 \times \mathcal{I}_2\}, \tag{3.7}$$
wobei $\mathcal{I}_1, \mathcal{I}_2$ die beiden Söhne von $\mathcal{I} \in T$ sind.

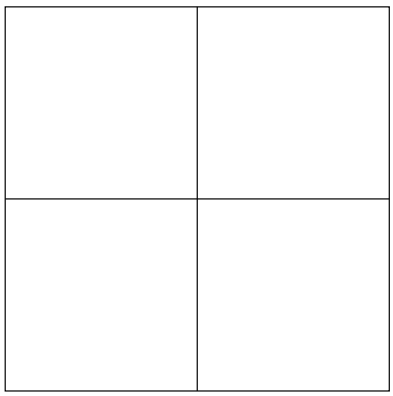
3. $p > 1$:
$$P_2(\mathcal{I}, T) := P_2(\mathcal{I}_1, T_1) \cup \{\mathcal{I}_1 \times \mathcal{I}_2\} \cup \{\mathcal{I}_2 \times \mathcal{I}_1\} \cup P_2(\mathcal{I}_2, T_2), \tag{3.8}$$
wobei mit T_k die $\mathcal{H}$–Partitionierung der Teilmenge $\mathcal{I}_k$ bezeichnet wird.

Zur graphischen Veranschaulichung können folgende Bilder dienen:

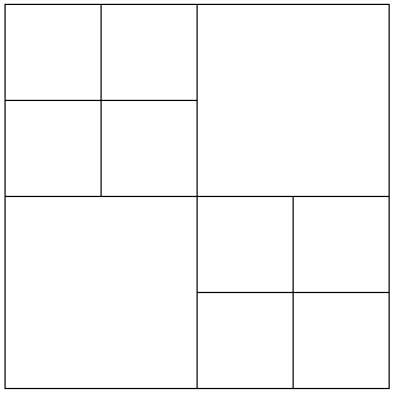
p = 0:



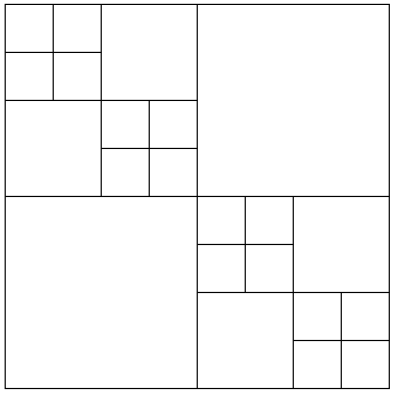
p = 1:



p = 2:



p = 3:



3.1.3 $\mathcal{H}$–Matrizen

Sei P_2 die oben definierte allgemeine $\mathcal{H}$-Block–Partitionierung von $\mathcal{I}\times\mathcal{I}$ und $k \in \mathbb{N}$. Mit k wird der Rang jedes Matrixblockes beschränkt. Die Menge der durch P_2 induzierten $\mathcal{H}$–Matrizen wird definiert durch

$$\mathcal{M}_{\mathcal{H},k}(\mathcal{I}\times\mathcal{I}, P_2) := \left\{ M \in \mathbb{R}^{\mathcal{I}\times\mathcal{I}} : \text{ jeder Block } M^b, b \in P_2, \text{ erfüllt } \operatorname{rang}(M^b) \leq k \right\}.$$

Eine Matrix A wird ab jetzt als Rang–k–Matrix bezeichnet, falls $\operatorname{rang}(A) \leq k$ gilt.
Nach obiger Block–Partitionierung $P_2(\mathcal{I}, T)$ hat eine $n \times n$ $\mathcal{H}$–Matrix A die Block–Struktur

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}, \tag{3.9}$$

wobei A_{11} und A_{22} rekursiv definierte $\frac{n}{2} \times \frac{n}{2}$–$\mathcal{H}$–Matrizen kleinerer Dimension mit gleicher Struktur und A_{12}, A_{21} Rang–k–Matrizen sind.

3.2 Rang–1–Matrizen

Zur Vereinfachung werden zunächst Rang–1–Matrizen betrachtet, wobei sich herausstellen wird, dass sich die wesentlichen Eigenschaften auf Rang–k–Matrizen verallgemeinern lassen.

3.2.1 Multiplikation von Rang–1–Matrizen

Jede $n \times m$ Rang–1–Matrix A kann in der Form $A = \underline{a}\,\underline{b}^T, \underline{a} \in \mathbb{R}^n, \underline{b} \in \mathbb{R}^m$ geschrieben werden. Daraus ergibt sich sofort das folgende Lemma:

Lemma 3.1. *1. Der Speicheraufwand für eine Rang–1–Matrix $A \in \mathbb{R}^{n\times m}$ beträgt $n + m$.*

2. Für die Matrix–Vektor–Multiplikation $A\underline{c} = \underline{a}\,\underline{b}^T\,\underline{c} = (\underline{b}^T\,\underline{c})\,\underline{a}$ mit $\underline{c} \in \mathbb{R}^m$ werden $m + n$ Multiplikationen benötigt.

3. Bei der Multiplikation $BA = B\underline{a}\,\underline{b}^T = (B\underline{a})\underline{b}^T = \underline{\tilde{a}}\,\underline{b}^T$, $\underline{\tilde{a}} = B\underline{a}$, mit einer allgemeinen Matrix B benötigt man nur den Arbeitsaufwand, um die Matrix–Vektor–Multiplikation $B\underline{a}$ zu berechnen.

4. *Für die Matrix–Matrix–Multiplikation* $AB = \underline{a}_1\,\underline{b}_1^T\,\underline{a}_2\,\underline{b}_2^T = (\underline{b}_1^T\,\underline{a}_2)\underline{a}_1\,\underline{b}_2^T$ *zweier Rang–1–Matrizen* $A \in \mathbb{R}^{n\times m}$ *und* $B \in \mathbb{R}^{m\times l}$ *muss nur ein Skalarprodukt sowie die Skalierung eines Vektors mit einer reellen Zahl berechnet werden, so dass sich als Aufwand* $m + \min\{n, l\}$ *Multiplikationen ergeben. Außerdem ergibt sich als Ergebnis wiederum eine Rang–1–Matrix.*

5. *Die Berechnung eines jeden Eintrages* $A_{ij} = a_i b_j$ *erfordert genau eine Operation.*

6. *Die Berechnung einer kompletten Zeile von A erfordert m Operationen. Entsprechendes gilt für eine Spalte mit n Operationen.*

3.2.2 Summe von Rang–1–Matrizen und Singulärwertzerlegung

Durch die exakte Addition zweier Rang–1–Matrizen entsteht im allgemeinen eine Matrix vom Rang 2. Mit Hilfe der im Kapitel 2 behandelten Singulärwertzerlegung kann man aus zwei Rang–1–Matrizen eine approximative Rang–1–Matrix für die Summe bestimmen. Als Schreibweise für die näherungsweise Addition wird

$$\tilde{C} = A \oplus_{R1} B$$

gewählt. Nach der Theorie der Singulärwertzerlegung lässt sich eine $n \times m$ Matrix A in

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

mit unitärer $n \times n$ Matrix U, unitärer $m \times m$ Matrix V und einer $n \times m$ Diagonalmatrix Σ der Gestalt (2.1) mit den Einträgen $d_1, \ldots, d_r$ für $r = \operatorname{rang}(A)$ zerlegen, wobei $d_1 \geq d_2 \geq \ldots d_r > 0$ gilt. Die einfachste approximative $n \times m$ Matrix vom Rang $\hat{r} \in [1, r]$ ist

$$\hat{A} = U\hat{\Sigma}V^T,$$

wobei in $\hat{\Sigma}$ die ersten $\hat{r}$ Diagonaleinträge von Σ übernommen und die restlichen Einträge zu Null gesetzt werden.
Zur Untersuchung der approximativen Summe seien nun die Rang–1–Matrizen $A = \underline{a}_1\,\underline{b}_1^T$ und $B = \underline{a}_2\,\underline{b}_2^T$ gegeben, und $C := A + B$ sei die echte Summe mit der Singulärwertzerlegung $C = U\Sigma V^T$. Die Spalten der unitären Matrix V sind dabei die normierten Eigenvektoren von C^TC. Wegen

$$\begin{aligned} C^TC\underline{v} &= (\underline{a}_1\,\underline{b}_1^T + \underline{a}_2\,\underline{b}_2^T)^T(\underline{a}_1\,\underline{b}_1^T + \underline{a}_2\,\underline{b}_2^T)\underline{v} \\ &= (\underline{b}_1\,\underline{a}_1^T + \underline{b}_2\,\underline{a}_2^T)(\underline{a}_1\,\underline{b}_1^T + \underline{a}_2\,\underline{b}_2^T)\underline{v} \\ &= \underline{b}_1\,\underline{a}_1^T\,\underline{a}_1\,\underline{b}_1^T\,\underline{v} + \underline{b}_1\,\underline{a}_1^T\,\underline{a}_2\,\underline{b}_2^T\,\underline{v} + \underline{b}_2\,\underline{a}_2^T\,\underline{a}_1\,\underline{b}_1^T\,\underline{v} + \underline{b}_2\,\underline{a}_2^T\,\underline{a}_2\,\underline{b}_2^T\,\underline{v} \\ &= \alpha\underline{b}_1 + \beta\underline{b}_2 = \lambda\underline{v} \end{aligned}$$

gilt für einen Eigenvektor $\underline{v}$ von C^TC, dass er im Aufspann der Vektoren $\underline{b}_1, \underline{b}_2$ enthalten ist. Der Ansatz $\underline{v} = \alpha_1\underline{b}_1 + \alpha_2\underline{b}_2$ führt dann auf das Eigenwertproblem

$$\begin{aligned} \lambda\underline{v} &= C^TC\underline{v} = (A+B)^T(A+B)\underline{v} \\ &= (\underline{a}_1\,\underline{b}_1^T + \underline{a}_2\,\underline{b}_2^T)^T(\underline{a}_1\,\underline{b}_1^T + \underline{a}_2\,\underline{b}_2^T)(\alpha_1\underline{b}_1 + \alpha_2\underline{b}_2) \\ &= (\underline{b}_1\,\underline{a}_1^T + \underline{b}_2\,\underline{a}_2^T)(\underline{a}_1\,\underline{b}_1^T + \underline{a}_2\,\underline{b}_2^T)(\alpha_1\underline{b}_1 + \alpha_2\underline{b}_2) \\ &= (\underline{b}_1\,\underline{a}_1^T\,\underline{a}_1\,\underline{b}_1^T + \underline{b}_1\,\underline{a}_1^T\,\underline{a}_2\,\underline{b}_2^T + \underline{b}_2\,\underline{a}_2^T\,\underline{a}_1\,\underline{b}_1^T + \underline{b}_2\,\underline{a}_2^T\,\underline{a}_2\,\underline{b}_2^T)(\alpha_1\underline{b}_1 + \alpha_2\underline{b}_2) \\ &= \underline{b}_1(\alpha_1\underline{a}_1^T\,\underline{a}_1\,\underline{b}_1^T\,\underline{b}_1 + \alpha_2\underline{a}_1^T\,\underline{a}_1\,\underline{b}_1^T\,\underline{b}_2 + \alpha_1\underline{a}_1^T\,\underline{a}_2\,\underline{b}_2^T\,\underline{b}_1 + \alpha_2\underline{a}_1^T\,\underline{a}_2\,\underline{b}_2^T\,\underline{b}_2) \\ &+ \underline{b}_2(\alpha_1\underline{a}_2^T\,\underline{a}_1\,\underline{b}_1^T\,\underline{b}_1 + \alpha_2\underline{a}_2^T\,\underline{a}_1\,\underline{b}_1^T\,\underline{b}_2 + \alpha_1\underline{a}_2^T\,\underline{a}_2\,\underline{b}_2^T\,\underline{b}_1 + \alpha_2\underline{a}_2^T\,\underline{a}_2\,\underline{b}_2^T\,\underline{b}_2). \end{aligned}$$

Aus $\lambda \underline{v} = \alpha_1 \lambda \underline{b}_1 + \alpha_2 \lambda \underline{b}_2$ folgen durch Koeffizientenvergleich die Gleichungen

$$\lambda\alpha_1 = \alpha_1 \underline{a}_1^T \underline{a}_1 \underline{b}_1^T \underline{b}_1 + \alpha_2 \underline{a}_1^T \underline{a}_1 \underline{b}_1^T \underline{b}_2 + \alpha_1 \underline{a}_1^T \underline{a}_2 \underline{b}_2^T \underline{b}_1 + \alpha_2 \underline{a}_1^T \underline{a}_2 \underline{b}_2^T \underline{b}_2$$

und

$$\lambda\alpha_2 = \alpha_1 \underline{a}_2^T \underline{a}_1 \underline{b}_1^T \underline{b}_1 + \alpha_2 \underline{a}_2^T \underline{a}_1 \underline{b}_1^T \underline{b}_2 + \alpha_1 \underline{a}_2^T \underline{a}_2 \underline{b}_2^T \underline{b}_1 + \alpha_2 \underline{a}_2^T \underline{a}_2 \underline{b}_2^T \underline{b}_2,$$

was äquivalent zum 2×2–System

$$\lambda \begin{pmatrix} \alpha_1 \\ \alpha_2 \end{pmatrix} = \begin{pmatrix} \underline{a}_1^T \underline{a}_1 & \underline{a}_1^T \underline{a}_2 \\ \underline{a}_2^T \underline{a}_1 & \underline{a}_2^T \underline{a}_2 \end{pmatrix} \begin{pmatrix} \underline{b}_1^T \underline{b}_1 & \underline{b}_1^T \underline{b}_2 \\ \underline{b}_2^T \underline{b}_1 & \underline{b}_2^T \underline{b}_2 \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \end{pmatrix} \tag{3.10}$$

ist.
Aus dem System (3.10) werden die beiden Eigenwerte λ_1, λ_2 berechnet und der zum größeren der beiden gehörige Eigenvektor $\underline{\alpha} = \begin{pmatrix} \alpha_1 \\ \alpha_2 \end{pmatrix}$ bestimmt. Daraus kann dann der zum ursprünglichen System gehörige Eigenvektor $\underline{v} = \alpha_1 \underline{b}_1 + \alpha_2 \underline{b}_2$ von $C^T C$ ermittelt werden. Die approximative Matrix $\tilde{C} = A \oplus_{R1} B$ kann nun durch $\tilde{C} = \underline{a}_3 \underline{b}_3^T$ mit $\underline{b}_3 := \underline{v}$ und $\underline{a}_3 := C\underline{v} = A\underline{v} + B\underline{v}$ dargestellt werden.

Satz 3.2. *Die Rang–1–Addition $\oplus_{\mathrm{R1}}$ zweier $n \times m$ Matrizen kostet $5n + 6m + \mathcal{O}(1)$ Operationen.*

Beweis. Seien $A = \underline{a}_1 \underline{b}_1^T, B = \underline{a}_2 \underline{b}_2^T$ gegeben und $\tilde{C} = A \oplus_{R1} B = \underline{a}_3 \underline{b}_3^T$ sei die approximative Matrix von $C = A + B$. Die Berechnung der beiden Gram–Matrizen $G_a = (\underline{a}_i^T \underline{a}_j)_{i,j=1,2}$ und $G_b = (\underline{b}_i^T \underline{b}_j)_{i,j=1,2}$ beinhaltet 6 Skalarprodukte und somit $3n + 3m = 3(n + m)$ Multiplikationen. Sei ferner $\underline{\alpha}$ der Eigenvektor von $G_a G_b$ zum größeren Eigenwert, der durch das Lösen des 2×2–Eigenwertproblems mit Aufwand $\mathcal{O}(1)$ erhalten wird. Um den normierten Vektor $\underline{b}_3 := \alpha_1 \underline{b}_1 + \alpha_2 \underline{b}_2$ zu bestimmen, werden $3m + \mathcal{O}(1)$ Operationen benötigt. Um dies einzusehen, schreibt man ohne Beschränkung der Allgemeinheit $\underline{b}_3 := \alpha_1 \underline{b}_1 + \alpha_2 \underline{b}_2 = \alpha_1 (\underline{b}_1 + \frac{\alpha_2}{\alpha_1} \underline{b}_2)$. Man berechnet nun $\tilde{\underline{b}} = \underline{b}_1 + \frac{\alpha_2}{\alpha_1} \underline{b}_2$, die Norm $|\tilde{\underline{b}}|$ und schließlich $\underline{b}_3 = \frac{\tilde{\underline{b}}}{|\tilde{\underline{b}}|}$ jeweils mit m Multiplikationen. Bei der Berechnung von $\underline{a}_3$ wird dann ausgenutzt, dass $\underline{b}_i^T \underline{b}_j$ für $i, j = 1, 2$ bereits aus der Bestimmung der Matrix G_b bekannt sind. Somit ergibt sich

$$\begin{aligned} \underline{a}_3 &= (A + B)\underline{b}_3 = (\underline{a}_1 \underline{b}_1^T + \underline{a}_2 \underline{b}_2^T)(\alpha_1 \underline{b}_1 + \alpha_2 \underline{b}_2) \\ &= \underline{a}_1 \alpha_1 \underline{b}_1^T \underline{b}_1 + \underline{a}_1 \alpha_2 \underline{b}_1^T \underline{b}_2 + \underline{a}_2 \alpha_1 \underline{b}_2^T \underline{b}_1 + \underline{a}_2 \alpha_2 \underline{b}_2^T \underline{b}_2 \\ &= (\alpha_1 \underline{b}_1^T \underline{b}_1 + \alpha_2 \underline{b}_1^T \underline{b}_2)\underline{a}_1 + (\alpha_1 \underline{b}_2^T \underline{b}_1 + \alpha_2 \underline{b}_2^T \underline{b}_2)\underline{a}_2. \end{aligned}$$

Den Vektor $\underline{a}_3$ muss man hierbei nicht mehr normieren, da der Eigenwert bereits enthalten ist. Es kommen also noch $2n + \mathcal{O}(1)$ weitere Multiplikationen dazu. Nach Addition ergibt sich $3(n + m) + \mathcal{O}(1) + 3m + \mathcal{O}(1) + 2n + \mathcal{O}(1) = 5n + 6m + \mathcal{O}(1)$ als Gesamtaufwand. □

Als nächstes kann man die Überlegungen bezüglich der Rang–1–Matrizen auf Rang–k–Matrizen verallgemeinern. Eine solche Rang–k–Matrix A lässt sich schreiben als

$$A = \sum_{i=1}^{k} \underline{a}_i \underline{b}_i^T$$

und hat folgende Eigenschaften:

1. Der Speicheraufwand für $n \times m$ Matrizen vom Rang k beträgt $k \cdot (n+m)$.

2. Für $A = \sum_{i=1}^{k} \underline{a}_i \underline{b}_i^T$ und $B = \sum_{j=1}^{k} \underline{c}_j \underline{d}_j^T$ mit $A \in \mathbb{R}^{n\times m}, B \in \mathbb{R}^{m\times l}$ erfordert das Produkt
$$AB = \sum_{i=1}^{k} \underline{a}_i \underline{b}_i^T \sum_{j=1}^{k} \underline{c}_j \underline{d}_j^T = \sum_{i,j=1}^{k} \underline{a}_i \underline{b}_i^T \underline{c}_j \underline{d}_j^T = \sum_{i,j=1}^{k} (\underline{b}_i^T \underline{c}_j)\underline{a}_i \underline{d}_j^T$$
k^2 Skalarprodukte und Skalierungen, also einen Aufwand von $k^2 \cdot (m + \min\{n, l\})$ Multiplikationen.

3. Bei der approximierenden Addition zweier Rang–k–Matrizen, bei der dann wiederum eine Rang–k–Matrix erhalten wird, benötigt man die Lösung eines $(2k \times 2k)$–Eigenwertproblems, so dass man die Kosten $\mathcal{O}(k^2 \cdot (n+m) + k^3)$ erhält. In der Praxis wird dies häufig mit Hilfe einer QR–Zerlegung der beiden Matrizen realisiert. Dazu sei auf Kapitel 4 verwiesen.

Da durch Rang–k–Matrizen im Gegensatz zu Rang–1–Matrizen keine zusätzlichen Probleme auftreten und die Komplexitätsabschätzungen von der gleichen Ordnung sind, wird im Folgenden zur Vereinfachung immer von Rang–1–Matrizen ausgegangen.

3.3 Komplexitätsabschätzungen für $\mathcal{H}$–Matrizen

Nachdem der Aufbau der $\mathcal{H}$–Matrizen im betrachteten Beispiel aus Definition 3.2 vorgestellt wurde, werden im Folgenden Komplexitätsbetrachtungen für die wichtigsten Matrixoperationen für diese speziellen $\mathcal{H}$–Matrizen vorgenommen. Hierbei werden einige Vorüberlegungen zu Aufwandsabschätzungen von Rang–1–Matrizen aus Lemma 3.1 verwendet.

3.3.1 Speicherplatz für $\mathcal{H}$–Matrizen

Zunächst wird die Anzahl der Blöcke einer $n \times n$ $\mathcal{H}$–Matrix in der Partitionierung $P_2(\mathcal{I}, T)$ mit $n = 2^p$ bestimmt, die mit $N_{\text{block}}(p)$ bezeichnet wird. Nach Definition 3.2 gelten $N_{\text{block}}(0) = 1$ und $N_{\text{block}}(1) = 4$. Die Blockaufteilung aus (3.9) führt auf $N_{\text{block}}(p) = 2 + 2N_{\text{block}}(p-1)$ für $p > 1$. Daraus folgt durch Induktion

$$N_{\text{block}}(p) = 3 \cdot 2^p - 2. \tag{3.11}$$

Mit $N_{R1}(p)$ wird im Folgenden der Speicherplatz einer Rang–1–Matrix der Größe $n \times n$ mit $n = 2^p$ bezeichnet. Nach Lemma 3.1 wird für die Rang–1–Matrix $A = \underline{a}\,\underline{b}^T$ der Speicherplatz $N_{R1}(p) := n + n = 2^{p+1}$ benötigt.
Sei nun $N_{\text{storage}}(p)$ der Speicherplatz, der für eine $\mathcal{H}$–Matrix der Dimension $2^p \times 2^p$ benötigt wird. Dann gilt aufgrund der Blockaufteilung aus (3.9)

$$N_{\text{storage}}(p) = 2N_{R1}(p-1) + 2N_{\text{storage}}(p-1) = 2^{p+1} + 2N_{\text{storage}}(p-1). \tag{3.12}$$

Zusammen mit $N_{\text{storage}}(0) = 1$ können wir folgendes Lemma folgern:

Lemma 3.3. *Der benötigte Speicherplatz für eine $n \times n$–$\mathcal{H}$–Matrix mit $n = 2^p$ beträgt $N_{storage}(p) = (2p+1) \cdot 2^p = (1 + 2\log_2 n) \cdot n$.*

Beweis. Der Beweis erfolgt durch Induktion über p. Der Induktionsanfang $N_{\text{storage}}(0) = 1$ ergibt sich sofort als richtig. Um den Induktionsschritt durchzuführen, wird die für p bereits richtige Aussage $N_{\text{storage}}(p) = (2p+1) \cdot 2^p$ und die Rekursionsvorschrift (3.12) verwendet, so dass

$$\begin{aligned} N_{\text{storage}}(p+1) &= 2^{p+2} + 2N_{\text{storage}}(p) = 2 \cdot 2^{p+1} + 2 \cdot (2p+1) \cdot 2^p \\ &= 2 \cdot 2^{p+1} + 2^{p+1} \cdot (2p+1) = 2^{p+1}(2 \cdot (p+1) + 1), \end{aligned}$$

also die Behauptung für $p+1$, gilt. □

3.3.2 Addition für $\mathcal{H}$–Matrizen

Die exakte Addition zweier $\mathcal{H}$–Matrizen kommt durch die exakte Addition aller Blöcke zustande. Man definiert nun die näherungsweise Addition zweier $\mathcal{H}$–Matrizen dadurch, dass die exakte Addition der Blöcke durch die Addition $\oplus_{R1}$ ersetzt wird und wählt als Schreibweise wiederum

$$\tilde{C} = A \oplus_{R1} B.$$

Mit $N_{\mathcal{H}+\mathcal{H}}(p)$ seien nun die Kosten der Addition zweier $2^p \times 2^p$ $\mathcal{H}$–Matrizen bezeichnet. Dann gilt wegen der Blockstruktur (3.9) und wegen der Aufwandsabschätzung aus Satz 3.2

$$\begin{aligned} N_{\mathcal{H}+\mathcal{H}}(p) &= 2N_{\mathcal{H}+\mathcal{H}}(p-1) + 2N_{R1+R1}(p-1) \\ &= 2N_{\mathcal{H}+\mathcal{H}}(p-1) + 2 \cdot (5\frac{n}{2} + 6\frac{n}{2} + \mathcal{O}(1)) \\ &= 2N_{\mathcal{H}+\mathcal{H}}(p-1) + 11n + \mathcal{O}(1). \end{aligned}$$

Analog gilt für die Summe einer $n \times n$ $\mathcal{H}$–Matrizen A mit einer Rang–1–Matrix B aufgrund der Form $B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$

$$N_{\mathcal{H}+R1}(p) = 2N_{\mathcal{H}+R1}(p-1) + 2N_{R1+R1}(p-1), \tag{3.13}$$

so dass durch Induktion folgendes Lemma folgt:

Lemma 3.4. *Die Rang–1–Addition zweier $n \times n$ $\mathcal{H}$–Matrizen oder einer $\mathcal{H}$–Matrix und einer Rang–1–Matrix benötigt $11 \cdot 2^p \cdot p + \mathcal{O}(2^p) = 11n \cdot \log_2 n + \mathcal{O}(n)$ Operationen.*

Beweis. Der Beweis erfolgt wiederum durch vollständige Induktion über p. Der Induktionsanfang $N_{\mathcal{H}+\mathcal{H}}(0) = \mathcal{O}(1)$ erweist sich sofort als richtig. Im Induktionsschritt folgt aus der Rekursionsvorschrift (3.13) und der Induktionsvoraussetzung $N_{\mathcal{H}+\mathcal{H}}(p) = 11p \cdot 2^p + \mathcal{O}(2^p)$

$$\begin{aligned} N_{\mathcal{H}+\mathcal{H}}(p+1) &= 2N_{\mathcal{H}+\mathcal{H}}(p) + 11 \cdot 2^{p+1} + \mathcal{O}(1) \\ &= 2 \cdot (11p \cdot 2^p + \mathcal{O}(2^p)) + 11 \cdot 2^{p+1} + \mathcal{O}(1) \\ &= 11 \cdot 2^{p+1} \cdot (p+1) + \mathcal{O}(2^{p+1}). \end{aligned}$$

□

3.3.3 Matrix–Vektor–Multiplikation bei $\mathcal{H}$–Matrizen

Sei A eine $n \times n$ $\mathcal{H}$–Matrix und $\underline{x} \in \mathbb{R}^n$ ein Vektor. Dann kann man $\underline{x}$ wegen $n = 2^p$ entsprechend der Blockstruktur von A in die Vektoren $\underline{x}_1, \underline{x}_2 \in \mathbb{R}^{\frac{n}{2}}$ zerlegen, womit sich für die Multiplikation $A\underline{x}$

$$A\underline{x} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} \underline{x}_1 \\ \underline{x}_2 \end{pmatrix} = \begin{pmatrix} A_{11}\underline{x}_1 + A_{12}\underline{x}_2 \\ A_{21}\underline{x}_1 + A_{22}\underline{x}_2 \end{pmatrix} \tag{3.14}$$

ergibt. Für die Berechnung von $A_{12}\underline{x}_2$ und $A_{21}\underline{x}_1$ erhält man nach Lemma 3.1 als Aufwand $(\frac{n}{2} + \frac{n}{2}) \cdot 2 = 2n$ Operationen. Dazu kommt dann noch zweimal der durch die Rekursion feststehende Aufwand für $A_{11}\underline{x}_1$ und $A_{22}\underline{x}_2$. Mit $N_{\mathrm{MV}}(p)$ seien nun die Kosten der Matrix–Vektor–Multiplikation $A\underline{x}$ bezeichnet. Dann ergibt sich aus

$$N_{\mathrm{MV}}(p) = 2N_{\mathrm{MV}}(p-1) + 2n \tag{3.15}$$

und $N_{\mathrm{MV}}(0) = 1$ das nächste Lemma.

Lemma 3.5. *Der Aufwand der Matrix–Vektor–Multiplikation einer $n \times n$ $\mathcal{H}$–Matrix mit einem allgemeinen Vektor $\underline{x}$ beträgt $2 \cdot 2^p \cdot p + \mathcal{O}(2^p) = 2n \cdot \log_2 n + \mathcal{O}(n)$ Operationen.*

Beweis. Der Beweis erfolgt wiederum durch vollständige Induktion über p. Der Induktionsanfang $N_{\mathrm{MV}}(0) = 0 + \mathcal{O}(1)$ ergibt sich sofort. Im Induktionsschritt ergibt sich aus der Rekursionsvorschrift (3.15) und der Induktionsvoraussetzung $N_{\mathrm{MV}}(p) = 2p \cdot 2^p + \mathcal{O}(2^p)$

$$\begin{aligned} N_{\mathrm{MV}}(p+1) &= 2N_{\mathrm{MV}}(p) + 2 \cdot 2^{p+1} = 2 \cdot (2p \cdot 2^p + \mathcal{O}(2^p)) + 2 \cdot 2^{p+1} \\ &= 2 \cdot 2^{p+1} \cdot (p+1) + \mathcal{O}(2^{p+1}). \end{aligned}$$

□

3.3.4 Matrix–Matrix–Multiplikation von $n \times n$ $\mathcal{H}$–Matrizen

Als bisheriges Ergebnis wurde erhalten, dass sowohl die Multiplikation zweier Rang–1–Matrizen als auch die Multiplikation einer $\mathcal{H}$–Matrix mit einer Rang–1–Matrix eine Rang–1–Matrix ergeben. Dies wird bei den folgenden Betrachtungen der Multiplikation von zwei $n \times n$ $\mathcal{H}$–Matrizen wiederverwendet werden. Wie schon im 4. Unterpunkt von Lemma 3.1 angedeutet, erfordert die Multiplikation zweier $n \times n$ Rang–1–Matrizen $A = \underline{a}_1 \underline{b}_1^T$ und $B = \underline{a}_2 \underline{b}_2^T$ wegen $AB = \underline{a}_1 \underline{b}_1^T \underline{a}_2 \underline{b}_2^T = (\underline{b}_1^T \underline{a}_2)\underline{a}_1 \underline{b}_2^T = \underline{a}_3 \underline{b}_2^T$ mit $\underline{a}_3 = (\underline{b}_1^T \underline{a}_2)\underline{a}_1$ $2n$ Operationen, was sich durch das Skalarprodukt $\underline{b}_1^T \underline{a}_2$ und eine weitere Multiplikation für jede Komponente von $\underline{a}_1$ ergibt. Als nächstes werden die Kosten $N_{\mathcal{H}*R1}(p)$ der Multiplikation einer $n \times n$ $\mathcal{H}$–Matrix A und der Rang–1–Matrix $\underline{a}\,\underline{b}^T$ untersucht, die sich wegen $AB = A\underline{a}\,\underline{b}^T = (A\underline{a})\underline{b}^T$ aus Lemma 3.5 über die Matrix–Vektor–Multiplikation zu $N_{\mathcal{H}*R1}(p) = 2 \cdot 2^p \cdot p + \mathcal{O}(2^p)$ ergeben. Um das approximative Produkt $A \otimes_{R1} B$ zweier $\mathcal{H}$–Matrizen zu definieren, benutzt man die Darstellung

$$AB = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix} = \begin{pmatrix} A_{11}B_{11} + A_{12}B_{21} & A_{11}B_{12} + A_{12}B_{22} \\ A_{21}B_{11} + A_{22}B_{21} & A_{21}B_{12} + A_{22}B_{22} \end{pmatrix} \tag{3.16}$$

und berechnet für jeden Block die bereits eingeführten Rang–1–Approximationen. Mit der Schreibweise $N_{\mathcal{H}*\mathcal{H}}(p)$ für die Kosten des approximativen Produkts $A \otimes_{R1} B$ zweier $\mathcal{H}$–Matrizen ergibt sich als Rekursionsgleichung mit Hilfe der vorigen Lemmata über die Aufwandsabschätzungen

$$\begin{aligned}
N_{\mathcal{H}*\mathcal{H}}(p) &= 2 \cdot (N_{\mathcal{H}*\mathcal{H}}(p-1) + N_{R1*R1}(p-1) + N_{\mathcal{H}+R1}(p-1)) \\
&+ 4N_{\mathcal{H}*R1}(p-1) + 2N_{R1+R1}(p-1) \\
&= 2N_{\mathcal{H}*\mathcal{H}}(p-1) + 2 \cdot (2 \cdot 2^{p-1}) + 2 \cdot (11 \cdot 2^{p-1} \cdot (p-1) + \mathcal{O}(2^{p-1})) \\
&+ 4(2 \cdot 2^{p-1} \cdot (p-1) + \mathcal{O}(2^{p-1})) + 2(6 \cdot 2^{p-1} + 5 \cdot 2^{p-1} + \mathcal{O}(1)) \\
&= 2N_{\mathcal{H}*\mathcal{H}}(p-1) + 2 \cdot 2^p + 11 \cdot 2^p \cdot (p-1) + \mathcal{O}(2^p) \\
&+ 4 \cdot 2^p \cdot (p-1) + 2\mathcal{O}(2^p) + 11 \cdot 2^p + 2\mathcal{O}(1) \\
&= 2N_{\mathcal{H}*\mathcal{H}}(p-1) + 15 \cdot p \cdot 2^p - 2 \cdot 2^p + \mathcal{O}(2^p), \qquad (3.17)
\end{aligned}$$

woraus man folgendes Lemma über die Matrix–Matrix–Multiplikation zweier $n \times n$ $\mathcal{H}$–Matrizen schließen kann:

Lemma 3.6. *Der Aufwand bei der Matrix–Matrix–Multiplikation zweier $n \times n$ $\mathcal{H}$–Matrizen beträgt $\frac{15}{2} \cdot 2^p \cdot p^2 + \mathcal{O}(p \cdot 2^p) = \frac{15}{2} n \cdot (\log_2 n)^2 + \mathcal{O}(p \cdot 2^p)$ Operationen.*

Beweis. Der Beweis erfolgt durch vollständige Induktion über p. Der Induktionsanfang $p = 0$ erweist sich sofort als richtig. Im Induktionsschritt $p > 0$ wird die Induktionsvoraussetzung $N_{\mathcal{H}*\mathcal{H}}(p) = \frac{15}{2} \cdot 2^p \cdot p^2 + \mathcal{O}(p \cdot 2^p)$ und die Rekursionsgleichung (3.17) verwendet, so dass sich

$$\begin{aligned}
N_{\mathcal{H}*\mathcal{H}}(p+1) &= 2N_{\mathcal{H}*\mathcal{H}}(p) + 15 \cdot (p+1) \cdot 2^{p+1} - 2 \cdot 2^{p+1} + \mathcal{O}(2^{p+1}) \\
&= 2 \cdot (\frac{15}{2} p^2 \cdot 2^p + \mathcal{O}(p \cdot 2^p)) + 15 \cdot p \cdot 2^{p+1} + 15 \cdot 2^{p+1} + \mathcal{O}(2^{p+1}) \\
&= \frac{15}{2} p^2 \cdot 2^{p+1} + \mathcal{O}(p \cdot 2^{p+1}) + 15 \cdot p \cdot 2^{p+1} + \frac{15}{2} \cdot 2^{p+1} + \mathcal{O}(2^{p+1}) \\
&= \frac{15}{2} \cdot (p+1)^2 \cdot 2^{p+1} + \mathcal{O}(p \cdot 2^{p+1}) + \mathcal{O}(2^{p+1}) \\
&= \frac{15}{2} \cdot 2^{p+1} \cdot (p+1)^2 + \mathcal{O}((p+1) \cdot 2^{p+1}),
\end{aligned}$$

also die Aussage für $p+1$, ergibt. □

3.3.5 Invertierung einer $n \times n$ $\mathcal{H}$–Matrix

Als letzte interessante Matrixoperation wird nun die Invertierung der $n \times n$ $\mathcal{H}$–Matrix

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}$$

untersucht, wobei A_{11}, A_{22} $\frac{n}{2} \times \frac{n}{2}$ $\mathcal{H}$–Matrizen und A_{12}, A_{21} Rang–1–Matrizen sind. Die approximative Inverse wird mit $B = \mathrm{Inv}_{R1}(A)$ bezeichnet und soll wiederum eine $n \times n$ $\mathcal{H}$–Matrix sein. Als Voraussetzung seien A und A_{11} invertierbar. Dann gilt nach der block-

weisen Anwendung des Gaußschen Eliminationsverfahrens

$$\begin{aligned}
\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} A^{-1} &= \begin{pmatrix} I & 0 \\ 0 & I \end{pmatrix} \\
\begin{pmatrix} I & A_{11}^{-1}A_{12} \\ A_{21} & A_{22} \end{pmatrix} A^{-1} &= \begin{pmatrix} A_{11}^{-1} & 0 \\ 0 & I \end{pmatrix} \\
\begin{pmatrix} I & A_{11}^{-1}A_{12} \\ 0 & A_{22} - A_{21}A_{11}^{-1}A_{12} \end{pmatrix} A^{-1} &= \begin{pmatrix} A_{11}^{-1} & 0 \\ -A_{21}A_{11}^{-1} & I \end{pmatrix} \\
\begin{pmatrix} I & A_{11}^{-1}A_{12} \\ 0 & I \end{pmatrix} A^{-1} &= \begin{pmatrix} A_{11}^{-1} & 0 \\ -(A_{22} - A_{21}A_{11}^{-1}A_{12})^{-1}A_{21}A_{11}^{-1} & (A_{22} - A_{21}A_{11}^{-1}A_{12})^{-1} \end{pmatrix} \\
\begin{pmatrix} I & 0 \\ 0 & I \end{pmatrix} A^{-1} &= \begin{pmatrix} A_{11}^{-1} + A_{11}^{-1}A_{12}S^{-1}A_{21}A_{11}^{-1} & -A_{11}^{-1}A_{12}S^{-1} \\ -S^{-1}A_{21}A_{11}^{-1} & S^{-1} \end{pmatrix}
\end{aligned} \tag{3.18}$$

mit dem sogenannten Schur–Komplement $S = A_{22} - A_{21}A_{11}^{-1}A_{12}$.

Die approximative Inverse $\mathrm{Inv}_{R1}(A)$ wird dadurch erhalten, dass die in den einzelnen Blöcken stehenden Matrizen approximativ berechnet werden. Dies geschieht zum einen durch approximative Invertierung von kleineren Matrizen, was durch Rekursion dann bereits erklärt ist, zum anderen durch die bereits besprochenen Additionen und Multiplikationen von $\mathcal{H}$– beziehungsweise Rang–1–Matrizen. Mit $N_{\mathrm{Inv}}(p)$ seien im Folgenden die Kosten bezeichnet, um eine $n \times n$ $\mathcal{H}$–Matrix approximativ zu berechnen. Für $p = 0$ sei $\mathrm{Inv}_{R1}(A) := A^{-1}$ als exakte Inverse definiert. Um den Gesamtaufwand der Invertierung zu erhalten, werden nun die durch das Gauß–Verfahren erhaltenen Blöcke betrachtet:

1. Beim Block rechts unten erhält man den Aufwand durch Berechnung und anschließende approximative Invertierung des Schur–Komplements S, so dass sich durch die approximative Invertierung $\mathrm{Inv}_{R1}(A_{11})$, die Multiplikationen der erhaltenen $\mathcal{H}$–Matrix mit den Rang–1–Matrizen A_{12} und daraufhin A_{21}, sowie durch die Addition der jetzt erhaltenen Rang–1–Matrix mit der $\mathcal{H}$–Matrix A_{22} und der abschließenden nochmaligen approximativen Invertierung eine Komplexität von

$$2N_{\mathrm{Inv}}(p-1) + N_{\mathcal{H}*R1}(p-1) + N_{R1*R1}(p-1) + N_{\mathcal{H}+R1}(p-1) \tag{3.19}$$

   ergibt.

2. Bei den Blöcken rechts oben und links unten erhält man zusammen einen Aufwand von

$$3N_{\mathcal{H}*R1}(p-1), \tag{3.20}$$

   wobei ausgenutzt wurde, dass $\mathrm{Inv}_{R1}(A_{11})A_{12}$ aus vorigen Berechnungen bereits bekannt ist.

3. Die Kosten des Blockes links oben ergeben sich zu

$$N_{R1*R1}(p-1) + N_{\mathcal{H}+R1}(p-1), \tag{3.21}$$

   da $\mathrm{Inv}_{R1}(A_{11})A_{12}$ sowie $\mathrm{Inv}_{R1}(S)A_{21}\mathrm{Inv}_{R1}(A_{11})$ bereits bekannt sind.

In der Blockstruktur von $\mathrm{Inv}_{R1}(A)$ kann man auch schön erkennen, dass es tatsächlich wieder eine $\mathcal{H}$–Matrix der bekannten Form mit kleineren $\mathcal{H}$–Matrizen auf der Hauptdiagonalen und Rang–1–Matrizen sonst ist. Zur Bestimmung des Gesamtaufwandes $N_{\mathrm{Inv}}(p)$ für die Invertierung einer $n \times n$ $\mathcal{H}$–Matrix werden die Kosten der einzelnen Blöcke aufsummiert, so dass sich als Rekursionsformel

$$\begin{aligned}
N_{\mathrm{Inv}}(p) &= 2N_{\mathrm{Inv}}(p-1) + 4N_{\mathcal{H}*R1}(p-1) + 2N_{R1*R1}(p-1) + 2N_{\mathcal{H}+R1}(p-1) \\
&= 2N_{\mathrm{Inv}}(p-1) + 4 \cdot (2 \cdot 2^{p-1} \cdot (p-1) + \mathcal{O}(2^{p-1})) + 2 \cdot (2 \cdot 2^{p-1}) \\
&+ 2 \cdot (11 \cdot 2^{p-1} \cdot (p-1) + \mathcal{O}(2^{p-1})) \\
&= 2N_{\mathrm{Inv}}(p-1) + 4 \cdot (p-1) \cdot 2^p + 2\mathcal{O}(2^p) + 2 \cdot 2^p + 11 \cdot (p-1) \cdot 2^p + \mathcal{O}(2^p) \\
&= 2N_{\mathrm{Inv}}(p-1) + 4 \cdot p \cdot 2^p - 4 \cdot 2^p + 2\mathcal{O}(2^p) + 2 \cdot 2^p + 11 \cdot p \cdot 2^p - 11 \cdot 2^p + \mathcal{O}(2^p) \\
&= 2N_{\mathrm{Inv}}(p-1) + 15 \cdot p \cdot 2^p - 13 \cdot 2^p + \mathcal{O}(2^p) \\
&= 2N_{\mathrm{Inv}}(p-1) + 15 \cdot p \cdot n - 13n + \mathcal{O}(n) \qquad (3.22)
\end{aligned}$$

ergibt. Hieraus lässt sich folgendes Lemma schließen:

Lemma 3.7. *Die approximative Invertierung einer $n \times n$ $\mathcal{H}$–Matrix erfordert einen Aufwand von $N_{Inv}(p) = \frac{15}{2}p^2 \cdot 2^p + \mathcal{O}(p \cdot 2^p) = \frac{15}{2}n \cdot (\log_2 n)^2 + \mathcal{O}(n \cdot \log_2 n)$ Operationen.*

Beweis. Der Beweis erfolgt durch vollständige Induktion über p. Der Induktionsanfang erweist sich sofort als richtig. Im Induktionsschritt verwenden wir die Induktionsvoraussetzung $N_{\mathrm{Inv}}(p) = \frac{15}{2}p^2 \cdot 2^p + \mathcal{O}(p \cdot 2^p)$ und die Rekursionsgleichung (3.22), um

$$\begin{aligned}
N_{\mathrm{Inv}}(p+1) &= 2N_{\mathrm{Inv}}(p) + 15 \cdot (p+1) \cdot 2^{p+1} - 13 \cdot 2^{p+1} + \mathcal{O}(2^{p+1}) \\
&= 2 \cdot (\frac{15}{2}p^2 \cdot 2^p + \mathcal{O}(p \cdot 2^p)) + 15 \cdot (p+1) \cdot 2^{p+1} - 13 \cdot 2^{p+1} + \mathcal{O}(2^{p+1}) \\
&= \frac{15}{2}p^2 \cdot 2^{p+1} + \mathcal{O}(p \cdot 2^{p+1}) + 15p \cdot 2^{p+1} + \frac{15}{2} \cdot 2^{p+1} + \mathcal{O}(2^{p+1}) \\
&= \frac{15}{2} \cdot (p+1)^2 \cdot 2^{p+1} + \mathcal{O}((p+1) \cdot 2^{p+1})
\end{aligned}$$

zu schließen. □

3.3.6 Zusammenfassung der Komplexitätsbetrachtungen

Zusammenfassend bekommt man im hier vorgestellten Sonderfall der Partitionierung für $n = 2^p$ sehr gute Ergebnisse für alle Matrixoperationen. Durch die spezielle Struktur der $\mathcal{H}$–Matrizen erhält man $\mathcal{O}(n \cdot (\log_2 n)^q)$ mit $q = 1, 2$, also nahezu linearen Aufwand. Im Gegensatz zur exakten Berechnung, die quadratischen oder sogar kubischen Aufwand verlangt, kann man bei großen Matrizen sehr viel Zeit einsparen. Allerdings muss man natürlich noch eine exakte Fehlerbetrachtung durchführen, um festzustellen, wie genau die vorgestellten Approximationen die exakten Ergebnisse annähern. Dazu sei auf die folgenden Kapitel verwiesen.

Kapitel 4

Komplexität und Implementierung

Michael Luttenberger

In den vorangegangenen Kapiteln wurden die Grundlagen der Hierarchischen Matrizen angesprochen. Inhalt dieses Kapitels sind die Datenstrukturen und Algorithmen, mit welchen die $\mathcal{H}$-Matrizen und die Operationen auf ihnen effizient realisiert werden könnnen, und deren Komplexität. Dieses Kapitel gliedert sich in zwei große Abschnitte:
Der erste Abschnitt soll grundlegende Begriffe einführen und gibt einen Überblick über die wichtigsten Punkte, welche bei einer effizienten Umsetzung von Algorithmen beachtet werden sollten. Abschnitt zwei skizziert die für die Implementierung von $\mathcal{H}$-Matrizen verwendeten Datenstrukturen und der hierauf aufbauenden Algorithmen und untersucht deren Komplexität.

4.1 Grundlagen

Der erste Teil dieses Kapitels soll einen Einblick geben, welche Punkte bei einer effizienten Implementierung von numerischen Algorithmen zu beachten sind. Für detailierte Informationen sei im Besonderen auf Kapitel 3 in [10] verwiesen. Der zweite Teil wird bekannte Definitionen kurz vorstellen, welche in den folgenden beiden Kapitel benötigt werden.

4.1.1 Implementierung

Dieser Abschnitt soll mit einem Beispiel eingeleitet werden. Hierfür soll die Matrix-Vektor-Multiplikation betrachtet werden. Für die Speicherung einer vollbesetzten Matrix in dem linear angeordneten Speicher eines Computers kommen zwei Anordnungen in Frage, welche als Column-Major und Row-Major bezeichnet werden.
Es sei folgende $n \times n$-Matrix gegeben:

$$\begin{pmatrix} a_{0,0} & a_{0,1} & \cdots & a_{0,n-1} \\ a_{1,0} & a_{1,1} & \cdots & a_{1,n-1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n-1,0} & a_{n-1,1} & \cdots & a_{n-1,n-1} \end{pmatrix}$$

Diese soll in einem Array double $A[n \times n]$ abgelegt werden. Die Column-Major-Anordnung

führt auf folgende Ablage der Matrixeinträge im Speicher:

$$A[r + c \cdot n] = a_{r,c} \rightsquigarrow \left(\begin{array}{cccccccc} a_{0,0} & a_{1,0} & \dots & a_{n-1,0} & a_{0,1} & a_{1,1} & \dots \end{array} \right),$$

während die Row-Major-Anordnung wie folgt aussieht:

$$A[c + r \cdot n] = a_{r,c} \rightsquigarrow \left(\begin{array}{cccccccc} a_{0,0} & a_{0,1} & \dots & a_{0,n-1} & a_{1,0} & a_{1,1} & \dots \end{array} \right).$$

Man betrachte zunächst die direkte Implementierung der Matrix-Vektor-Multiplikation nach der üblichen Formel

$$(y_r)_{r=0}^{n-1} = \sum_{c=0}^{n-1} a_{r,c} \cdot x_c.$$

Für das Column-Major Format erhält man somit

```
for( row = 0; row < n; row++ )
  for( col = 0; col < n; col++ )
    y[ row ] += A[ row + n * col ] * x[col];
```

Algorithmus 1: Matrix-Vektor-Multiplikation im Column-Major-Format (1)

und für das Row-Major Format

```
for( row = 0; row < n; row++ )
  for( col = 0; col < n; col++ )
    y[ row ] += A[ col + n * row ] * x[col];
```

Algorithmus 2: Matrix-Vektor-Multiplikation im Row-Major-Format (1)

Vergleicht man die Laufzeit beider Varianten auf einem AthlonXP 3000+ für $n = 4096$, so benötigt die Column-Major-Variante 1879 ms, die Row-Major-Variante jedoch nur 110 ms. Dabei wurde der Code mittels dem .NET Professional Compiler erzeugt, wobei alle Compiler-Schalter für die Optimierung der Geschwindigkeit gesetzt wurden.
Verwendet man dagegen folgende Varianten

```
for( col = 0; col < n; col++ ) {
  double tmp_x = x[col];
  const double *_A   = &A[ n * col ];
  for( row = 0, next_row = 1; row < n; row += 2, next_row +=2 ) {
    double tmp1 = _A[row] * tmp_x;
    double tmp2 = _A[next_row] * tmp_x;
    y[row     ] += tmp1;
    y[next_row] += tmp2;
  }
}
```

Algorithmus 3: Matrix-Vektor-Multiplikation im Column-Major-Format (2)

und

```
for( unsigned long row = 0; row < n; row += 2 ) {
  const double *A0 = &A[ row * n ];
  const double *A1 = A0 + n;
  double *y0 = &y[row];
  double *y1 = &y[row+1];
  for( unsigned long col = 0; col < n; ++col ) {
    double tmp1 = A0[col] * x[row];
    double tmp2 = A1[col] * x[row];
    (*y0) += tmp1;
    (*y1) += tmp2;
  }
}
```

Algorithmus 4: Matrix-Vektor-Multiplikation im Row-Major-Format (2)

so verringert sich die Laufzeit der Column-Major-Variante auf 113 ms und die der Row-Major-Variante auf 100 ms, bzw. im Fall von gcc-3.3.3 auf 100ms (Col) und 92ms (Row).[1]
Die Gründe für die Laufzeitverbesserungen der beiden zuletzt aufgeführten Versionen liegen sowohl in der sogenannten Cache-Lokalität der Daten als auch in der besseren Ausnutzung der Pipeline-Struktur des Prozessors. Darauf soll nun etwas genauer eingegangen werden.

Pipeline-Struktur von Prozessoren

Prozessoren unterteilen die für jeden Befehl abzuarbeitenden Schritte in mehrere Phasen. Eine einfache Unterteilung könnte z.B. Instruction Fetch (IF), Instruction Decude (ID), Instruction Execute (EX), Memory Load/Store (MEM), Write-Back (WB) sein. Der Befehlsdurchsatz wird nun dadurch erhöht, dass diese Phasen separat voneinander zu jedem Zeitpunkt an einem Befehl arbeiten. Dies wird als Pipelining bezeichnet. Der maximale Durchsatz ist natürlich nur möglich, falls das gegebene Programmfragment es erlaubt, die Befehle unabhängig voneinander in der Pipeline zu verarbeiten. Pipelining ist dabei natürlich nicht auf diese grobe Unterteilung beschränkt. So besitzen Prozessoren z.B. verschieden lange Pipelines für die Floating-Point-Einheit (FPU) und die Arithmetisch-Logische-Einheit (ALU).
Sind aufeinander folgende Befehle nicht unabhängig, so kann es zu sogenannten Pipeline-Konflikten kommen. Diese werden in verschiedene Klassen unterteilt. Die erste Klasse bilden die Strukturkonflikte, welche dadurch entstehen, dass ein Befehl auf eine funktionale Einheit (FU, z.B. Floating-Point-Einheit) zugreifen muss, diese jedoch durch einen vorhergehenden Befehl noch beschäftigt ist. Die Zeit, welche ein Befehl benötigt, bevor das Ergebnis (zumindest in der Pipeline) zur Verfügung steht, wird als Latenz bezeichnet. Eine Latenz von 0 Zyklen bedeutet dabei, dass das Ergebnis sofort von dem direkt nachfolgenden Befehl verwendet werden kann ohne zu warten. Übliche Auswege sind hierbei, dass häufig beanspruchte bzw. langsame FUs mehrmals zur Verfügung gestellt werden. Mittels entsprechender "Buchführung" (z.B. Instruction Window Buffer/Scoreboard) kann die CPU dann die Befehle - soweit sie unabhängig bezüglich der benötigten Daten sind - auf die verschiedenen FUs verteilen, bzw. auch umordnen oder Ergebnisse zwischenspeichern. Im Fall des Athlon finden sich z.B. drei ALUs auf der CPU, auf welche die Befehle durch sogenannte Instruction Control Unit und Scheduler verteilt werden.
Die zweite Klasse von Pipeline-Konflikten bilden die Datenkonflikte. Es seien A und B zwei Befehle, wobei A im Programmcode vor B steht. Dann werden drei Subtypen unterschieden:

- RAW ("read after write"): B will das Ergebnis von A lesen, dieses ist jedoch noch nicht bekannt / geschrieben.
  - In diese Klasse fällt auch der oben betrachtete Fall der Matrix-Vektor-Multiplikation. In der Pipeline existieren für diesen Fall entsprechende "Abkürzungen", welche es erlauben, Daten innerhalb der Pipeline an nachfolgende Befehle zu "forwarden", so dass diese zumindest nicht auf das explizite Schreiben der Daten in die Register oder in den Speicher warten müssen.
- WAR ("write after read"): B will an eine Speicherstelle/in ein Register schreiben, bevor A gelesen hat.

[1] Kompiliert man den Code mittels gcc-3.3.3 mit den flags -O3 -mcpu=athlon-xp, so benötigt die Row-Major-Variante ˜148ms bzw. 91ms, die Column-Major Variante ˜508ms bzw. 74ms.

- WAW ("write after write"): B will eine Speicherstelle/in ein Register schreiben, bevor A geschrieben hat.

  – Sowohl WAR als auch WAW können bei entsprechendem Parallelismus auf der CPU (durch z.B. mehrere FUs) eintreten.

Die dritte Klasse bilden schließlich die durch Sprünge verursachten Stalls der Pipeline. Da die unterschiedlichen Stufen der Pipeline zukünftige Befehle auf "Verdacht" verarbeitet, kann ein Sprungbefehl durch z.B. eine IF-ELSE-Anweisung diese Zwischenergebnisse unbrauchbar machen. Um den negativen Einfluss von Sprüngen zu verkleinern, besitzen Prozessoren daher Logiken für eine Sprungvorhersage (Branch-Prediction-Logic, BPL), um den wahrscheinlicheren Ast einer IF-ELSE-Anweisung vorherzusagen, und so die Pipeline mit den Befehlen dieses Astes zu laden. Im Fall der betrachteten Matrix-Vektor-Multiplikation wird diese BP die Pipeline mit den Befehlen des Schleifenrumpfes laden, anstatt möglichen Codes, welcher auf die Schleife folgt.
Wie bereits erwähnt, kann die CPU Befehle teilweise umordnen, um Pipeline-Konflikten vorzubeugen. Die CPU kann dabei jedoch nur eine "endliche Zeit in die Zukunft" blicken. Generell empfiehlt es sich daher - wie auch obiges Beispiel zeigt - möglichen Konflikten vorzubeugen. Die geläufigsten Techniken sind dabei das Entrollen, Umordnen und Zusammenfassen von Schleifen, welche meistens kombiniert werden. In [11, 15] können weitere Optimierungsansätze gefunden werden.
Schleifenentrollen wurde bei jeder der beiden zweiten Versionen angewandt, wobei dieses Beispiel bereits zeigt, dass man generell nicht darauf hoffen kann, dass der Compiler entsprechende Optimierungen erkennt. Ziel ist hierbei, den Overhead durch die Schleifenkontrolle, z.B. die jeweilige Sprungabfrage, zu verringern, aber auch mehr Parallelismus auf der Befehlsebene zu erlauben[2], durch entsprechendes Umordnen und Zusammenfassen von unabhängigen Befehlen zu Blöcken. Eine natürliche Obergrenze für das Entrollen einer Schleife bildet dabei die Anzahl der Register und der L1-Cache eines Prozessors. Sobald Zwischenergebnisse nicht mehr in Registern gehalten werden können, wird durch den hierdurch entstehen Overhead an Speicher-/Cachezugriffen der Durchsatz entsprechend gesenkt.
Dies führt auf den zweiten, für die Implementierung von numerischen Algorithmen wichtigen Punkt, die Haltung der Daten im Cache des Prozessors.

Cache-Lokalität

Sieht man von den Forwarding-Mechanismen innerhalb der Pipeline ab, so erlauben die Register den schnellsten Zugriff auf Daten durch Befehle. Daten aus den Registern können ohne zusätzliche Wartezyklen verarbeitet werden. Müssen die Daten jedoch zunächst aus dem Speicher geladen werden, so sind zusätzliche Wartezyklen notwendig. Um diese Wartezyklen so niedrig wie möglich zu halten, besitzen Prozessoren eine Hierarchie von Caches (Level 1 /

[2] Vergleicht man die beiden Row-Major Varianten, so muss die erste Variante auf das Ergebnis der Multiplikation warten, bevor die Addition ausgeführt werden kann. In der zweiten Variante wird es dem Compiler ermöglicht, die Unabhängigkeit der beiden Additionen und Multiplikationen auszunutzen. Hierdurch können zunächst die voneinander unabhängigen Multiplikationen ausgeführt werden, und daran anschließend die unabhängigen Additionen. Hierbei muss jedoch bemerkt werden, dass es durchaus Hardware gibt, welche sogenannte MADs (Multiplikation mit anschließender Addition) erlaubt, so z.B. die GPUs moderner Graphikkarten, auf welchen obige Aufteilung der Befehle in separate Addition und Multiplikation sich negativ auswirken könnte.

Level 2 / Level 3 Cache), welche mit wachsender Hierarchiestufe größer, aber auch langsamer werden. In diese Hierarchie ordnen sich dann Register quasi als L0 Cache ein, während Hauptspeicher und Festplatte als L4 bzw. L5 Cache betrachtet werden können. Die Caches dienen zur Zwischenspeicherung von bereits dekodierten Befehlen (L1-Befehlscache) und gelesenen Daten aus dem Speicher (L1-Daten / L2 / L3 Cache). Caches arbeiten in der Regel blockorientiert, das heisst, liest ein Befehl ein word (=32bit) ein, so wird statt nur ein word zu lesen, direkt z.B. ein qword (=128bit) gelesen, und im Cache zwischengespeichert[3]. Hierdurch kann z.B. bei Bearbeitung eines Arrays der folgende Speicherzugriff entfallen, da die benötigten Daten bereits im Cache liegen. Entsprechend unterstützen Prozessoren auch das "prefetching" von Daten, so dass z.B. die Daten für die nächste Schleifeniteration bereits im voraus angefordert und in den Cache geladen werden können. Ein Beispiel findet sich in [1, S. 72].
Hiermit lässt sich auch das extrem schlechte Abschneiden der ersten Version der Column-Major-Variante erklären, da hier der Prozessor die sich im Cache befindenden Daten nicht weiter nutzen kann, und somit auf direkte Speicherzugriffe zurückfallen muss.
Ein weit weniger triviales Beispiel für die Notwendigkeit der Cache-Lokalität findet sich in [19]. Dort wird auf den Seiten 28 bis 30 die Implementierung des Gauss-Seidel-Verfahrens im Fall der Poisson-Finite-Differenzen-Diskretisierung diskutiert.
Üblicherweise will man sich mit solchen Implementierungsdetails möglichst wenig beschäftigen und greift daher auf bestehende Bibliotheken zurück. Für numerische Algorithmen im Bereich der linearen Algebra finden sich hierfür auf *www.netlib.org* die Bibliotheken BLAS, LAPACK und ATLAS. Diese werden im abschließenden Abschnitt kurz vorgestellt.

BLAS, LAPACK, ATLAS

Für die hier betrachteten $\mathcal{H}$-Matrizen sind Algorithmen aus der linearen Algebra von Interesse. Auf *www.netlib.org* finden sich hierfür die Bibliotheken *BLAS* (Basic Linear Algebra Subprograms), *LAPACK* (Linear Algebra PACKage, setzt auf BLAS auf) und *ATLAS* (Automatically Tuned Linear Algebra Software). Sowohl BLAS als auch LAPACK sind ursprünglich in Fortran 77 geschrieben, und bieten jedoch mit CBLAS und CLAPACK C-Portierungen. ATLAS ist hingegen direkt in C implementiert[4].
Die Befehle von BLAS werden in 3 Level gegliedert. Level 1 BLAS Routinen stellen Vektor-Vektor und Vektor-Skalar Operationen zur Verfügung, wie das Skalieren, Addieren und dem Skalarprodukt von Vektoren. Level 2 umfasst Matrix-Vektor Operationen, und Level 3 Matrix-Matrix Operationen.
Hierauf setzt LAPACK auf. LAPACK bietet Implementierungen für viele numerische Algorithmen aus der linearen Algebra, so für die LU-Zerlegung, QR-Faktorisierung und die Singulärwertzerlegung. Ein ausführlicher User-Guide findet sich unter *www.netlib.org/lapack/lug*.
ATLAS umfasst den kompletten BLAS-Befehlsatz und einige LAPACK-Funktionen. Das Besondere an ATLAS stellt dabei die automatische Code-Generierung/Optimierung dar, welche die in den vorangegangenen Abschnitten angesprochenen Punkte für den Benutzer transparent umsetzt. Dies wird durch eine Kombination von u.a. für bestimmte Hardware optimierten Assembler-Code (z.B. 3dNow, SSE), nach z.B. Cachegröße parametrisierten Code, und einer

[3] Typischerweise ist es für Speicherbausteine auch effizienter, ganze lineare Speicherblöcke zu lesen, anstatt nur einzelne Speicherstellen.

[4] BLAS ist in der von *www.netlib.org* erhältlichen Version kaum optimiert. Ziel ist hier, eine einheitliche Schnittstelle zu definieren, so dass von Prozessorherstellern optimierte Implementierungen der BLAS-Bibliothek transparent für den Anwender angeboten und eingebunden werden können.

Suche nach den besten Komponenten erreicht.
Alle drei Bibliotheken halten sich an ein vorgegebens Namensschema für die Routinen, wobei ATLAS noch die Präfices *clapack_* und *cblas_* an die jeweiligen Befehl voranstellt. Der eigentliche Befehlsnamen ist von der Form $XYYZZZZ$, das heisst, jeder Befehl besteht aus höchstens 7 Zeichen. Hierbei gibt $X \in \{s, d, c, z\}$ den Datentyp an, wobei s dem C-Datentyp float und d double entspricht, während die Bezeichner c und z für komplexwertige Datentypen stehen. YY gibt das Matrixformat an, so steht GE für eine allgemeine Matrix, und DI z.B. für eine Diagonalmatrix. Schließlich codiert der aus höchstens vier Zeichen bestehende Suffix $ZZZZ$ die eigentliche Operation. Im Weiteren sind folgende Befehle von Bedeutung:

- ```
  void cblas_Xscal( const int N, SCALAR alpha, TYPE *x, const int incx );
  ```

  berechnet $\underline{x} \leftarrow \alpha \cdot \underline{x}$, wobei α und $\underline{x}$ in dem durch X signalisierten Datentyp gegeben sind, und incx den "Array-Stride" angibt, das heisst, nur die Elemente $x_0, x_{incx}, x_{2 \cdot incx}, \ldots$ werden skaliert.

- ```
  void cblas_Xcopy( const int N, TYPE *x, const int incx,
                                  TYPE *y, const int incy );
  ```

  kopiert die Werte $x_0, x_{incx}, \ldots, x_{(N-1) \cdot incx}$ nach $y_0, y_{incy}, \ldots, y_{(N-1) \cdot incy}$.

- ```
  void cblas_Xaxpy( const int N, SCALAR alpha, TYPE *x, const int incx,
                                                TYPE *y, const int incy );
  ```

  kopiert die Werte $\alpha \cdot x_0, \alpha \cdot x_{incx}, \ldots, \alpha \cdot x_{(N-1) \cdot incx}$ nach $y_0, y_{incy}, \ldots, y_{(N-1) \cdot incy}$.

- ```
  void cblas_Xgemv(const enum CBLAS_ORDER Order, const enum CBLAS_TRANSPOSE TA,
                   const int M, const int N, const SCALAR alpha, const TYPE *A,
                   const int lda, const TYPE *X, const int incX,
                   const TYPE beta, TYPE *Y, const int incY)
  ```

  berechnet $\underline{y} \leftarrow \alpha \cdot op(A) \cdot \underline{x} + \beta \cdot \underline{y}$. Dabei ist A eine $M \times N$ Matrix, und

  $$\text{Order} \in \{\text{CblasRowMajor}, \text{CblasColMajor}\}$$

  das Matrixformat von A an.

  $$\text{TA} \in \{\text{CblasNoTrans}, \text{CblasTrans}, \text{CblasConjTrans}\}$$

  signalisiert, ob für $op(A)$ A bzw. $A^\top$ bzw. $\overline{A}^\top$ verwendet werden soll. Der Parameter lda muss schließlich die Bedingung $lda \geq \max\{1, M\}$erfüllen. Der Wert von lda gibt an, dass die Matrix A in einem $lda \times N$ Array abgespeichert ist. Für den hier im weiteren relevanten Fall des Column-Major-Formats bedeutet dies, dass die Matrix als eindimensionaler Array wie folgt abgelegt ist:

  $$a_{r,c} = A[r + c \cdot lda] \rightsquigarrow \begin{pmatrix} a_{0,0} & \ldots & a_{m-1,0} & a_{m,0} & \ldots & a_{lda-1,0} & a_{0,1} & \ldots \end{pmatrix}$$

  Hiermit können Teilmatrizen übergeben werden. Soll z.B. von einer 4×4 Matrix A (Column-Major Format) die Teilmatrix $\begin{pmatrix} a_{2,2} & a_{2,3} \\ a_{3,2} & a_{3,3} \end{pmatrix}$ übergeben werden, so kann dies in C mittels $\&A[10]$ und $lda = 4$ geschehen.

- ```
  void cblas_Xgemm(const enum CBLAS_ORDER Order,
                   const enum CBLAS_TRANSPOSE TA, const enum CBLAS_TRANSPOSE TB,
                   const int M, const int N, const int K,
                   const TYPE * alpha, const TYPE *A, const int lda,
                   const TYPE *B, const int ldb, const TYPE * beta,
                   TYPE *C, const int ldc)
  ```
  ```

führt die Operation $C \leftarrow \alpha \cdot op(A) \cdot op(B) + \beta \cdot C$ aus.

- ```
  int clapack_Xgetrf(const enum CBLAS_ORDER Order, const int M, const int N,
                     TYPE *A, const int lda, int *ipiv)
  ```

  berechnet die LU-Zerlegung von A mit $AP = LU$, wobei P eine Permutationsmatrix ist, welche in dem Array ipiv zurückgegeben wird. A wird mit den Matrizen L und U überschrieben.

- ```
  int clapack_Xgetri(const enum CBLAS_ORDER Order, const int N, TYPE *A,
                     const int lda, const int *ipiv)
  ```

  berechnet A^{-1} nachdem A mittels GETRF LU-faktorisiert wurde.

- ```
  int Xgeqrf_(integer *m, integer *n, TYPE *A, integer * lda,
              TYPE *tau, TYPE *work, integer *lwork, integer *info)
  ```

  berechnet eine QR-Zerlegung von einer $m \times n$ Matrix A (gespeichert in einem $lda \times n$ Array), wobei A überschrieben wird, und Q implizit als Produkt von Householder-Transformationen repräsentiert wird.

- ```
  int Xorgqr_(integer *m, integer *n, integer *k, TYPE * A,
              integer *lda, TYPE *tau, TYPE *work, integer *lwork,
              integer *info)
  ```

  Rechnet die durch Xgeqrf erhaltene Matrix Q in die explizite Darstellung um.

- ```
  int Xgesvd_(char *jobU, char *jobVt, integer *m, integer *n,
              TYPE *A, integer *lda, TYPE *S, TYPE *U, integer * ldU,
              TYPE *Vt, integer *ldVt, TYPE *work, integer *lwork,
              integer *info)
  ```

  Berechnet eine Singulärwertzerlegung $USV^\top$ von A. Die Parameter $jobU$ und $jobVt$ bekommen als Wert einen String übergeben. $'A'$ bedeutet dabei, dass die jeweilige Matrix vollständig berechnet werden soll, während bei $'S'$ nur die ersten $\min\{M, N\}$ Spalten von U bzw. V berechnet werden.

Soweit ATLAS die entsprechenden Routinen von BLAS bzw. LAPACK anbietet, wurden diese oben angegeben. Nur für die letzten drei Befehle existieren zur Zeit keine ATLAS-Implementierungen. Anzumerken ist noch, dass, während ATLAS sowohl das Column-Major- als auch das Row-Major-Format unterstützt, LAPACK und BLAS im Allgemeinen nur das Column-Major-Format unterstützen, da Fortran dies für das Speichern eines zwei-dimensionalen Arrays verwendet.

4.1.2 Grundlegende Notationen und Definitionen zu $\mathcal{H}$-Matrizen

In diesem Abschnitt sollen die wichtigsten Definitionen und Notationen für $\mathcal{H}$-Matrizen zusammengefasst werden.

Definition 4.1. Hierfür sei zunächst $d \in \mathbb{N}$ mit $d > 0$ fest gegeben. Die Komponenten von $x \in \mathbb{R}^d$ werden für $k \in \{1, \ldots, d\}$ mit $x^{(k)}$ bezeichnet. Auf $\mathbb{R}^d$ wird die übliche partielle Ordnung $\leq \subseteq \mathbb{R}^d \times \mathbb{R}^d := \{(x, y) \mid \forall i \in \{1, \ldots, d\} : x^{(i)} \leq y^{(i)}\}$ vorausgesetzt. Für $a, b \in \mathbb{R}^d$ sei dann $[a, b] := \{x \in \mathbb{R}^d \mid \min^{\leq}\{a, b\} \leq x \leq \max^{\leq}\{a, b\}\}$ der von a und b beschriebene *Quader* im $\mathbb{R}^d$. Weiterhin sei für $c \in \mathbb{R}^d, \lambda \in \mathbb{R}$ definiert: $c + \lambda \cdot [a, b] := [c + \lambda \cdot a, c + \lambda \cdot b]$.
Durch $\operatorname{diam}([a, b]) := \|a - b\|_2$ wird jedem Quader sein Durchmesser zugeordnet. Für $A, B \subseteq \mathbb{R}^d$ bezeichnet wie üblich $\operatorname{dist}(A, B) := \inf_{a \in A, b \in B} \|a - b\|_2 \geq 0$ die Distanz der beiden Mengen voneinander. Hieraus folgt $\operatorname{dist}([a, b], [c, d]) = \sqrt{\sum_{i=1}^{d} \operatorname{dist}(\left[a^{(i)}, b^{(i)}\right], \left[c^{(i)}, d^{(i)}\right])^2}$.
  ```

$\mathcal{I}$ sei eine beliebige, endliche Menge und $M_{\mathcal{I}} := \{m_i \in \mathbb{R}^d \mid i \in \mathcal{I}\}$ eine beliebige, mit $\mathcal{I}$ indizierte Menge von Punkten des $\mathbb{R}^d$. Für jeden Punkt m_i seien weiterhin $a_i, b_i \in \mathbb{R}^d$ mit $a_i \leq b_i$ so gegeben, dass $m_i \in [a_i, b_i]$. Dann wird der Quader $B_i := [a_i, b_i]$ als die *Axis-Aligned-Bounding-Box* (AABB) von m_i bezeichnet. Für $I \subseteq \mathcal{I}$ sei $M_I := \{m_i \mid i \in I\}$ und $B_I := [a_I, b_I]$ mit $a_I := \max^{\leq}\{a \in \mathbb{R}^d \mid \forall i \in I : a \leq a_i\}$und $b_I := \min^{\leq}\{b \in \mathbb{R}^d \mid \forall i \in I : b \geq b_i\}$, so dass $\bigcup\{B_i \mid i \in I\} \subseteq B_I$ gilt. B_I ist somit der kleinste Quader, welcher alle B_j mit $j \in I$ umfasst.

Weiter wird der Begriff der Zulässigkeit zweier Mengen $J, K \subseteq I$ benötigt. Hier wird nur der einfachste Fall betrachtet, in welchem die Zulässigkeit auf die Boundingboxen B_J und B_K zurückgeführt wird:

Definition 4.2. Sei $\eta \in \mathbb{R}^+$ gegeben. Dann heißen die Teilmengen $J, K \subseteq \mathcal{I}$ zueinander zulässig (bezüglich η), falls

$$\min\{\operatorname{diam}(B_J), \operatorname{diam}(B_K)) \leq \eta \cdot \operatorname{dist}(B_J, B_K)$$

erfüllt ist.

Zur Beschreibung der Cluster-Bäume werden noch folgende Begriffe benötigt.

Definition 4.3. Ein (endlicher) *gerichteter Graph* $G = (V, E)$ besteht aus einer endlichen Menge V, welche als Knotenmenge bezeichnet wird, und der als Kantenrelation bezeichneten Mengen $E \subseteq V \times V$. Die Elemente von V werden als Knoten bezeichnet. Ist der Graph nicht durch den Kontext eindeutig bestimmt, so wird V_G und E_G geschrieben.
Für $u \in V$ bezeichnet $uE = \{w \in V \mid (u, w) \in E\}$ die Menge der (direkten) Nachfolger von u. Entsprechend wird mit $Ew = \{u \in V \mid (u, w) \in E\}$ die Menge der (direkten) Vorgänger von w bezeichnet. Ein Pfad π in G von einem Knoten u zu einem Knoten w wird als Wort über$V$ geschrieben, das heisst, $\pi \in V^* := \bigcup_{n\in\mathbb{N}} V^n$ mit $\pi_0 = u$, $\pi_{|\pi|-1} = w$ und $\forall i \in |\pi| : (v_i, v_{i+1}) \in E$. Dabei wird die übliche Einbettung der natürlichen Zahlen in die Mengentheorie durch Identifikation einer Zahl $n \in \mathbb{N}$ mit der Menge ihrer Vorgänger $\{0, \ldots, n-1\}$, insbesondere also $0 \equiv \emptyset$ verwendet. Um die Mächtigkeit einer Menge anzugeben, wird sowohl $|\cdot|$ also auch $\#$ geschrieben werden.
Für $R \subseteq A \times B$ und $S \subseteq B \times C$ sei $R \circ S := \{(a, c) \in A \times C \mid \exists b \in B : (a, b) \in R \wedge (b, c) \in S\}$. Hiermit gelte: $E^0 := \{(v, v) \mid v \in V\}$, $E^1 := E$ und $E^{n+1} := E^n \circ E$. Damit gibt uE^n alle Nachfolger von u an, welche über einen Pfad genau der Länge n erreichbar sind. Weiterhin sei $E^* := \bigcup_{n\in\mathbb{N}} E^n$ und $E^+ := \bigcup_{n>0} E^n$. Damit ist uE^* die Menge aller Knoten, welche von u aus erreichbar sind. Entsprechend ist E^*w die Menge aller Knoten in G, welche einen Pfad nach w besitzen.
Ein Pfad π in G heißt Zyklus, falls $\pi_0 = \pi_{|\pi|-1}$. Ein Graph heißt *azyklisch*, falls es keine Zyklus in ihm gibt, das heisst, $\forall v \in V : v \notin vE^*$ bzw. äquivalent $\forall v \in V : v \notin E^+v$. Ein azyklischer (hier ohne Einschränkung gerichteter) Graph heißt (gerichteter) *Baum*, falls es einen eindeutig bestimmten Knoten $r \in V$ mit $rE^* = V \wedge Er = \emptyset$ gibt. r wird dann als Wurzel des Baums G bezeichnet. Jedem Knoten wird mittels $h : V \to \mathbb{N} : v \mapsto \min\{n \in \mathbb{N} \mid v \in rE^n\}$ eine Distanz zu r definiert, welche als Höhe des Knotens in dem Baum G mit Wurzel r bezeichnet wird. Die Höhe des Baums G ist dann durch $h_G := \max_{u\in V} h(u)$ gegeben. Ein Knoten $u \in V$ heißt Blatt, falls $uE = \emptyset$ gilt. Die Menge der Blätter von G wird mit $L := \{u \in V \mid uE = \emptyset\}$ bezeichnet. Dann ist $L_i := L \cap h^{-1}(i)$ die Menge aller Blätter mit Höhe i und $L_{\leq i} = \bigcup_{0\leq k\leq i} L_k$ der Menge der Blätter mit Höhe $\leq i$. Entsprechend ist $V_i := V \cap h^{-1}(i)$ die Menge aller Knoten mit Höhe

i und $V_{\leq i} := \bigcup_{0\leq k\leq i} V_i$. Alternativ wird für einen Baum auch $T = (V, E)$ geschrieben. Die Nachfolger uE eines Baumknotens werden in diesem Fall auch als Söhne von u bezeichnet. Da T ein Baum ist, gilt $|Ew| \leq 1$ und Ew wird ohne Einschränkung mit seinem - soweit existent - enthaltenen Knoten identifiziert. Ew wird dann als Vater(-knoten) von w bezeichnet.
Falls notwendig für das Verständnis werden die Mengen L, V, E mit dem zugehörigen Graphen G indiziert, so z.B. $L_{G,\leq i}$.

Für die Beschreibung des Clusterings spezialisiert man den Begriff des Baums weiter:

Definition 4.4. Sei I eine endliche Menge. Ein *(hierarchischer) Partitionsbaum* über $\mathcal{I}$ ist ein Baum $T_\mathcal{I} = (V_\mathcal{I}, E_\mathcal{I})$ mit $V_\mathcal{I} \subseteq \mathcal{P}(\mathcal{I})$, wobei $\mathcal{I}$ die Wurzel von $T_\mathcal{I}$ ist und

$$\forall u \in V_\mathcal{I} : uE_\mathcal{I} \neq \emptyset \Rightarrow \left(\bigcup_{w\in uE_\mathcal{I}} w = u \wedge \forall x, y \in uE_\mathcal{I} : x \cap y \neq \emptyset \Leftrightarrow x = y \right)$$

gilt. Dabei ist $\mathcal{P}(\mathcal{I})$ die Potenzmenge von $\mathcal{I}$. Synonym für Partitionsbaum wird auch $\mathcal{H}$-Baum verwendet. Mittels Induktion ergibt sich hieraus sofort, dass $\forall h \leq h_{T_\mathcal{I}} : \dot{\bigcup} L_{T_\mathcal{I},\leq h} = \mathcal{I}$ gilt.
Entsprechend ist für endliche Mengen $\mathcal{I}$ und $\mathcal{J}$ ein Baum $T_{\mathcal{I}\times\mathcal{J}} = (V_{\mathcal{I}\times\mathcal{J}}, E_{\mathcal{I}\times\mathcal{J}})$ ein *(hierarchischer) Produktpartitionsbaum* (kurz $\mathcal{H}_\times$-Baum) über $\mathcal{I} \times \mathcal{J}$, falls $T_{\mathcal{I}\times\mathcal{J}}$ ein Partitionsbaum über $\mathcal{I} \times \mathcal{J}$ ist mit der zusätzlichen Bedingung:

$$\forall u \in V_{\mathcal{I}\times\mathcal{J}} : \exists I \subseteq \mathcal{I}\ \exists J \subseteq \mathcal{I} : u = I \times J.$$

Damit lässt sich jeder Knoten von $T_{\mathcal{I}\times\mathcal{J}}$ als kartesisches Produkt schreiben, so dass jedem Knoten eines $\mathcal{H}_\times$-Baums eine Matrix zugeordnet werden kann.
Für zwei Partitionsbäume $T_\mathcal{I}$ und $T_\mathcal{J}$ ist schließlich der kanonische Produktpartitionsbaum $T_\mathcal{I} \otimes T_\mathcal{J} = \left(V_{T_\mathcal{I}\otimes T_\mathcal{J}}, E_{T_\mathcal{I}\otimes T_\mathcal{J}}\right)$ der $\mathcal{H}_\times$-Baum über $\mathcal{I} \times \mathcal{J}$ mit Höhe $h_{T_\mathcal{I}\otimes T_\mathcal{J}} = \min\left\{h_{T_\mathcal{I}}, h_{T_\mathcal{J}}\right\}$ und

$$\forall\,(I \times J) \in T_\mathcal{I} \otimes T_\mathcal{J} : I \in V_{T_\mathcal{I}} \wedge J \in V_{T_\mathcal{J}} \wedge (I \times J)\, E_{T_\mathcal{I}\otimes T_\mathcal{J}} = \left\{I' \times J' \mid I' \in IE_{T_\mathcal{I}} \wedge J' \in JE_{T_\mathcal{J}}\right\}.$$

Für gegebene $\mathcal{H}$-Bäume $T_\mathcal{I}$ und T_J heißt ein $\mathcal{H}_\times$-Baum $T = (V, E)$ über $\mathcal{I} \times \mathcal{J}$ aus $T_\mathcal{I}$ *und* $T_\mathcal{J}$ *gebildet*, falls $V \subseteq V_{T_\mathcal{I}\otimes T_\mathcal{J}}$ und $E \subseteq E_{T_\mathcal{I}\otimes T_\mathcal{J}}$ gilt.

Bemerkung 4.1. Das heisst, T entsteht aus $T_\mathcal{I} \otimes T_\mathcal{J}$ durch Streichen der Kanten $E_{T_\mathcal{I}\otimes T_\mathcal{J}} \setminus E$ und Entfernen der Knoten $V_{T_\mathcal{I}\otimes T_\mathcal{J}} \setminus (\mathcal{I} \times \mathcal{J})E^*$. Da T ein $\mathcal{H}_\times$-Baum und damit ein $\mathcal{H}$-Baum über $\mathcal{I} \times \mathcal{J}$ ist, folgt $\forall u \in T : uE \in \{\emptyset, uE_{T_\mathcal{I}\otimes T_\mathcal{J}}\}$, das heisst, für einen gegebenen Knoten $u \in V$ werden entweder alle ausgehenden Kanten gestrichen, oder keine. Die Blätter von T bilden daher eine Partition von $\mathcal{I} \times \mathcal{J}$, und können für eine gegebene Matrix $A \in \mathbb{R}^{\mathcal{I}\times\mathcal{J}}$ mit deren Submatrizen $A\,|_{r\times s\in L_T}$ identifiziert werden.

Definition 4.5. Sei T ein aus $T_\mathcal{I}$ und $T_\mathcal{J}$ gebildeter $\mathcal{H}_\times$-Baum über $\mathcal{I} \times \mathcal{J}$. Dann heißt ein Blatt $I \times J \in L_T$ *zulässig*, falls $I \times J$ die Zulässigkeitsbedingung aus Definition 4.2 erfüllt. Ansonsten heißt $I \times J$ *nicht zulässig*. Mit L_T^Z seien die zulässigen Blätter von T bezeichnet. Entsprechend sei $L_T^N := L_T \setminus L_T^Z$ die Menge der nicht zulässigen Blätter.
Für $k \in \mathbb{N}$ ist dann die Klasse $\mathcal{H}(k, T)$ der $\mathcal{H}$-Matrizen über T wie folgt definiert: Sei $M \in \mathcal{H}(k, T)$. Ist $I \times J \in L_T^Z$, dann liegt $M\,|_{I\times J}$ im $\mathcal{R}_k$-Format vor, das heisst, es gilt $M\,|_{I\times J} = AB^\top$ für geeignete Matrizen $A \in \mathbb{R}^{\#I\times k}$ und $B \in \mathbb{R}^{\#J\times k}$. Ansonsten ist für $I \times J \in L_T^N$ $M\,|_{I\times J} \in \mathbb{R}^{\#I\times\#J}$.

Damit sind die wichtigsten Begriffe definiert.

4.2 Komplexitätsabschätzungen

Im Folgenden soll der Speicher- und Rechenaufwand für Hierarchische Matrizen und für die auf ihnen aufbauenden Algorithmen abgeschätzt werden. Hierfür werden beginnend bei der Konstruktion des Clusterbaums $T_{\mathcal{I}}$ schrittweise die jeweiligen Algorithmen und Datenstrukturen zunächst in Pseudo-Code beschrieben und entsprechend untersucht. Für dieses Kapitel sei eine feste (endliche) Indexmenge $\mathcal{I}$ mit zugehörigem Datensatz $M = \{m_i \in \mathbb{R}^d \mid i \in \mathcal{I}\}$ und Boundingboxen $\mathcal{B} = \{B_i \subset \mathbb{R}^d \mid i \in \mathcal{I}\}$ fixiert für eine feste Raumdimension $d \in \mathbb{N}$.

4.2.1 Konstruktion des Cluster-Baums T_I

Es wird im Folgenden nur die geometrische Konstruktion von $T_{\mathcal{I}}$ aufbauend auf den bekannten Quadtree bzw. Qctree Algorithmen betrachtet. Diese erleichtert die Komplexitätsabschätzungen.

Definition 4.6. Durch $C_{\max} \in \mathbb{N}$ sei die maximale Größe eines Blattes von $T_{\mathcal{I}}$ vorgegeben, das heisst, $\forall v \in V_{T_{\mathcal{I}}} : vE_{T_{\mathcal{I}}} = \emptyset \Leftrightarrow \#v \leq C_{\max}$. Mit $C_{\mathcal{I}} = [l_{\mathcal{I}}, u_{\mathcal{I}}]$ sei der kleinste $M_{\mathcal{I}}$ umfassende Quader bezeichnet, das heisst, $l_{\mathcal{I}} := \max^{\leq} \{a \in \mathbb{R}^d \mid \forall i \in \mathcal{I} : a \leq m_i\}$ und $u_{\mathcal{I}} := \min^{\leq} \{b \in \mathbb{R}^d \mid \forall i \in \mathcal{I} : b \geq m_i\}$. Weiterhin sei $\Lambda_{\mathcal{I}}$ die längste Kante von $C_{\mathcal{I}}$.

Beginnend bei $J = \mathcal{I}$ und $C = C_{\mathcal{I}}$ lässt sich die Konstruktion von $T_{\mathcal{I}}$ wie folgt beschreiben:

1. Es gelte $0 < \#I \leq C_{\max}$:
   (a) Gilt $J = \emptyset$, so bricht die Rekursion ab.
   (b) Ansonsten ist J ein Blatt von $T_{\mathcal{I}}$ und die Rekursion bricht ebenfalls ab. Berechne B_J und gib B_J zurück.
2. Sei $\#I > C_{\max}$. Aus $C = [c, d]$ entstehen durch einmalige Halbierung aller Kanten für $k \in \{0,1\}^d = 2^d$ die halb-offenen Quader

$$C^{(k)} := \{x \in C \mid \forall_{1 \leq n \leq d} k^{(n)} = 0 \Leftrightarrow x^{(n)} \leq \frac{d^{(n)} - c^{(n)}}{2}\}.$$

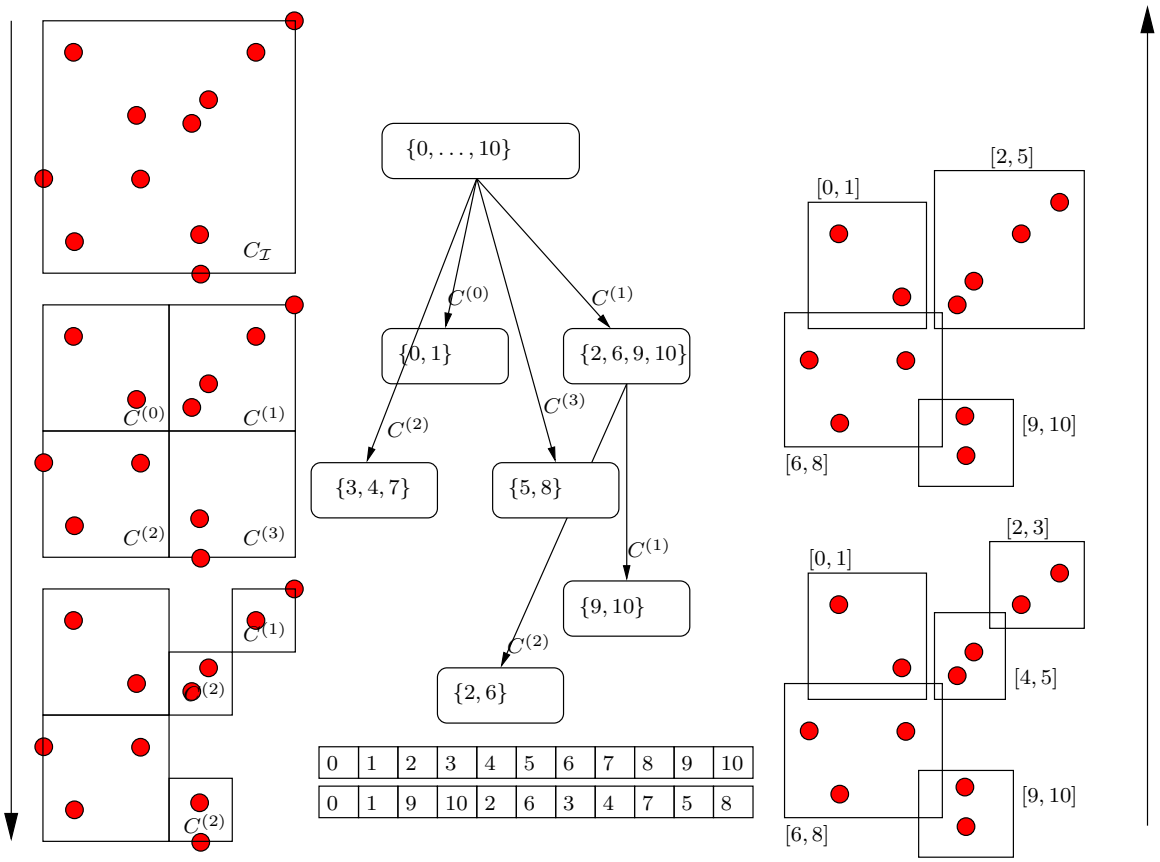
   Die Punkte M_J werden mittels $M_J \cap C^{(k)}$ partitioniert.
   (a) Existiert ein $l \in 2^d$, so dass $M_J \subseteq C^{(l)}$, so ist J kein Knoten von $T_{\mathcal{I}}$ und es wird mit J und $\overline{C^{(l)}}$ rekursiv abgestiegen. Die zurückgegebene Boundingbox B_J wird an die aufrufende Instanz weitergeleitet.
   (b) Ansonsten ist J ein innerer Knoten von $T_{\mathcal{I}}$ mit $JE_{T_{\mathcal{I}}} \geq 2$ und für alle $l \in 2^d$ wird mit $J \cap C^{(l)}$ und $C_J := \overline{C^{(l)}}$ rekursiv entsprechend verfahren. Berechne aus den erhaltenen Boundingboxen $B_0, \ldots, B_{2^d-1}$ die Boundingbox B_J.

Algorithmus 5: Konstruktion von T_I

Offensichtlich erfüllt der so erhaltene Baum über $\mathcal{I}$ tatsächlich die Eigenschaften eines $\mathcal{H}$-Baums. Wegen 2(a) folgt insbesondere, dass jeder Knoten von $T_{\mathcal{I}}$ entweder ein Blatt ist, oder mindestens zwei Nachfolger besitzt. Damit ergibt sich sofort mittels Induktion über $\#\mathcal{I}$, dass

der so konstruierte Baum $T_{\mathcal{I}}$ höchstens $2 \cdot \#\mathcal{I} - 1$ Knoten besitzt. Die für die Konstruktion von $T_{\mathcal{I}}$ benötigte Zeit ergibt sich aus der Zeit, welche in jedem rekursivem Aufruf verbraucht wird und der Anzahl der Knoten des Aufrufbaums. Zunächst sei der Aufwand für die Berechnung der B_J vernachlässigt.
In jedem Aufruf müssen die Punkte M_J mittels jeweils d Vergleichen auf die 2^d Quader aufgeteilt werden, wofür $O(d \cdot \#J)$ Zeit benötigt wird. Insbesondere werden - auf Grund der Partitionierung - auf jeder Ebene des Aufrufgraphens höchstens $\#\mathcal{I}$ Punkte aus M_I betrachtet, so dass sich die Laufzeit durch $O(d \cdot \#\mathcal{I})$ mal der Höhe des durch die rekursiven Aufrufe gebildeten Baums abschätzen lässt, welches offensichtlich auch eine obere Schranke für die Höhe von $T_{\mathcal{I}}$ ist.

Abbildung 4.1: Clusterbaum $T_{\mathcal{I}}$ zu $C_{\max} = 3$



Nach $r \geq 0$ rekursiven Abstiegen entspricht der Wert des Parameters C bis auf Translation dem Quader $2^{-r} \cdot C_{\mathcal{I}}$. Soll der Baum maximal die Höhe h besitzen, so ist hierfür hinreichend, dass sich in jedem Quader der Form $2^{-h} \cdot C_{\mathcal{I}}$ höchstens $C_{\max}$ viele Punkte aus $M_{\mathcal{I}}$ befinden. Man definiert daher:

Definition 4.7. Die Punktmenge $M_{\mathcal{I}}$ ist zu C_{sep} *separierbar*, falls es ein $\underline{\lambda} \in \mathbb{R}$ gibt, so dass

$$\forall c \in \mathbb{R}^d : \#\{i \in \mathcal{I} \mid x_i \in c + [0, \underline{\lambda}]^d\} \leq C_{\text{sep}}$$

gilt. Ohne Einschränkung sei im Weiteren $M_{\mathcal{I}}$ zu $C_{\max}$ für ein maximales $\underline{\lambda}$ separierbar.

Hiermit folgt $h_{T_\mathcal{I}} \leq \left\lceil log_2 \frac{\Lambda_\mathcal{I}}{\underline{\lambda}} \right\rceil$. Ist $\mathcal{I}$ in Form eines Arrays $\mathbb{A}$ gegeben und sortiert man in jedem Schritt $\mathbb{A}\,|_J$ entsprechend der Partitionierung in Schritt 2, so muss nicht explizit J in jedem Aufruf weitergegeben werden, sondern nur der erste Index in $\mathbb{A}$, an welchem J beginnt, und $\#J$. Entsprechend kann $T_\mathcal{I}$ abgespeichert werden. Für die Sortierung von J bezüglich der Quader $C^{(l)}$ kann z.B. eine Quicksort-Variante verwendet werden, welche (im Durchschnitt) nur Zeit $O(\#J \cdot \log 2^d) = O(d \cdot \#J)$ benötigt, siehe hierfür [14, S. 96/97].

Es verbleibt der Aufwand für die Berechnung und das Speichern der Boundingboxen B_J. In den Blättern lässt sich dieser durch das $C_{\max}$-malige komponentenweise Bestimmen des Maximums und Minimums aus den B_j abschätzen, was auf $O(d \cdot C_{\max})$ führt. Das rekursive Zusammensetzen der B_J in den inneren Knoten von $T_\mathcal{I}$ lässt sich entsprechend mit $O(d \cdot 2^d) = O(1)$ angeben. Der zusätzlich benötigte Speicherplatz pro Knoten des $\mathcal{H}$-Baums ist offensichtlich durch $O(d)$ beschränkt, da für jede Box B_J nur die zwei Eckpunkte $a_J, b_J \in \mathbb{R}^d$ gespeichert werden müssen.

Schließlich fällt in jedem inneren Knoten noch der benötigte Speicher für Zeiger auf die Nachfolger an. Dieser ist durch $O(2^d)$ beschränkt, und somit als konstant anzusehen.

Lemma 4.1. *Die Konstruktion von $T_\mathcal{I}$ nach Algorithmus 5 benötigt $O(\#\mathcal{I} \cdot \left\lceil log_2 \frac{\Lambda_\mathcal{I}}{\underline{\lambda}} \right\rceil)$ Zeit. Weiterhin gilt $\#V_{T_\mathcal{I}} \leq 2 \cdot \#\mathcal{I}$ und $\forall I \in V_{T_\mathcal{I}} : |IE_{T_\mathcal{I}}| \in \{0, 2, \ldots, 2^d\}$. Der zur Konstruktion und Speicherung von $T_\mathcal{I}$ benötigte Platz ist durch $O(\#V_{T_\mathcal{I}}) = O(\#I)$ nach oben beschränkt.*

4.2.2 Konstruktion des Blockclusterbaums T

Seien T_I und T_J zwei $\mathcal{H}$-Bäume zu den Indexmengen I und J. Bekanntlich werden die $\mathcal{H}$-Matrizen bezüglich einem über T_I und T_J gebildeten $\mathcal{H}_\times$-Baum T und einer Zulässigkeitsbedingung definiert. Wie bereits erwähnt, soll hier nur die in Definition 4.2 erklärte vereinfachte Zulässigkeitsbedingung betrachtet werden für festes $\eta \in \mathbb{R}^+$. Hiermit lässt sich die Konstruktion von T beginnend bei $I = \mathcal{I}$ und $J = \mathcal{J}$ wie folgt beschreiben[5]:

1. Ist $I \times J$ zulässig, d.h gilt

$$\min\{\mathrm{diam}(B_I), \mathrm{diam}(B_J)) \leq \eta \cdot \mathrm{dist}(B_I, B_J),$$

   so ist $I \times J$ ein Blatt von T.

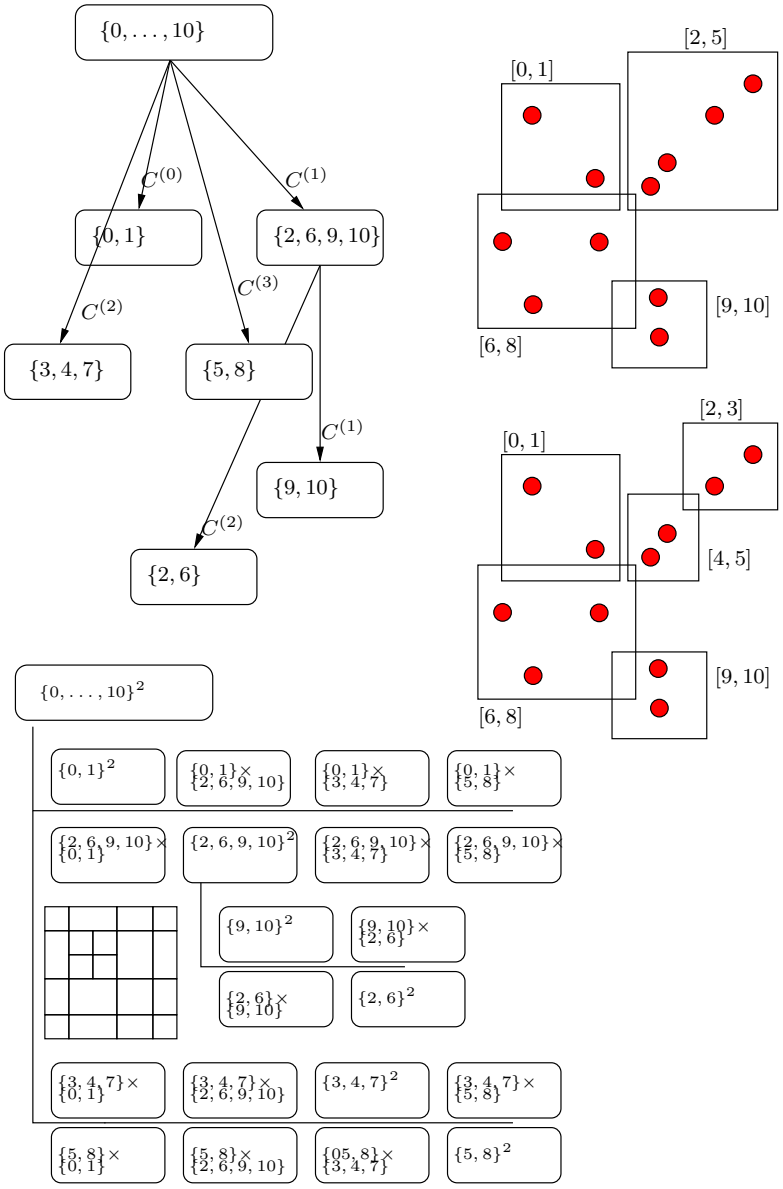
2. Gilt $\#I \leq C_{\max} \vee \#J \leq C_{\max}$, was genau dann gilt, falls $I \in L_{T_\mathcal{I}} \vee J \in L_{T_\mathcal{J}}$, so ist $I \times J$ ein Blatt von T .

3. Ansonsten ist $I \times J$ ein innerer Knoten von T mit $(I \times J)E_T := IE_{T_\mathcal{I}} \times JE_{T_\mathcal{J}}$ und es wird rekursiv mit den Elementen aus $IE_{T_\mathcal{I}} \times JE_{T_\mathcal{J}}$verfahren.

Algorithmus 6: Konstruktion des Blockclusterbaums

[5]Dier hier getroffene Einschränkung, dass jeweils nur Knoten gleicher Höhe beider $\mathcal{H}$-Bäume zu einem Knoten des $\mathcal{H}_\times$-Baums zusammengefasst werden, erleichtert die anschließenden Komplexitätsbetrachtungen.

Abbildung 4.2: Blockclusterbaum T



Man erhält wegen dem zweiten Schritt sofort, dass $h_T \leq \min\{h_{T_I}, h_{T_J}\}$ gilt. Für die Übergabe von I und J kann entsprechend dem letztem Abschnitt verfahren werden. Die Berechnung von $\min\{\text{diam}(B_I), \text{diam}(B_J))$ kann offensichtlich in Zeit $O(d)$ stattfinden. Die Berechnung von $\text{dist}(B_I, B_J)$ lässt sich mit der in Definition 4.1 eingeführten Notation

$$\text{dist}(B_I, B_J) = \sqrt{\sum_{i=1}^{d} \text{dist}(\left[a^{(i)}, b^{(i)}\right], \left[c^{(i)}, d^{(i)}\right])^2}$$

schreiben, und damit ebenfalls durch $O(d)$ abschätzen. Die Laufzeit von Algorithmus 6 hängt somit nur von der Anzahl der Knoten von T ab. Man definiert hierfür:

Definition 4.8. Der aus $T_\mathcal{I}$ und $T_\mathcal{J}$ gebildete $\mathcal{H}_\times$-Baum T heißt schwachbesetzt zur Konstanten C_{sp}, falls

$$\begin{aligned}&\forall h \leq h_{T_\mathcal{I}} \forall I \in V_{T_\mathcal{I},h} \left|\left\{J \in V_{T_\mathcal{J},h} \mid I \times J \in V_{T,h}\right\}\right| \leq C_{sp}\\ &\wedge\\ &\forall h \leq h_{T_\mathcal{J}} \forall J \in V_{T_\mathcal{J},h} \left|\{I \in V_{T_\mathcal{I},h} \mid I \times J \in V_{T,h}\}\right| \leq C_{sp}\end{aligned}$$

gilt. Das heisst, die Anzahl der Knoten auf der Höhenstufe h von T lässt sich durch C_{sp} und die Anzahl der Knoten auf Höhe h in $T_\mathcal{I}$ und $T_\mathcal{J}$ abschätzen.

Nach Algorithmus 6 befinden sich $I \times J$, I und J auf jeweils der selben Höhe in den jeweiligen Bäumen T, $T_\mathcal{I}$ und $T_\mathcal{J}$. Damit besitzt T auf Höhe h höchstens $\#V_{T,h} \leq C_{sp} \cdot \max\{\#V_{T_{\mathcal{I},h}}, \#V_{T_{\mathcal{J},h}})$ Knoten, und somit insgesamt höchstens

$$\#V_T \leq C_{sp} \cdot \sum_{h=0}^{\min\{h_{T_\mathcal{I}}, h_{T_\mathcal{J}}\}} \underbrace{\max\{\#V_{T_{\mathcal{I},h}}, \#V_{T_{\mathcal{J},h}})}_{\leq \#V_{T_{\mathcal{I},h}} + \#V_{T_{\mathcal{J},h}}} \leq C_{sp} \cdot (\#V_{T_\mathcal{I}} + \#V_{T_\mathcal{J}}) \leq 2 \cdot C_{sp} \cdot (\#\mathcal{I} + \#\mathcal{J})$$

Knoten. Damit ergibt sich:

Lemma 4.2. *Die Konstruktion von T aus $T_\mathcal{I}$ und $T_\mathcal{J}$ benötigt die Zeit und den Speicher $O(C_{sp} \cdot (\#I + \#J))$. Der Baum T besitzt höchstens $\#V_T \leq 2 \cdot C_{sp} \cdot (\#I + \#J)$ viele Knoten.*

Es verbleibt C_{sp} durch die gegebenen Punkte $M_\mathcal{I}$ und Boundingboxen $\{B_i \mid i \in \mathcal{I}\}$ abzuschätzen.

Abschätzung von C_{sp}

Entsprechend [5, 6] wird nur der Fall $\mathcal{I} = \mathcal{J}$ betrachtet. Man betrachte die Bedingung in Definition 4.8. Damit $I \times J$ ein Knoten auf Höhe h in T sein kann, muss ein Vaterknoten $E_T(I \times J) = E_{T_\mathcal{I}} I \times E_{T_\mathcal{I}} J$ auf Höhe $h-1$ existieren, der nach Algorithmus 6 somit nicht zulässig ist. Damit lassen sich die Knoten auf Höhe h in T durch die Anzahl der inneren (und damit nicht zulässigen) Knoten auf Höhe $h-1$ in T mal der maximalen Anzahl von Söhnen, also 2^{2d} abschätzen:

$$\#V_{T,h} \leq 2^{2d} \cdot \left|\{I \times J \in V_{T,h-1} \setminus L_T\}\right|.$$

Es folgt daher, dass für die Bedingung aus Definition 4.8

$$\begin{aligned}&\forall h < h_{T_\mathcal{I}} \forall I \in V_{T_\mathcal{I},h} :\\ &\max\left\{\left|\{J \in V_{T_\mathcal{I},h} \mid I \times J \in V_{T,h} \setminus L_T\}\right|, \left|\{I \in V_{T_\mathcal{I},h} \mid I \times J \in V_{T,h} \setminus L_T\}\right|\right\} \leq \tfrac{C_{sp}}{2^{2d}} (*)\end{aligned}$$

hinreichend ist. Es muss daher die Anzahl der nicht zulässigen Knoten in T auf Höhe h abgeschätzt werden. Sei hierfür $I \in T_{\mathcal{I},h}$ unter der Annahme $\exists J \in T_{\mathcal{I},h} : I \times J \in T_h$. Die Punkte M_I sind nach Algorithmus 5 in einem zu $2^{-h} \cdot C_\mathcal{I}$ kongruenten Quader C_J eingeschlossen. Dieser besitzt die maximale Kantenlänge $2^{-h} \cdot \Lambda_\mathcal{I}$. Damit ist die Boundingbox B_I in einem Quader $C_J + [-\max_{i \in I} \operatorname{diam} B_i, \max_{i \in I} \operatorname{diam} B_i]$ eingeschlossen, da im schlechtesten Fall $m_i \in \delta C_I \cap \delta B_i$ gelten kann.

Definition 4.9. Im Weiteren sei $\overline{\lambda} := \max_{i \in \mathcal{I}} \operatorname{diam} B_i$.

Dann lässt sich B_I in einem Würfel mit der Diagonalen $\sqrt{d} \cdot 2^{-h} \cdot \Lambda_{\mathcal{I}} + \overline{\lambda}$ einschließen, das heisst, es gilt $\text{diam}B_I \leq \sqrt{d} \cdot 2^{-h} \cdot \Lambda_{\mathcal{I}} + \overline{\lambda}$. Entsprechend lässt sich $\text{diam}B_J$ durch den selben Wert $\sqrt{d} \cdot 2^{-h} \cdot \Lambda_{\mathcal{I}} + \overline{\lambda}$ abschätzen. Damit $I \times J$ nicht zulässig ist, muss

$$\min\{\text{diam}(B_I), \text{diam}(B_J)) > \eta \cdot \text{dist}(B_I, B_J)$$

gelten. Dabei lässt sich $\text{dist}(B_I, B_J)$ nach obiger Betrachtung mittels C_I und C_J durch

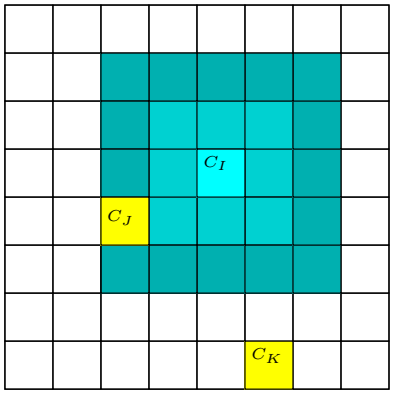
$$\text{dist}(B_I, B_J) \geq \text{dist}(C_I, C_J) - 2 \cdot \overline{\lambda}$$

abschätzen. Die Anzahl der nicht zulässigen Knoten nimmt daher nur zu, falls die Zulässigkeitsbedingung durch

$$\sqrt{d} \cdot 2^{-h} \cdot \Lambda_{\mathcal{I}} + \overline{\lambda} > \eta \cdot \text{dist}(C_I, C_J) - 2 \cdot \overline{\lambda} \Leftrightarrow \text{dist}(C_I, C_J) < \frac{\sqrt{d}}{2^h \cdot \eta} \Lambda_{\mathcal{I}} + \frac{\overline{\lambda}}{\eta}$$

ersetzt wird.
Es muss daher die Anzahl der Quader C_J abgeschätzt werden, welche von C_I einen Abstand $< \frac{\sqrt{d}}{2^h \cdot \eta} \Lambda_{\mathcal{I}} + \frac{\overline{\lambda}}{\eta}$ besitzen. Nach Konstruktion sind die C_J's Teil einer räumlichen Partition Π_h von $C_{\mathcal{I}}$ in zu $2^{-h} \cdot C_{\mathcal{I}}$ kongruente Quader. Dann lässt sich die k-Nachbarschaft von I als die Quader dieser Partition Π_h erklären, welche sich in einem Achsen parallelen Würfel um C_I befinden, auf dessen Kanten genau $2 \cdot k + 1$ Quader der Partition Π_h liegen:

Abbildung 4.3: k-Nachbarschaft von C_I



Das Gatter zeigt die Partition Π_h. Die 2-Nachbarschaft von C_I ist durch einen d-dimensionalen Würfel gegeben, auf dessen Kanten jeweils $1 + 2 \cdot 2$ Quader der Partition Π_h liegen. C_J und C_K sind weitere Knoten von $T_{\mathcal{I}}$ auf der Höhenstufe h.

In der k-Nachbarschaft von C_I befinden sich daher höchstens $(1+2 \cdot k)^d$ Quader der Partition Π_h, welche maximal den Abstand $(k-1) \cdot \sqrt{d} \cdot 2^{-h} \cdot \Lambda_{\mathcal{I}}$ besitzen. Alle Knoten mit Abstand $< \frac{\sqrt{d}}{2^h \cdot \eta} \Lambda_{\mathcal{I}} + \frac{\overline{\lambda}}{\eta}$ müssen sich daher in der $\left\lceil 1 + \frac{\frac{\sqrt{d}}{2^h \cdot \eta} \Lambda_{\mathcal{I}} + \frac{\overline{\lambda}}{\eta}}{\sqrt{d} \cdot 2^{-h} \cdot \Lambda_{\mathcal{I}}} \right\rceil$-Nachbarschaft von I befinden. Mit

$$\left\lceil 1 + \frac{\frac{\sqrt{d}}{2^h \cdot \eta} \Lambda_{\mathcal{I}} + \frac{\overline{\lambda}}{\eta}}{\sqrt{d} \cdot 2^{-h} \cdot \Lambda_{\mathcal{I}}} \right\rceil = \left\lceil 1 + \eta^{-1} (1 + \frac{2^h \cdot \overline{\lambda}}{\sqrt{d} \cdot \Lambda_{\mathcal{I}}}) \right\rceil$$

bedeutet dies, dass I zu höchstens $(1 + 2 \cdot \left\lceil 1 + \eta^{-1}(1 + \frac{2^h \cdot \overline{\lambda}}{\sqrt{d} \cdot \Lambda_{\mathcal{I}}}) \right\rceil)^d$ Knoten aus $T_{\mathcal{I},h}$ nicht zulässig sein kann. Mit $(*)$ ergibt sich somit:

Lemma 4.3. *Die Konstante C_{sp} ist durch*

$$2^{2d} \cdot (1 + 2 \cdot \left\lceil 1 + \eta^{-1}(1 + 2^{h_{T_{\mathcal{I}}}-1} \frac{\cdot \overline{\lambda}}{\sqrt{d} \cdot \Lambda_{\mathcal{I}}}) \right\rceil)^d$$

nach oben beschränkt.

4.2.3 Benötigter Speicher einer $\mathcal{H}$-Matrix

Sei $M \in \mathcal{H}(k, T)$ eine $\mathcal{H}$-Matrix bezüglich dem $\mathcal{H}_\times$-Baum T über $I \times I$. Der benötigte Speicherplatz lässt sich nun leicht mit den Ergebnissen der letzten Abschnitte angeben. Der Speicherplatz zur Haltung der Baumstruktur ist dabei bereits durch $\#V_T \leq 4 \cdot C_{sp} \cdot \#\mathcal{I}$ abgeschätzt. Es bleibt der benötigte Speicher in Blättern von T für die eigentlichen Matrizen abzuschätzen. In den zulässigen Blättern $I \times J \in L_T^Z$ von T wird dabei der Platz $k \cdot (\#I + \#J)$ benötigt, während in den nicht zulässigen Blättern der Speicherbedarf $\#I \cdot \#J \leq \max\{\#I, \#J\} \cdot C_{\max} \leq C_{\max} \cdot (\#I + \#J)$ beträgt, da hier $\min\{\#I, \#J\} \leq C_{\max}$ gilt. Damit erhält man als obere Schranke für den benötigten Speicher:

$$\begin{aligned}
&\textstyle\sum_{I \times J \in L_T^Z} k \cdot (\#I + \#J) + \sum_{I \times J \in L_T^N} \#I \cdot \#J \\
&\leq \max\{C_{\max}, k\} \cdot \textstyle\sum_{I \times J \in L_T} (\#I + \#J) \\
&\leq \max\{C_{\max}, k\} \cdot \textstyle\sum_{h=0}^{h_T} \sum_{I \times J \in L_{T,h}} (\#I + \#J)
\end{aligned}$$

Auf jeder Höhenstufe von T lässt sich für gegebens $I \in T_{\mathcal{I},h}$die Anzahl der Knoten $I \times J \in T_h$ nach Definition 4.8 durch $C_{sp} \cdot \#I$ abschätzen, so dass sich

$$\sum_{I \times J \in L_{T,h}} (\#I + \#J) \leq 2 \cdot C_{sp} \sum_{I \in T_{\mathcal{I},h}} \#I$$

ergibt. Nach Konstruktion von $T_{\mathcal{I}}$ folgt weiterhin $\sum_{I \in T_{\mathcal{I},h}} \#I \leq \#\mathcal{I}$. Hieraus folgt weiter:

$$\begin{aligned}
&\max\{C_{\max}, k\} \cdot \textstyle\sum_{h=0}^{h_T} \sum_{I \times J \in L_{T,h}} (\#I + \#J) \\
&\leq 2 \cdot C_{sp} \cdot \max\{C_{\max}, k\} \cdot \textstyle\sum_{h=0}^{h_T} \#\mathcal{I} \\
&\leq 2 \cdot C_{sp} \cdot \max\{C_{\max}, k\} \cdot (h_{T_{\mathcal{I}}} + 1) \cdot \#\mathcal{I}
\end{aligned}$$

Dabei wurde im letzten Schritt verwendet, dass $h_T \leq h_{T_{\mathcal{I}}}$ nach Konstruktion von T gilt.

Lemma 4.4. *Eine $\mathcal{H}$-Matrix aus $\mathcal{H}(k, T)$, wobei T aus $T_{\mathcal{I}}$ gebildet wurde, benötigt höchstens den Speicher*

$$W_{\mathcal{H},k,C_{\max},\mathcal{I},\eta} := 2 \cdot C_{sp} \cdot \max\{C_{\max}, k\} \cdot (h_{T_{\mathcal{I}}} + 1) \cdot \#\mathcal{I}$$

Es sei festgehalten:

Lemma 4.5.

$$\sum_{I \times J \in V_T} (\#I + \#J) \leq 2 \cdot C_{sp} \cdot (h_{T_{\mathcal{I}}} + 1) \cdot \#\mathcal{I}$$

4.2.4 Aufwand der Matrix-Vektor-Multiplikation

Als nächstes soll die Matrix-Vektor-Multiplikation für eine Matrix aus $M \in \mathcal{H}(k,T)$ betrachtet werden, wobei wiederum T ein $\mathcal{H}_\times$-Baum über $I \times I$ ist, welcher aus $T_\mathcal{I}$ gebildet wurde. Die Zeit für den rekursiven Abstieg ist wiederum durch die Anzahl der Knoten von T bereits abgeschätzt, es verbleibt der Aufwand in den Blättern. Ziel ist es, $w = Mv$ zu berechnen. Ohne Einschränkung kann davon ausgegangen werden, dass der Zielvektor w zu 0 initialisiert ist. Man betrachte daher die allgemeinere Operation $w \leftarrow w + Mv$, welche in den Blättern die nötige zeilenweise Addition Ergebnisse aus der restlichen Blättern beachtet.
In einem zulässigen Blatt $I \times J$ von T lässt sich die Operation als $w\,|_I \leftarrow w\,|_I + AB^\top v\,|_J$ schreiben: Für $v' = B^\top v\,|_J$ werden hierbei k Skalarprodukte von $\#J$-dimensionalen Vektoren berechnet, wofür $k \cdot \#J$ Multiplikationen und $k \cdot \#J - \#J$ Additionen benötigt werden. Entsprechend werden für $w\,|_I = w\,|_I + A \cdot v'$ $\#I \cdot k$ Multiplikationen und Additionen berechnet. Es ergibt sich somit ein Aufwand von $2 \cdot k \cdot (\#I + \#J) - \#J$.
Entsprechend benötigt man in einem nicht zulässigen Blatt $I \times J$ für $w\,|_I \leftarrow w\,|_I + M\,|_{I\times J} \cdot v\,|_J$ $\#I \cdot \#J$ Multiplikationen und $\#I \cdot \#J - \#I$ Additionen. Der Aufwand in den Blättern ist daher durch

$$\sum_{I\times J \in L_T^Z} [2 \cdot k \cdot (\#I + \#J) - \#J] + \sum_{I\times J \in L_T^N} [2 \cdot \#I \cdot \#J - \#J]$$

gegeben, welcher sich daher wieder durch $W_{\mathcal{H},k,C_{\max},\mathcal{I},\eta}$ ausdrücken lässt.

Lemma 4.6. *Der Aufwand der Operation $w \leftarrow w + M \cdot v$ für eine $\mathcal{H}$-Matrix ist durch $2 \cdot W_{\mathcal{H},k,C_{\max},\mathcal{I},\eta} =: W^{w+\mathcal{H}(k,T)\cdot v}_{\mathcal{H},k,C_{\max},\mathcal{I},\eta}$ beschränkt.*

4.2.5 Addition zweier $\mathcal{H}$-Matrizen

Es gelte $M, N \in \mathcal{H}(k,T)$. Zu berechnen ist $M + N$. Da nach Annahme M und N beide über T definiert sind, folgt, dass die Addition wieder auf die Blätter von T zurückzuführen ist. In den nicht zulässigen Blättern $I \times J$ werden hierfür genau $\#I \cdot \#J$ Additionen durchgeführt. Ist jedoch $I \times J$ ein zulässiges Blatt, so gilt bekanntlich, dass der Rang der resultierenden Matrix wegen $A_M B_M^\top + A_N B_N^\top = \begin{bmatrix} A_M & A_N \end{bmatrix} \cdot \begin{bmatrix} B_M & B_N \end{bmatrix}^\top$ nur noch durch $2 \cdot k$ beschränkt ist. Die exakte Addition $+ : \mathcal{H}(k,T) \times \mathcal{H}(k,T) \to \mathcal{H}(2k,T)$ benötigt daher in den zulässigen Blättern den Aufwand $2k \cdot (\#I + \#J)$ für das Kopieren der Einträge von A_M, B_M, A_N, B_N. Damit entspricht der Aufwand in den Blättern genau den Abschätzungen des Speicherbedarfs, wenn von dem zusätzlichen Faktor 2 abgesehen wird. Dies führt auf:

Lemma 4.7. *Die exakte Addition $+ : \mathcal{H}(k,T) \times \mathcal{H}(k,T) \to \mathcal{H}(2k,T)$ lässt sich in Zeit $2W_{\mathcal{H},k,C_{\max},\mathcal{I},\eta} =: W^{\mathcal{H}(k,T)+\mathcal{H}(k,T)}_{\mathcal{H},k,C_{\max},\mathcal{I},\eta}$ durchführen.*

Im Fall der formatierten Addition $\oplus : \mathcal{H}(k,T) \times \mathcal{H}(k,T) \to \mathcal{H}(k,T)$ fällt jedoch noch zusätzlich in den zulässigen Blättern die Rangreduktion an. Diese soll nun zunächst abgeschätzt werden:

Rangreduktion einer $\mathcal{R}_{\tilde{k}}$ Matrix auf Rang k

Gegeben seien die Matrizen $A \in \mathbb{R}^{m\times\tilde{k}}$ und $B \in \mathbb{R}^{\tilde{k}\times n}$. Die Rangreduktion verläuft wie folgt:

1. QR-Zerlegung von $A = Q_A R_A$ und $B = Q_B R_B$ in Zeit $O(m \cdot \tilde{k}^2)$ bzw. $O(n \cdot \tilde{k}^2)$
2. Berechnung von $M = R_A R_B^\top$ in Zeit $O(\tilde{k}^3)$
3. Singulärwert-Zerlegung von $M = U\Sigma V^\top$ in Zeit $O(\tilde{k}^3)$
4. Kürzen von Σ auf Rang k in $O(k) = O(\tilde{k})$
5. Multiplikation von $Q_A U\Sigma$ und $Q_B V$ in Zeit $O(k \cdot \tilde{k} \cdot (m+n)) = O(\tilde{k}^2 \cdot (m+n))$

Algorithmus 7: Rangreduktion einer $\mathcal{R}_k$-Matrix

Damit ergibt sich ein Gesamtaufwand von $O(\tilde{k}^2 \cdot (m+n) + \tilde{k}^3)$. Auf Seite 20 in [6] wird der Aufwand genauer durch $5 \cdot (m+n)\tilde{k}^2 + 23 \cdot \tilde{k}^3$ abgeschätzt, abweichend hiervon wird jedoch in [5] der Aufwand mit $6 \cdot (m+n)\tilde{k}^2 + 23 \cdot \tilde{k}^3$ angegeben. Dieses Vorgehen ist natürlich nur in dem Fall $\tilde{k} < \min\{m, n\}$ effizient, wofür $\tilde{k} \leq C_{\max}$ hinreichend ist.
Ansonsten wird $A \cdot B$ explizit berechnet und die Rangreduktion mittels der vollbesetzten Matrix F durchgeführt ($\rightsquigarrow O(\tilde{k} \cdot m \cdot n)$), wobei unter 1. entweder $F = QR$ oder $F = RQ^\top$ berechnet wird , je nachdem, ob $m \geq n$ oder $m < n$ ($\rightsquigarrow O(\max\{m,n\} \cdot \min\{m,n\}^2)$. Der zweite Schritt kann dann entfallen, und im dritten tritt R an die Stelle von M ($\rightsquigarrow O(\min\{m,n\}^3)$). Schließlich muss im 5. Schritt nur eine der beiden Multiplikationen mit Q ausgeführt werden ($\rightsquigarrow O(k \cdot \min\{m,n\} \cdot \max\{m,n\})$). Insgesamt ergibt sich daher mit $k \leq \tilde{k}$ und $\tilde{k} \geq \min\{m,n\}$ ein Aufwand von $O(\tilde{k} \cdot \min\{m,n\} \cdot \max\{m,n\}) = O(\tilde{k} \cdot m \cdot n)$.
In [6] wird für den Aufwand der Konvertierung einer $m \times n$ Matrix mit $11(m^3+n^3)$ angegeben.

Lemma 4.8. *Die Rangreduktion einer $\mathcal{R}_k$-Matrix vom Rang $\tilde{k}$ benötigt*

$$\min\{6 \cdot (m+n)\tilde{k}^2 + 23 \cdot \tilde{k}^3, 11(m^3+n^3) + 2 \cdot n \cdot m \cdot \tilde{k}\}$$

Zeit. Die Konvertierung einer vollbesetzten Matrix $M \in \mathbb{R}^{m \times n}$ in das $\mathcal{R}_k$-Format mit Rang $\tilde{k}$ hat dabei den Aufwand

$$11 \cdot (m^3 + n^3)$$

Bemerkung 4.2. Entsprechend [6, S. 19] wird im weiteren davon ausgegangen, dass stets für den Rang k der $\mathcal{R}_k$-Matrizen $k \ll m, n$, das heisst, $k \ll C_{\max}$ gilt.

Man nehme an, dass k geeignet gewählt wurde, so dass die Abschätzung $6 \cdot (m+n)\tilde{k}^2 + 23 \cdot \tilde{k}^3$ verwendet werden kann. Dann folgt für die Rangreduktion durch Abschätzung des Aufwands in den zulässigen Blättern von T:

Lemma 4.9. *Die Konvertierung einer Matrix $M \in \mathcal{H}(\tilde{k}, T)$ nach $\mathcal{H}(k, T)$ für $k < \tilde{k}$ ist durch*

$$\begin{aligned} & \textstyle\sum_{I \times J \in L_T^Z} \left[6 \cdot (\#I + \#J)\tilde{k}^2 + 23 \cdot \tilde{k}^3\right] \\ \leq\ & 23 \cdot \tilde{k}^3 \cdot \#L_T^Z + \tilde{k} \cdot \textstyle\sum_{I \times J \in L_T^Z} 6 \cdot (\#I + \#J)\tilde{k} \\ \leq\ & 6 \cdot \tilde{k} \cdot W_{\mathcal{H},k,C_{\max},\mathcal{I},\eta} + 23 \cdot \tilde{k}^3 \cdot \#L_T^Z \\ =:\ & W_{\mathcal{H},k,C_{\max},\mathcal{I},\eta}^{\mathcal{H}(\tilde{k},T) \to \mathcal{H}(k,T)} \end{aligned}$$

beschränkt.

4.2.6 Aufwand der formatierten Addition $\oplus : \mathcal{H}(k,T) \times \mathcal{H}(k,T) \to \mathcal{H}(k,T)$

Man nehme an, dass k geeignet gewählt wurde, so dass die Abschätzung $5 \cdot (m+n)\tilde{k}^2 + \tilde{k}^3$ verwendet werden kann. Mit $\tilde{k} = 2 \cdot k$ ergibt sich somit für den Aufwand in Blättern von T:

Lemma 4.10. *Der Aufwand der formatierten Addition $\oplus : \mathcal{H}(k,T) \times \mathcal{H}(k,T) \to \mathcal{H}(k,T)$ lässt sich durch*

$$\begin{aligned}
& \textstyle\sum_{I\times J\in L_T^Z} \left[2\cdot k\cdot(\#I+\#J) + 4\cdot 6\cdot(\#I+\#J)k^2 + 8\cdot 23\cdot k^3\right] + \sum_{I\times J\in L_T^N} \#I\cdot\#J \\
\leq\ & 26\cdot k\cdot\left[\textstyle\sum_{I\times J\in L_T^Z} k\cdot(\#I+\#J) + \sum_{I\times J\in L_T^N} \#I\cdot\#J\right] + 184\cdot k^3\cdot \#L_T^Z \\
\leq\ & 26\cdot k\cdot W_{\mathcal{H},k,C_{\max},\mathcal{I},\eta} + 184\cdot k^3\cdot \#L_T^Z \\
=:\ & W_{\mathcal{H},k,C_{\max},\mathcal{I},\eta}^{\mathcal{H}(k,T)\oplus\mathcal{H}(k,T)}
\end{aligned}$$

abschätzen.

4.2.7 Multiplikation zweier $\mathcal{H}$-Matrizen

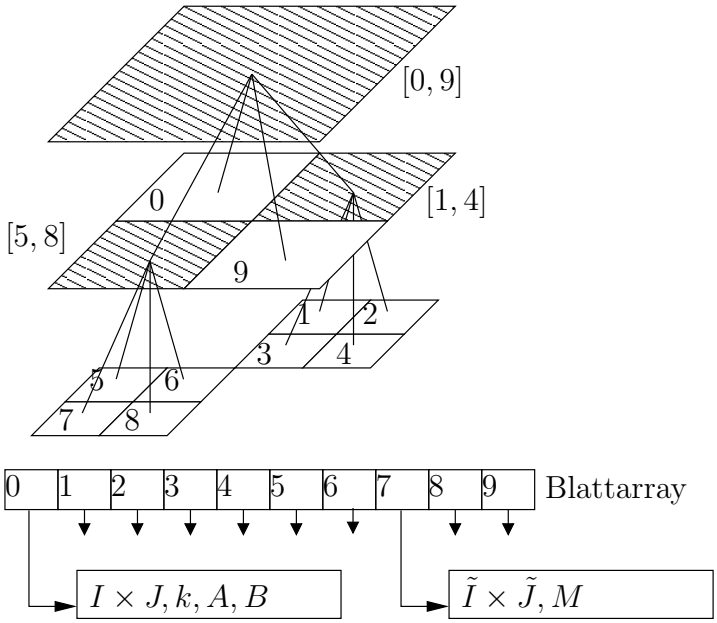
Die Komplexitätsanalyse des Matrixprodukts $M \cdot N$ zweier $\mathcal{H}$-Matrizen $M \in \mathcal{H}(k,T)$ und $N \in \mathcal{H}(\tilde{k},\widetilde{T})$ wird unter der Einschränkung durchgeführt, dass die beiden $\mathcal{H}_\times$-Bäume T und $\widetilde{T}$ kompatibel sind: Es wird gefordert, dass T aus $T_\mathcal{I}$ und $T_\mathcal{J}$ gebildet ist, und $\widetilde{T}$ aus $T_\mathcal{J}$ und $T_\mathcal{K}$, wobei ohne Einschränkung alle drei $\mathcal{H}$-Bäume $T_\mathcal{I}$, $T_\mathcal{J}$ und $T_\mathcal{K}$ bezüglich dem Parameter $C_{\max}$ konstruiert wurden. Weiterhin wird entsprechend dem Fall der Matrix-Vektor-Multiplikation die allgemeinere Operation $L+ = M \cdot N$ betrachtet, wobei angenommen wird, dass $\hat{T}$ aus $T_\mathcal{I}$ und $T_\mathcal{K}$ gebildet ist und $L \in \mathcal{H}(\hat{k},\hat{T})$ gilt.
Es liegt nahe, $L = L + M \cdot N$ wieder rekursiv auszuwerten, und so die eigentlichen Berechnungen auf den Fall zurückzuführen, dass mindestens eine der beteiligten Teilmatrizen in expliziter Form vorliegt, das heisst, einem Blatt in dem jeweiligen Baum entspricht. Dies lässt sich wie folgt formulieren, wobei zu Beginn wieder $I = \mathcal{I}$, $J = \mathcal{J}$ und $K = \mathcal{K}$ gelte:

1. Es gelte $I \times J \in V_T \setminus L_T \wedge J \times K \in V_{\widetilde{T}} \setminus L_{\widetilde{T}}$. Wegen der Konstruktion von T und $\widetilde{T}$ gilt $(I \times J)E_T = (IE_{T_\mathcal{I}}) \times (JE_{T_\mathcal{J}})$ und $(J \times K)E_{\widetilde{T}} = (JE_{T_\mathcal{J}}) \times (KE_{T_\mathcal{K}})$ mit $\biguplus_{I' \in IE_{T_\mathcal{I}}} = I$, $\biguplus_{J' \in JE_{T_\mathcal{J}}} = J$ und $\biguplus_{K' \in KE_{T_\mathcal{K}}} = K$, so dass rekursiv bezüglich der Tripel $(IE_{T_\mathcal{I}}) \times (JE_{T_\mathcal{J}}) \times (KE_{T_\mathcal{K}})$ abgestiegen werden kann.
2. Ansonsten folgt $I \times J \in L_T \vee J \times K \in L_{\widetilde{T}}$. Berechne $L\,|_{I\times K} += M\,|_{I\times J} \cdot N\,|_{J\times K}$.

Algorithmus 8: Exaktes Matrixprodukt ohne Beachtung von $\hat{T}$

Als erstes soll das unter 2. anfallende Matrixprodukt betrachtet werden. Hier ist mindestens eine der beiden Matrizen entweder als vollbesetzte Matrix oder als $\mathcal{R}_k$-Matrix gegeben. In letzterem Fall muss dabei nur das Produkt mit einem der beiden Faktoren des $\mathcal{R}_k$-Formats durchgeführt werden, so dass es ausreicht, das Produkt einer vollbesetzten Matrix mit einer im Allgemeinen $\mathcal{H}$-Matrix zu betrachten:
Bei geeigneter Speicherung des Blockclusterbaums T können die Blätter in der Reihenfolge ihrer Entstehung durchnummeriert werden. Jedem inneren Knoten von T können daher seine

Abbildung 4.4: Blockclusterbaum T - Blattnummerierung ($d = 1$)



Blätter zugeordnet werden, indem in ihm der niedrigste und höchste "Entstehungsindex" der von ihm aus erreichbaren Blätter abgespeichert wird. Auf Grund der Rekursion beschreiben diese "Entstehungsindizes" genau die an ihm hängenden Blätter. Vor allem können sie ohne zusätzlichen Aufwand während der Konstruktion von T durch MinMax-Bildung über den Indizes der Söhne gebildet werden.

Die eigentliche $\mathcal{H}$-Matrix kann dann durch einen Zeiger auf den ihr zu Grunde liegenden Blockclusterbaum und einen Array mit Zeigern auf die Blätter/Teilmatrizen realisiert werden. Ist dann die unter 2. beschriebene Situation erreicht, so sei ohne Einschränkung $J \times K \notin L_{\widetilde{T}}$. Dann kann das Matrixprodukt mittels der beiden Indizes von $J \times K$ und dem Blattarray von N direkt durchgeführt werden.

Wie man weiter bemerkt, beachtet dieser Ansatz zunächst nicht die Struktur der Zielmatrix L bzw. $\hat{T}$. Erstes Ziel ist es, aus diesem Ansatz abzuleiten, wie $\hat{T}$, $L^Z_{\hat{T}}$ und $\hat{k}$ geeignet gewählt werden können, so dass sich das Ergebnis exakt abspeichern lässt, das heisst, ohne zusätzliche Konvertierungen von vollbesetzten Matrizen in das $\mathcal{R}_{\hat{k}}$-Format. Ziel ist hierbei natürlich, dass im allgemeinen diese Parameter nicht-trivial gewählt werden, das heisst, der Fall $\hat{T} = (\{\mathcal{I} \times \mathcal{K}\}, \emptyset)$ und $L^Z_{\hat{T}} = \emptyset$ vermieden wird. Dieser spezielle $\mathcal{H}_\times$-Baum wird im weiteren mit $T \cdot \widetilde{T}$ bezeichnet. Der benötigte Rang der $\mathcal{R}_k$-Matrizen in den Blättern $L_{T \cdot \widetilde{T}}$ wird mit $k \odot \tilde{k}$ bezeichnet.

Hierauf aufbauend kann dann die nachträgliche bzw. gleichzeitige Konvertierung im Fall eines allgemeinen $\hat{T}$ betrachtet werden, indem die Analyse sich auf $T \cdot \widetilde{T}$ stützt.

Exaktes Matrixprodukt

Der durch die Rekursion beschriebene Baum $\mathbb{T}$ ist das "drei-dimensionale" Analogon eines $\mathcal{H}_\times$-Baums über $\mathcal{I} \times \mathcal{J} \times \mathcal{K}$ mit der Kantenrelation

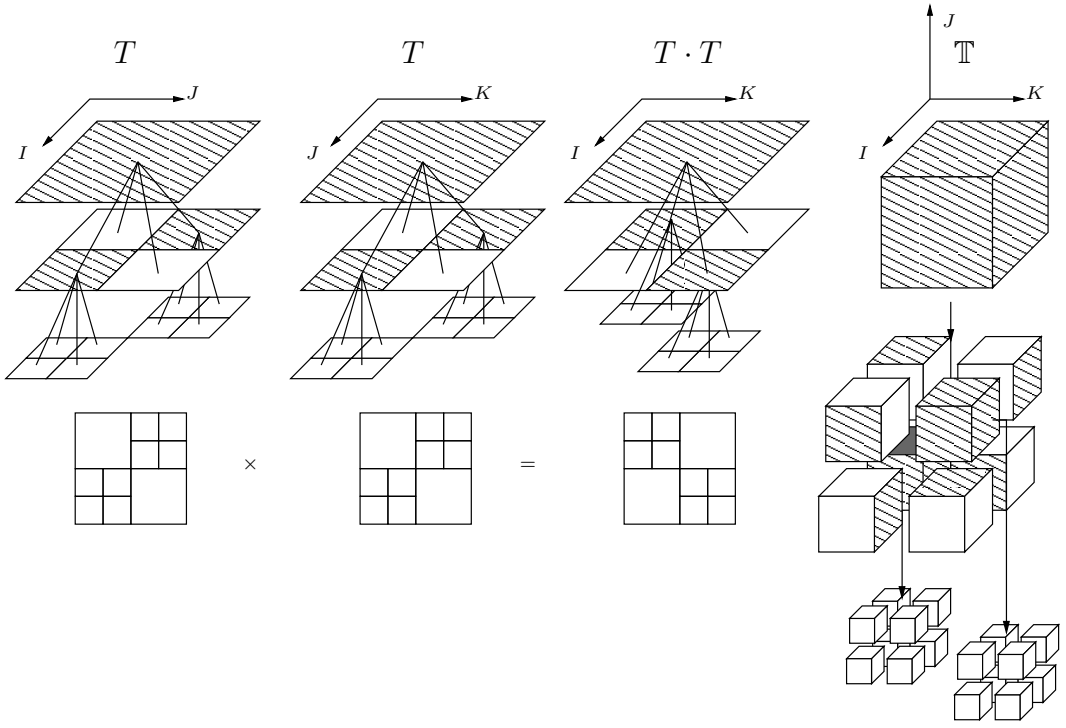
$$(I\times J\times K)E_{\mathbb{T}} := \begin{cases} (IE_{T_\mathcal{I}}) \times (JE_{T_\mathcal{J}}) \times (KE_{T_\mathcal{K}}), & \text{falls } (I \times J) \in V_T \setminus L_T \wedge (J \times K) \in V_{\widetilde{T}} \setminus L_{\widetilde{T}}, \\ \emptyset, & \text{sonst.} \end{cases}$$

Die Knoten von $\mathbb{T}$ sind dann durch $V_{\mathbb{T}} := (\mathcal{I} \times \mathcal{J} \times \mathcal{K})E^*_{\mathbb{T}}$ gegeben. Damit $T \cdot \widetilde{T}$ die geforderte Eigenschaft besitzt, das heisst, dass die Rekursion des obigen Algorithmus nur dann abbricht, falls die erste Bedingung erfüllt ist, kann $I \times K$ nur dann ein Blatt von $\hat{T}$ sein, falls kein $J \in V_{T_\mathcal{J}}$existiert, so dass $(I \times J \times K)E_{\mathbb{T}} \neq \emptyset$. Die existentielle Quantifizierung bezüglich J bzw. $T_\mathcal{J}$ entspricht der Projektion von $\mathbb{T}$ bezüglich der zweiten Komponente, das heisst, das Tripel $I \times J \times K$ wird auf $I \times K$ abgebildet. Dies führt auf

$$(I \times K)E_{T\cdot\widetilde{T}} := \{I' \times K' \mid \exists J, J' \in V_{T_\mathcal{J}} : (I' \times J' \times K') \in (I \times J \times K)E_{\mathbb{T}}\}$$

und $V_{T\cdot\widetilde{T}} := (\mathcal{I} \times \mathcal{K})E^*_{T\cdot\widetilde{T}}$.

Abbildung 4.5: Traversierungsbaum $\mathbb{T}$ und Produktbaum $T \cdot T$



Weiterhin folgt sofort $h_{T\cdot\widetilde{T}} \leq \min\{h_T, h_{\widetilde{T}}\}$. Somit ist die Baumstruktur von L definiert, jedoch muss noch erklärt werden, in welchen Blättern von $T \cdot \widetilde{T}$ Teilmatrizen von L im $\mathcal{R}_k$-Format gespeichert werden:
Wie bereits angesprochen, soll hierbei keine zusätzlichen Konvertierungen in das $\mathcal{R}_k$-Format anfallen. Um die "zulässigen" Blätter von $T \cdot \widetilde{T}$ zu bestimmen, das heisst, diejenigen Blätter,

in welchen die zugeordnete Teilmatrix direkt im $\mathcal{R}_k$-Format gespeichert werden kann, stellt man zunächst fest, dass das Matrixprodukt einer $\mathcal{R}_k$-Matrix $R = A \cdot B^\top$ mit einer beliebigen Matrix sich sofort als $\mathcal{R}_k$-Matrix schreiben lässt, indem die Multiplikation nur mit einer der beiden Matrizen A oder B explizit durchgeführt wird.
Entsprechend wurde bereits im Abschnitt über die Addition zweier $\mathcal{H}$-Matrizen bemerkt, dass die Summe zweier $\mathcal{R}_k$-Matrizen durch reines Kopieren der Einträge als $\mathcal{R}_{2k}$-Matrix darstellen lässt.
In Schritt 1 des obigen Algorithmus muss jedoch $I \times K$ kein Blatt von $T \cdot \widetilde{T}$ sein. An dieser Stelle muss somit eine Teilmatrix des Produkts in die an $I \times K$ hängenden Blätter von $T \cdot \widetilde{T}$ übergeben werden. Eine Teilmatrix $R\,|_{I' \times K'}$ einer $\mathcal{R}_k$-Matrix $R = AB^\top$lässt sich wegen $(AB^\top)\,|_{I' \times K'} = A\,|_{I' \times k}\,(B\,|_{J' \times k})^\top$ offensichtlich ebenfalls als $\mathcal{R}_k$-Matrix ohne zusätzliche Konvertierung übergeben.
Man definiert daher die zulässigen Knoten in $T \cdot \widetilde{T}$ als diejenigen, in deren Berechnung einzig Matrixprodukte beteiligt sind, in welchen jeweils mindestens eine $\mathcal{R}_k$-Matrix als Faktor auftritt. Damit hat jedes dieser Produkte maximal den Rang $\max\{k, \tilde{k}\}$. Ansonsten wird die Teilmatrix explizit als vollbesetzte Matrix abgespeichert.

Definition 4.10. Für ein Blatt $(I \times K) \in L_{T \cdot \widetilde{T}}$ definiert man hierfür

$$O_{T \cdot \widetilde{T}}(I \times K) := \left\{ (I^\uparrow \times J^\uparrow \times K^\uparrow) \in L_{\mathbb{T}} \mid (I^\uparrow \times K^\uparrow) \in E^*_{T \cdot \widetilde{T}}(I \times K) \right\}$$

als die Menge aller Blätter des Berechnungsbaums $\mathbb{T}$, in welchen ein Summand zu der Teilmatrix $L\,|_{I \times K}$ berechnet wird. Damit folgt:

$$L\,|_{I \times K} + M\,|_{I \times \mathcal{J}}\, N\,|_{\mathcal{J} \times K} = \sum_{(I^\uparrow \times J^\uparrow \times K^\uparrow) \in O(I \times K)} (L\,|_{I^\uparrow \times K^\uparrow} + M\,|_{I^\uparrow \times J^\uparrow}\, N\,|_{J^\uparrow \times K^\uparrow})\,|_{I \times K}$$

Die Definition der zulässigen Knoten von $T \cdot \widetilde{T}$ lässt sich hiermit wie folgt schreiben:

$$I \times K \in L^Z_{T \cdot \widetilde{T}} :\Leftrightarrow I \times K \in L_{T \cdot \widetilde{T}} \wedge \forall (I^\uparrow \times J^\uparrow \times K^\uparrow) \in O_{T \cdot \widetilde{T}}(I \times K) : (I^\uparrow \times J^\uparrow) \in L^z_T \vee (J^\uparrow \times K^\uparrow) \in L^Z_{\widetilde{T}}$$

Der maximale Rang in einem zulässigen Blatt $I \times K$ ist daher durch die Anzahl der Summanden, das heisst, $\#O_{T \cdot \widetilde{T}}(I \times K)$, und $\max\{k, \tilde{k}\}$ beschränkt:

$$\hat{k} = k \odot \tilde{k} := \max\{k, \tilde{k}\} \cdot \max_{I \times K \in L^Z_{T \cdot \widetilde{T}}} \#O_{T \cdot \widetilde{T}}(I \times K)$$

Damit sind $T \cdot \widetilde{T}$, $L^Z_{T \cdot \widetilde{T}}$ und $k \odot \tilde{k}$ beschrieben. Die Konstante $C_{sp, T \cdot \widetilde{T}}$ lässt dabei durch $C_{sp,T}$ und $C_{sp,\widetilde{T}}$ leicht abschätzen. Sei $h \le h_{T \cdot \widetilde{T}}$ und $I \in V_{T_{\mathcal{I}}, h}$, dann gilt:

$$\begin{aligned} \left| \left\{ K \in V_{\mathcal{K},h} \mid I \times K \in V_{T \cdot \widetilde{T}, h} \right\} \right| &= \left| \left\{ K \in V_{\mathcal{K},h} \mid \exists J \in V_{T_{\mathcal{J}},h} : I \times J \times K \in V_{\mathbb{T},h} \right\} \right| \\ &\le \left| \{ J \in V_{\mathcal{J},h} \mid I \times J \in V_{T,h} \} \times \left\{ K \in V_{\mathcal{K},h} \mid J \times K \in V_{\widetilde{T},h} \right\} \right| \\ &\le C_{sp,T} \cdot C_{sp,\widetilde{T}} \end{aligned}$$

Entsprechendes folgt für festes $K \in V_{\mathcal{K},h}$, so dass folgendes Lemma gilt:

Lemma 4.11. *Es gilt:* $C_{sp, T \cdot \widetilde{T}} \le C_{sp,T} \cdot C_{sp,\widetilde{T}}$.

Als nächstes soll das Vorgehen im Fall $\hat{T} \neq T \cdot \widetilde{T}$ beschrieben werden:

Formatiertes Matrixprodukt

Gilt für die "Zielmatrix" L nicht, dass sie über $T \cdot \widetilde{T}$ und $k \odot \tilde{k}$ definiert ist, so müssen noch zusätzliche Konvertierungsschritte in den Blättern von $L_{\hat{T}}$ durchgeführt werden. Hierbei können drei Fälle unterschieden werden:

1. $I \times K \in L_{\hat{T}} \wedge I \times K \in L_{T \cdot \widetilde{T}}$: Es muss eine der bereits besprochenen Konvertierungen zwischen einer vollbesetzten Matrix und einer $\mathcal{R}_k$-Matrix durchgeführt werden, das heisst, entweder Ausmultiplizieren von $A \cdot B^\top$ oder mittels Singulärwertzerlegung Konvertierung von M nach $A \cdot B^\top$.

2. $I \times K \in L_{\hat{T}} \wedge I \times K \in V_{T \cdot \widetilde{T}} \setminus L_{T \cdot \widetilde{T}}$: Das Blatt $I \times K$ in $\hat{T}$ ist die Wurzel eines Teilbaums $T_{sub}(I \times K)$ von $T \cdot \widetilde{T}$, das heisst, eine hierarchische Matrix über $((I \times K)E^*_{T \cdot \widetilde{T}}, E_{T \cdot \widetilde{T}})$ muss konvertiert werden. Ist $I \times K$ in $\hat{T}$ nicht zulässig, so reicht es, die zulässigen Blätter von $((I \times K)E^*_{T \cdot \widetilde{T}}, E_{T \cdot \widetilde{T}})$ auszumultiplizieren. Ansonsten muss eine hierarchische Matrix in das $\mathcal{R}_{\hat{k}}$-Format konvertiert werden.

3. $I \times K \in L_{\hat{T}} \wedge I \times K \notin V_{T \cdot \widetilde{T}}$: Ein Vorgänger $I^\uparrow \times K^\uparrow$ von $I \times K$ ist somit ein Blatt von $T \cdot \widetilde{T}$ - dies gilt, da $\hat{T}$ und $T \cdot \widetilde{T}$ beides aus $T_{\mathcal{I}}$ und $T_{\mathcal{K}}$ gebildete $\mathcal{H}_\times$-Bäume sind. Die Matrix über $I^\uparrow \times K^\uparrow$ muss daher auf die Blätter $L_{\hat{T}} \cap (I^\uparrow \times K^\uparrow)E^*_{\hat{T}}$ aufgeteilt, und dort notfalls entsprechend 1. konvertiert werden. Dabei lässt sich jede Teilmatrix einer $\mathcal{R}_k$-Matrix - wie bereits bemerkt - ebenfalls wieder als $\mathcal{R}_k$-Matrix ohne zusätzlichen Aufwand darstellen, so dass gegebenenfalls auch die Rangreduktion für $\mathcal{R}_k$-Matrizen verwendet werden kann.

Einzig die unter 2. mögliche Konvertierung eines Teilbaums in das $\mathcal{R}_k$-Format muss noch genauer betrachtet werden. Hierfür wird die "Ähnlichkeit" von $\hat{T}$ zu $T \cdot \widetilde{T}$ durch die maximale Anzahl der Knoten eines der Teilbäume $T_{sub}(I \times K)$ definiert:

Definition 4.11.

$$C_{id,\hat{T},T,\widetilde{T}} := \max_{I \times K \in L_{\hat{T}}} \left| (I \times K)E^*_{T \cdot \widetilde{T}} \right|$$

Nach Definition gilt immer $C_{id} \geq 1$. Für die Konvertierung lassen sich zwei Vorgehensweisen unterscheiden:
Entweder wird zunächst explizit das Ergebnis in $\mathcal{H}(k \odot \tilde{k}, T \cdot \widetilde{T})$ berechnet und nachträglich konvertiert, oder die Konvertierung/Approximation des Teilbaums als $\mathcal{R}_k$-Matrix geschieht parallel zur Berechnung des Produkts.
Es soll zunächst die erste Möglichkeit beschrieben werden:

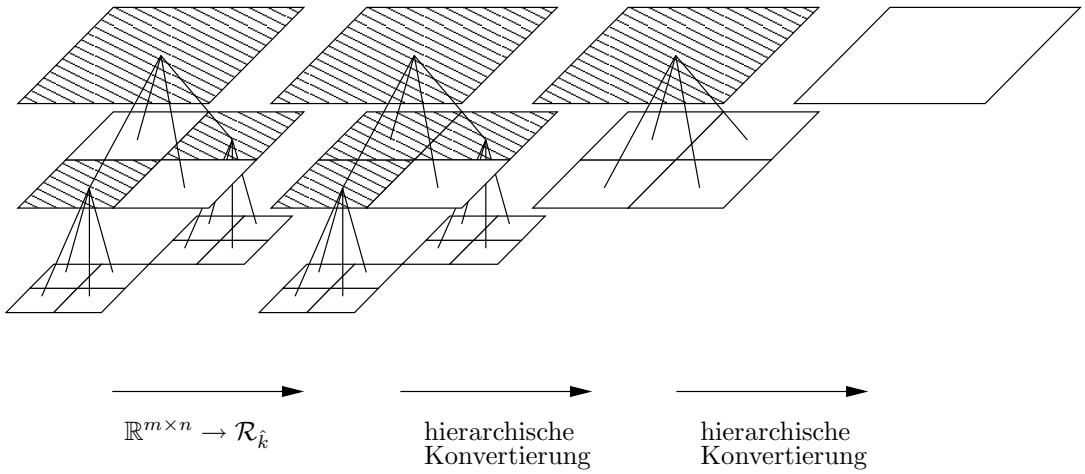
Für die nachträgliche Konvertierung kommen zwei Möglichkeiten in Frage:
In der ersten werden die Blätter der entsprechenden Teilbäume $(I \times K)E^*_{T \cdot \widetilde{T}}$ zunächst als (jeweils) eine $\mathcal{R}_k$-Matrix geschrieben, deren Rang dann wieder auf $\hat{k}$ reduziert wird, vgl. Lemma 4.8. Liegen einige der Teilmatrizen dabei als vollbesetzte Matrizen vor, so werden diese *kanonisch* als $\mathcal{R}_k$-Matrizen dargestellt: Sei $F \in \mathbb{R}^{m \times n}$ eine vollbestetzte Matrix. Ohne Einschränkung gelte $m \leq n$. Dann ist offensichtlich durch $A = Id \in \mathbb{R}^{m \times m}$ und $B = F^\top$ eine $\mathcal{R}_k$-Matrix vom Rang m gegeben, wobei in diesem Fall noch $O(m \cdot n)$ Operationen für das Transponieren von F anfallen. Entsprechend verfährt man im Fall $m > n$, wobei hier das Transponieren entfällt. Es liegt dann eine Blockmatrix vor, wobei jeder Block als $\mathcal{R}_k$-Matrix

gegeben ist. Diese Matrix soll zu einer $\mathcal{R}_k$-Matrix addiert werden. Dies lässt sich im Fall $d = 1$ leicht veranschaulichen:

$$\begin{aligned}
&\underbrace{AB^\top}_{m\times n} + \underbrace{\begin{pmatrix} A_{00}B_{00}^\top & A_{01}B_{01}^\top \\ A_{10}B_{10}^\top & A_{11}B_{11}^\top \end{pmatrix}}_{m\times n} \\
= \;& \underbrace{A}_{m\times \hat{k}}\underbrace{B^\top}_{\hat{k}\times n} + \underbrace{\begin{pmatrix} A_{00} \\ 0 \end{pmatrix}}_{m\times k_{00}}\underbrace{\begin{pmatrix} B_{00} \\ 0 \end{pmatrix}^\top}_{k_{00}\times n} + \underbrace{\begin{pmatrix} A_{01} \\ 0 \end{pmatrix}}_{m\times k_{01}}\underbrace{\begin{pmatrix} 0 \\ B_{01} \end{pmatrix}^\top}_{k_{01}\times n} + \dots \\
= \;& \underbrace{\begin{bmatrix} A & A_{00} & A_{01} & 0 & 0 \\ & 0 & 0 & A_{10} & A_{11} \end{bmatrix}}_{m\times(\hat{k}+k_{00}+\dots k_{11})} \underbrace{\begin{bmatrix} B & B_{00} & 0 & B_{10} & 0 \\ & 0 & B_{01} & 0 & B_{11} \end{bmatrix}^\top}_{(\hat{k}+k_{00}+\dots k_{11})\times n}.
\end{aligned}$$

Das Ergebnis kann somit nur mittels Kopieren der Teilmatrizen wieder als $\mathcal{R}_k$-Matrix entsprechend der formatierten Addition geschrieben werden. Auf diese Matrix wird dann die Rangreduktion angewendet. Hierbei hängt der Rang der entstehenden Matrix explizit von der Anzahl der Blätter des Teilbaums ab, welcher konvertiert werden soll.
Die zweite Vorgehensweise besteht darin, die Approximationen hierarchisch zu berechnen. Das heisst, es werden zunächst die Blätter explizit, insbesondere im Fall von vollbesetzten Matrizen auf Rang $\hat{k}$ reduziert. Rekursiv wird dann obiges Schema in jedem inneren Knoten des Baums $(I \times K)E^*_{T\cdot\widetilde{T}}$ bezüglich der direkten Nachfolger durchgeführt. Hierdurch hängt der Rang nur noch von der maximalen Anzahl von Nachfolgern in $T \cdot \widetilde{T}$ ab, das heisst, ist durch $2^{2\cdot d} \cdot \hat{k}$ begrenzt:

Abbildung 4.6: Hierarchische Konvertierung von $T \cdot T$



Findet die Approximation der Teilbäume während der Berechnung statt, so kann dies wie folgt geschehen:

Sei hierfür mit $\mathcal{T}_{\hat{k}}$ die Rangreduktion einer $\mathcal{R}_k$-Matrix auf Rang $\hat{k}$ bezeichnet. Wird dann während der Traversierung von $\mathbb{T}$ der Knoten $I \times J \times K$ erreicht mit $I \times K \in L^Z_{\hat{T}}$, so soll rekursiv

$$\mathcal{T}_{\hat{k}}(L\,|_{I\times K} + M\,|_{I\times J}\; N\,|_{J\times K})$$

berechnet werden. Dies lässt sich schreiben als (zur Vereinfachung sei $d = 1$):

$$\begin{aligned} & \mathcal{T}_{\hat{k}}\left(L_A L_B^\top + \begin{pmatrix} M_{00} & M_{01} \\ M_{10} & M_{11} \end{pmatrix}\begin{pmatrix} L_{00} & L_{01} \\ L_{10} & L_{11} \end{pmatrix}\right) \\ = \quad & \mathcal{T}_{\hat{k}}\left(L_A L_B^\top + \begin{pmatrix} M_{00}L_{00} + M_{01}L_{10} & M_{00}L_{01} + M_{01}L_{11} \\ M_{10}L_{00} + M_{11}L_{10} & M_{10}L_{01} + M_{11}L_{11} \end{pmatrix}\right) \end{aligned}$$

Hierbei sind L_A, L_B geeignete Matrizen mit $L\,|_{I\times K} = L_A L_B^\top$, welche wegen der Annahme $I \times K \in L^Z_{\hat{T}}$ existieren. Auf Grund der Rekursion seien die Ergebnisse $R_{i(j)k} = A_{i(j)k}B^\top_{i(j)k} = \mathcal{T}_{\hat{k}}(M_{ij}L_{jk})$ bereits bekannt. Als nächstes wird entlang j addiert entsprechend Abschnitt 4.2.5:

$$R_{ik} = A_{ik}B^\top_{ik} := \mathcal{T}_{\hat{k}}\left(\begin{bmatrix} A_{i(0)k} & A_{i(1)k} \end{bmatrix}\begin{bmatrix} B_{i(0)k} & B_{i(1)k} \end{bmatrix}^\top\right)$$

Dies lässt sich sofort mit der Rekursion kombinieren, da die Operation $L + MN$ betrachtet wird, indem man die bereits berechnete $\mathcal{R}_k$-Matrix $R_{i(0)k}$ der Berechnung von $R_{i(1)k}$ als Zielmatrix übergibt. Entsprechend wird im Fall $d > 1$ die bereits berechnete Summe über alle $j' < j$ als Parameter übergeben.
Schließlich wird das Ergebnis der Rangreduktion entweder durch

$$\mathcal{T}_{\hat{k}}\left(\begin{bmatrix} L_A & A_{00} & A_{01} & 0 & 0 \\ & 0 & 0 & A_{10} & A_{11} \end{bmatrix}\begin{bmatrix} L_B & B_{00} & B_{01} & 0 & 0 \\ & 0 & 0 & B_{10} & B_{11} \end{bmatrix}\right)$$

approximiert entsprechend der obigen hierarchischen Approximation (vgl. [5], Seite 65) oder es wird als Approximation

$$\begin{aligned} X_0 Y_0^\top &:= \mathcal{T}_{\hat{k}}\left(\begin{bmatrix} L_A & A_{00} \\ & 0 \end{bmatrix}\begin{bmatrix} L_B & B_{00} \\ & 0 \end{bmatrix}\right) \\ X_{i+1} Y_{i+1}^\top &:= \mathcal{T}_{\hat{k}}\left(\begin{bmatrix} & \vdots & \\ X_i & A_{i,k(i)} & \\ & \vdots & \end{bmatrix}\begin{bmatrix} & \vdots & \\ Y_i & B_{i,k(i)} & \\ & \vdots & \end{bmatrix}\right) \end{aligned}$$

mit einer geeigneten Aufzählung $k(i)$ verwendet, das heisst, die Teilergebnisse werden mittels formatierter Additionen schrittweise auf den Rang $\hat{k}$ reduziert, vgl. [6], Seite 141.

Komplexität

Die Komplexität der formatierten Multiplikation kann nach diesen Vorbetrachtungen auf die Komplexität der exakten Multiplikaten ($\hat{T} = T \cdot \widetilde{T} \wedge \hat{k} = k \odot \tilde{k}$) zuzüglich der jeweiligen Kosten für die Approximation zurückgeführt werden. Hierfür soll zunächst die Komplexität der hierarchischen Approximation untersucht werden.

Der Aufwand für die hierarchische Konvertierung einer $\mathcal{H}$-Matrix in das $\mathcal{R}_k$-Format lässt sich dabei wie folgt abschätzen: Sei $H \in \mathcal{H}(\hat{k}, T)$ eine hierarchische Matrix über $\mathcal{I} \times \mathcal{K}$ (stellvertretend für die Teilbäume $T_{sub}(I \times K)$), welche in das $\mathcal{R}_k$-Format mit Rang k konvertiert werden soll. Es sei wieder T ein über $T_\mathcal{I}$ und $T_\mathcal{J}$ gebildeter $\mathcal{H}_\times$-Baum. In den nicht zulässigen Blättern L_T^N müssen zunächst die vollbesetzten Matrizen konvertiert werden (siehe Lemma 4.8). Dieser Aufwand[6] sei mit W_T^N bezeichnet:

$$W_T^N \leq 21 \cdot C_{\max}^2 \cdot \sum_{I \times J \in L_T^N} (|I| + |J|)$$

In den zulässigen Blättern von T muss entsprechend der Rang auf k reduziert werden, womit sich im schlimmsten Fall ergibt:

$$W_T^Z \leq 23 \cdot \hat{k}^3 \cdot \left|L_T^Z\right| + 6 \cdot \hat{k}^2 \cdot \sum_{I \times J \in L_T^Z} (|I| + |J|)$$

In jedem inneren Blatt $I \times J$ von T müssen schließlich $|(I \times J)E_T| \leq 2^{2d} + 1$ $\mathcal{R}_k$-Matrizen vom Rang k (bzw. nach obiger Bemerkungen eine $\mathcal{R}_k$-Matrix vom Rang $(2^{2d}+1) \cdot k$) auf den Rang k reduziert werden (Annahme: $(2^{2 \cdot d} + 1) \cdot k \leq C_{\max}$, siehe auch Bemerkung 4.2 und Lemma 4.8):

$$W_{V \setminus L} \leq 23 \cdot 2^{6d} \cdot k^3 \cdot |V_T \setminus L_T| + 6 \cdot 2^{4 \cdot d} \cdot k^2 \sum_{I \times J \in V_T \setminus L_T} (|I| + |J|)$$

Insgesamt ist der Aufwand für die hierarchische Konvertierung daher durch

$$\begin{aligned} &23 \cdot \max\{\hat{k}, (2^d + 1) \cdot k\}^3 \cdot \left|V_T \setminus L_T^N\right| \\ &+6 \cdot \max\{k, (2^d + 1) \cdot \hat{k}\}^2 \cdot \textstyle\sum_{I \times J \in V_T \setminus L_T^N} (|I| + |J|) + 21 \cdot C_{\max}^2 \cdot \sum_{I \times J \in L_T^N} (|I| + |J|) \end{aligned}$$

gegeben. Dabei gilt:

$$\begin{aligned} \sum_{I \times J \in V_T \setminus L_T^N} (|I| + |J|) &\leq \sum_{h=0}^{h_T} \sum_{I \times J \in V_T, h} (|I| + |J|) \\ &= C_{sp} \cdot \sum_{h=0}^{h_T} (\underbrace{\sum_{I \in V_{T_\mathcal{I}}, h} |I|}_{\leq |\mathcal{I}|} + \underbrace{\sum_{J \in V_{T_\mathcal{J}}, h} |J|}_{\leq |\mathcal{J}|}) \\ &\leq C_{sp} \cdot (h_T + 1) \cdot (|\mathcal{I}| + |\mathcal{J}|) \end{aligned}$$

Setzt man $\kappa = \max\{\hat{k}, (2^d + 1) \cdot k, C_{\max}\}$, so folgt hiermit:

Lemma 4.12. *Die hierarchische Konvertierung (Approximation) einer $\mathcal{H}(\hat{k}, T)$-Matrix in eine $\mathcal{R}_k$-Matrix vom Rang k hat mit $\kappa = \max\{\hat{k}, (2^d + 1) \cdot k, C_{\max}\}$ den Aufwand:*

$$23 \cdot \kappa^3 \cdot |V_T| + 6 \cdot \kappa^2 \cdot C_{sp}(|\mathcal{I}| + |\mathcal{J}|) \cdot (h_T + 1).$$

[6] In [6, S. 74, Lemma 5.18] wird $11 \cdot (|I|^3 + |J|^3)$ durch $22 \cdot |C_{\max}|^3$ abgeschätzt, ohne genauer darauf einzugehen ($C_{\max} = \min\{|I|, |J|\}$ist klar). Auch in [5, S. 78, Lemma 6.10] wird der Aufwand für die Konvertierung einer vollbesetzten Matrix in das $\mathcal{R}_k$-Format durch $21 \cdot C_{\max}^2 \cdot (|I| + |J|)$ abgeschätzt, jedoch auch ohne genauere Begründung. Dies enspricht allerdings mit $C_{\max} = \min\{|I|, |J|\}$ und $|I| + |J| \geq \max\{|I|, |J|\}$ der unter Lemma 4.8 aufgeführten Aufwandsabschätzung. Weiterhin wird dort zwar der Rang der Matrizen in den inneren Knoten mit $2^{2 \cdot d} \cdot \hat{k}$ angegeben, in der eigentlichen Abschätzung scheint aber nur $2^d \cdot \hat{k}$ in Lemma 4.8 eingesetzt worden zu sein.

Komplexität der exakten Multiplikation

Entsprechend [5, 6] wird nur der Fall $\mathcal{I} = \mathcal{J} = \mathcal{K}$ mit $T = \widetilde{T} = \hat{T}$ und $k = \tilde{k} = \hat{k}$ betrachtet: Die Multiplikation einer hierarchischen Matrix mit einer vollbesetzten bzw. $\mathcal{R}_k$-Matrix wird über die anfallenden Matrix-Vektor-Multiplikationen nach Abschnitt 4.2.4 abgeschätzt. Man setze $\kappa = \max\{k, C_{\max}\}$. In den Blättern $I \times K$ von $T \cdot T$ treten bei jedem Teilprodukt über $(I^\uparrow \times J^\uparrow \times K^\uparrow) \in O_{T\cdot T}(I \times K)$ mindestens eine $\mathcal{R}_k$- oder vollbesetzte Matrix auf, so dass höchstens κ viele Matrix-Vektor-Multiplikationen benötigt werden. Die Addition dieser erfolgt entweder durch Kopieren oder durch explizite Addition; dieser Aufwand wurde jedoch bereits in dem Abschnitt 4.2.4 mitbetrachtet. Damit erhält man als Schranke für den Aufwand[7]:

$$
\begin{aligned}
L + MN &= \sum_{I\times K \in L(T\cdot T)} \sum_{(I^\uparrow \times J^\uparrow \times K^\uparrow)\in O(I\times K)} (L|_{I^\uparrow\times K^\uparrow} + M|_{I^\uparrow\times J^\uparrow}\, N|_{J^\uparrow\times K^\uparrow})|_{I\times K} \\
&= \sum_{I\times K \in L(T\cdot T)} \underbrace{\sum_{(I^\uparrow \times J^\uparrow \times K^\uparrow)\in O(I\times K)} L|_{I\times K} + M|_{I\times J^\uparrow}\, N|_{J^\uparrow\times K}}_{\leq \kappa \max\{W^{w+\mathcal{H}(k,T)\cdot v}_{\mathcal{H},k,C_{\max},I\times\mathcal{I},\eta}, W^{w+\mathcal{H}(k,T)\cdot v}_{\mathcal{H},k,C_{\max},\mathcal{I}\times K,\eta}\}} \\
&= \kappa \cdot \sum_{I\times K\in L_{T\cdot T}} \max\{W^{w+\mathcal{H}(k,T)\cdot v}_{\mathcal{H},k,C_{\max},I\times\mathcal{I},\eta}, W^{w+\mathcal{H}(k,T)\cdot v}_{\mathcal{H},k,C_{\max},\mathcal{I}\times K,\eta}\} \\
&\leq \kappa \cdot \sum_{I\times K\in L_{T\cdot T}} W^{w+\mathcal{H}(k,T)\cdot v}_{\mathcal{H},k,C_{\max},I\times\mathcal{I},\eta} + W^{w+\mathcal{H}(k,T)\cdot v}_{\mathcal{H},k,C_{\max},\mathcal{I}\times K,\eta} \\
&\leq \kappa \cdot \sum_{h=0}^{h_{T\cdot T}} \sum_{I\times K\in L_{T\cdot T,h}} W^{w+\mathcal{H}(k,T)\cdot v}_{\mathcal{H},k,C_{\max},I\times\mathcal{I},\eta} + W^{w+\mathcal{H}(k,T)\cdot v}_{\mathcal{H},k,C_{\max},\mathcal{I}\times K,\eta} \\
&\leq 2\cdot\kappa\cdot\sum_{h=0}^{h_T} C_{sp,T\cdot T}\cdot W^{w+\mathcal{H}(k,T)\cdot v}_{\mathcal{H},k,C_{\max},\mathcal{I},\eta} \\
&\leq 4\cdot\kappa\cdot(h_T+1)\cdot C^2_{sp,T}\cdot W_{\mathcal{H},k,C_{\max},\mathcal{I},\eta}
\end{aligned}
$$

Insgesamt ergibt sich somit folgendes Lemma:

Lemma 4.13. *Der Aufwand der exakten Multiplikation ist durch*

$$
W^{\mathcal{H}(k,T)+\mathcal{H}(k,T)\cdot\mathcal{H}(k,T)}_{\mathcal{H},k,C_{\max},\mathcal{I},\eta} := 4\cdot\kappa\cdot(h_T+1)\cdot C_{sp}\cdot W_{\mathcal{H},k,C_{\max},\mathcal{I},\eta}
$$

beschränkt.

Komplexität der Konvertierung

Bemerkung 4.3. Um die Komplexität der Konvertierung abzuschätzen, soll der Rang $k \odot k$ noch genauer betrachtet werden. Hierzu zerlegt man die Mengen $O_{T\cdot T}(I\times K)$ wieder bezüglich

[7] Nach Definition der nicht zulässigen Blätter von $T \cdot T$ muss es mindestens ein Teilprodukt über einem Knoten $(I^\uparrow \times J^\uparrow \times K^\uparrow) \in O_{T\cdot T}(I \times K)$ geben, in welchem beide Blätter aus T nicht zulässig sind. In [6] wird daher behauptet, dass für $I \times K \in L^N_{T\cdot T}$ daraus $|I|, |K| \leq C_{\max}$ folgt. Dies führt auf eine andere obere Schranke. Es wird allerdings nicht erwähnt, warum nicht der Fall $I = I^\uparrow \wedge K = K^\uparrow$ mit $|I|, |K| > C_{\max}$ und $\left|J^\uparrow\right| \leq C_{\max}$ eintreten kann. In [5] wird auf diese Behauptung verzichtet, womit die hier angegebene obere Schranke hergeleitet wird.

der Höhen: $O_{T\cdot T,h}(I \times K) := O_{T\cdot T}(I \times K) \cap h_{T\cdot T}^{-1}(h)$. Dann folgt

$$\begin{aligned}
|O_{T\cdot T,h}(I \times K)| &= \left|\left\{\left(I^{\uparrow} \times J^{\uparrow} \times K^{\uparrow}\right) \in L_{\mathbb{T},h} \mid \left(I^{\uparrow} \times K^{\uparrow}\right) \in E^{*}_{T\cdot T}\left(I \times K\right)\right\}\right| \\
&\leq \left|\left\{J^{\uparrow} \in V_{T_{\mathcal{I}},h} \mid \left(E^{h}_{T\cdot T} I \times J^{\uparrow}\right) \in V_{T,h} \wedge \left(J^{\uparrow} \times E^{h}_{T\cdot T} K\right) \in L_{\mathbb{T}}\right\}\right| \\
&\leq \max\{\left|\left\{J^{\uparrow} \in V_{T_{\mathcal{I}},h} \mid \left(J^{\uparrow} \times E^{h}_{T\cdot T} K\right) \in V_{T,h}\right\}\right|, \\
&\qquad \left|\left\{J^{\uparrow} \in V_{T_{\mathcal{I}},h} \mid \left(E^{h}_{T\cdot T} K \times J^{\uparrow}\right) \in V_{T,h}\right\}\right|\} \\
&\leq C_{sp}
\end{aligned}$$

und damit $|O_{T\cdot T}(I \times K)| \leq (h_{T_{\mathcal{I}}} + 1) \cdot C_{sp}$. Entsprechend folgt

$$k \odot k \leq k \cdot C_{sp} \cdot (h_{T_{\mathcal{I}}} + 1).$$

Es verbleibt den Aufwand der verschiedenen Konvertierungsverfahren zu betrachten. Die Varianten unterscheiden sich nur bezüglich des Aufwands in den zulässigen Blättern von T.
In den nicht zulässigen Blättern von T müssen - wie beschrieben - entweder vollbesetzte Matrizen kopiert (die Addition zur Zielmatrix wurde ja bereits in der exakten Multiplikation beachtet) werden oder schlimmstenfalls C_{id} $\mathcal{R}_k$-(Teil)-Matrizen ausmultipliziert werden, welche maximal den Rang $k \odot k$ besitzen. Dies führt auf den Aufwand

$$\begin{aligned}
W^N &\leq \sum_{I\times K \in L_T^N} 2 \cdot C_{id} \cdot k \odot k \cdot |I| \cdot |K| \\
&\leq 2 \cdot k \cdot C_{id} \cdot C_{sp} \cdot (h_T + 1) \cdot C_{\max} \cdot \underbrace{\sum_{I\times K \in L_T^N} |I| + |K|}_{\leq 2 \cdot C_{sp} \cdot |\mathcal{I}| \cdot (h_{T_{\mathcal{I}}} + 1)} \\
&\leq 4 \cdot k \cdot C_{id} \cdot C_{sp}^2 \cdot (h_{T_{\mathcal{I}}} + 1)^2 \cdot C_{\max} \cdot |\mathcal{I}|\,.
\end{aligned}$$

Die benötigten Operationen in den zulässigen Blättern von T lassen sich wie folgt abschätzen:

- In den zulässigen Blättern muss im Fall der *nachträglichen exakten Konvertierung* noch eine aus maximal C_{id} Blättern, welche maximal den Rang $\max\{C_{\max}, C_{sp} \cdot (h_T + 1) \cdot k\}$ besitzen, bestehende hierarchische Matrix in das $\mathcal{R}_k$-Format überführt werden. Sind unter diesen Blättern auch vollbesetzte Matrizen $F \in \mathbb{R}^{m\times n}$, so wird für diese die *kanonische* $\mathcal{R}_k$-Darstellung verwendet. Mit

  $$\tilde{k} := C_{id} \cdot \max\{C_{\max}, C_{sp} \cdot (h_{T_{\mathcal{I}}} + 1) \cdot k\}$$

  kann dann Lemma 4.9 angewandt werden:

  $$6 \cdot \tilde{k} \cdot W_{\mathcal{H},k,C_{\max},\mathcal{I},\eta} + 23 \cdot \tilde{k} \cdot \#L_T^Z$$

- Wird stattdessen die *hierarchische Konvertierung* verwendet, so fallen in jedem zulässigen Blatt $I \times K \in L_{T,h}$ noch nach Lemma 4.12 die Kosten

  $$23 \cdot \kappa^3 \cdot \left|V_{T_{sub}(I\times K)}\right| + 6 \cdot \kappa^2 \cdot C_{sp} \cdot (|I| + |K|) \cdot (h_{T_{sub}(I\times K)} + 1)$$

an. Da jeder innere Knoten von $T \cdot T$ nach Konstruktion wieder mindestens zwei Nachfolger besitzt, ist die Höhe eines jeden Teilbaums $T_{sub}(I \times K)$ durch $\log C_{id}$ beschränkt. Weiterhin sind die Teilbäume disjunkt, das heisst, es gilt:

$$\sum_{I \times K \in L_T^Z} \left|V_{T_{sub}(I\times K)}\right| \leq |V_{T \cdot T}| \leq C_{sp,T\cdot T} \cdot |V_{T_\mathcal{I}}| \leq 2 \cdot C_{sp}^2 \cdot |\mathcal{I}|$$

Verwendet man wieder $\sum_{I\times K \in L_T}(|I| + |K|) \leq 2 \cdot C_{sp} \cdot (h_{T_\mathcal{I}} + 1) \cdot |\mathcal{I}|$, so folgt ingesamt für den zusätzlichen Aufwand im Fall der nachträglichen hierarchischen Approximation:

$$\begin{array}{rc} & \left(23 \cdot \kappa^3 \cdot \left|V_{T_{sub}(I\times K)}\right| + 6 \cdot \kappa^2 \cdot C_{sp} \cdot (|I| + |K|) \cdot (h_{T_{sub}(I\times K)} + 1)\right) \cdot \left|L_T^Z\right| \\ \leq & 46 \cdot \kappa^3 \cdot C_{sp}^2 \cdot |\mathcal{I}| + 12 \cdot \kappa^2 \cdot C_{sp}^2 \cdot (\log C_{id} + 1)^2 \cdot |\mathcal{I}| \end{array}$$

mit

$$\begin{array}{rcl} \kappa & = & \max\{\tilde{k}, (2^d + 1) \cdot k, C_{\max}\} \\ & = & \max\{C_{id} \cdot \max\{C_{\max}, C_{sp} \cdot (h_{T_\mathcal{I}} + 1) \cdot k\}, (2^d + 1) \cdot k, C_{\max}\}. \end{array}$$

- Wird die *Approximation mittels hierarchischer Konvertierung parallel zur Berechnung* durchgeführt, so fallen pro Blatt $I \times K \in L_T^Z$ im allgemeinen mehrere hierarchische Konvertierungen an, da es mehrere $J \in V_{T_\mathcal{I}}$ mit $I \times J \times K \in V_\mathbb{T}$ gibt, welche getrennt voneinander traversiert werden. Für jeden Knoten $I \times K \in V_T$ ist jedoch die Anzahl $|\{J \in V_{T_\mathcal{I}} \mid I \times J \times K \in V_\mathbb{T}\}|$ durch C_{sp} beschränkt. Da weiterhin die Anzahl der Knoten der Teilbäume $T_{sub}(I \times K)$ durch C_{id} beschränkt ist, kann es höchsten $C_{id} \cdot C_{sp}$ Knoten geben in $\mathbb{T}$ pro Blatt $I \times K \in L_T^Z$, an welchen eine aus $\leq 2^{2\cdot d} + 1$ $\mathcal{R}_k$-Matrizen vom Rang k bestehende Blockmatrix auf den Rang k reduziert werden muss.
  Dafür ist der Rang der Blätter von $\mathbb{T}$ durch die anfallenden Produkte einer vollbesetzten bzw. $\mathcal{R}_k$-Matrix mit einer hierarchischen Matrix durch $\max\{C_{\max}, k\}$ beschränkt.

- Wird schließlich die schrittweise Konvertierung parallel zur Berechnung verwendet, so folgt aus der obigen Betrachtung, dass der zusätzliche Aufwand durch die Konvertierung aus den formatierten Additionen in $C_{id} \cdot C_{sp} \cdot \#L_T^Z$ Blättern von $\mathbb{T}$ von jeweils $2^{2\cdot d} + 1$ $\mathcal{R}_k$-Matrizen mit Rang k besteht:

  $$C_{id} \cdot C_{sp} \cdot \#L_T^Z \cdot 2^{2\cdot d} \cdot (6 \cdot (m + n)(2 \cdot k)^2 + 23 \cdot (2 \cdot k)^3)$$

  Wiederum fallen noch zusätzlich die notwendigen Reduktionen der direkten Matrixprodukte in den Blättern von $\mathbb{T}$ an.

Damit ist die Komplexität der Multiplikation vollständig untersucht. Es verbleibt C_{id} auf die Eigenschaften der gegebenen Daten entsprechend C_{sp} zurückzuführen. Hierfür sei auf die angegebene Literatur verwiesen. Entsprechendes gilt für die Invertierung von $\mathcal{H}$-Matrizen. Es sei nur angemerkt, dass deren Aufwand[8] dem Aufwand der Matrixmultiplikation entspricht, da die Invertierung der Blockmatrizen rekursiv auf die Blätter ausgelagert werden kann, so dass nur noch Matrixprodukte in den inneren Knoten anfallen.

[8] unter Annahme, dass der Aufwand der Invertierung einer vollbesetzten Matrix dem Aufwand der Matrixmultiplikation entspricht

Kapitel 5

Approximation mit Taylor-Reihen

Ulrich Poppendieck

Ein klassisches Anwendungsfeld hierarchischer Matrizen bilden Diskretisierungsverfahren bei Randintegralgleichungen. Dies soll am Beispiel des zweidimensionalen Einfachschichtpotentials V betrachtet werden. Gegeben sei eine hierarchische Partitionierung der Steifigkeitsmatrix V_h. Für paarweise wohlgetrennte Cluster können nun einzelne Blöcke von V_h durch Niedrigrangmatrizen approximiert werden. Hierbei orientieren wir uns an [17, S. 312-319].

5.1 Allgemeine Approximation

Definition 5.1. *Zwei Cluster ω_i^κ und ω_j^λ heißen zueinander **zulässig**, falls*

$$dist(\omega_i^\kappa, \omega_j^\lambda) \geq \eta \, max\{diam\, \omega_i^\kappa, diam\, \omega_j^\lambda\}$$

mit einer vorgegebenen Konstante $\eta > 1$ erfüllt ist.

Seien ω_i^λ und ω_j^λ zwei zueinander maximal zulässige Cluster, d. h. es existieren keine zwei zueinander zulässigen Cluster eines niedrigeren Levels, die ω_i^λ bzw. ω_j^λ enthalten. Zu berechnen sind die Einträge der Block-Matrix $V_h^{\lambda,ij}$,

$$V_h^{\lambda,ij}[l,k] = \int\limits_{\tau_l} \int\limits_{\tau_k} U^*(x,y) ds_y ds_x \quad \text{für } \tau_k \in \omega_i^\lambda, \tau_l \in \omega_j^\lambda. \tag{5.1}$$

Hierbei lässt sich die Fundamentallösung $U^*(x,y) = -\frac{1}{2\pi} \log|x-y|$ approximieren durch eine näherungsweise Aufspaltung in Funktionen, die nur vom Integrationspunkt $y \in \omega_i^\lambda$ beziehungsweise vom Beobachtungspunkt $x \in \omega_j^\lambda$ abhängen,

$$U_\varrho^*(x,y) = \sum_{m=0}^{\varrho} f_m^{\lambda,j}(x) g_m^{\lambda,i}(y) \quad \text{für } (x,y) \in \omega_j^\lambda \times \omega_i^\lambda. \tag{5.2}$$

Dabei soll die Fehlerabschätzung

$$|U^*(x,y) - U_\varrho^*(x,y)| \leq c(\eta, \varrho) \quad \text{für } (x,y) \in \omega_j^\lambda \times \omega_i^\lambda \tag{5.3}$$

gelten, wobei die Konstante $c(\eta, \varrho)$ vom Zulässigkeitsparameter η, der Approximationsordnung ϱ und den verwendeten Funktionen $f_m^{\lambda,j}$ und $g_m^{\lambda,i}$ abhängt. Mit (5.1) und (5.2) lassen sich nun die Einträge der Block-Matrix $V_h^{\lambda,ij}$ durch

$$\tilde{V}_h^{\lambda,ij}[l,k] := \int_{\tau_l} \int_{\tau_k} U_\varrho^*(x,y) ds_y ds_x \quad \text{für } \tau_k \in \omega_i^\lambda, \tau_l \in \omega_j^\lambda \tag{5.4}$$

approximieren. Mit der Fehlerabschätzung (5.3) ergibt sich also

$$|V_h^{\lambda,ij}[l,k] - \tilde{V}_h^{\lambda,ij}[l,k]| \leq c(\eta,\varrho)\Delta_k\Delta_l \quad \text{für } \tau_k \in \omega_i^\lambda, \tau_l \in \omega_j^\lambda. \tag{5.5}$$

Durch Einsetzen der Reihenentwicklung (5.2) in (5.4) erhält man

$$\tilde{V}_h^{\lambda,ij}[l,k] = \sum_{m=0}^{\varrho} \int_{\tau_l} f_m^{\lambda,j}(x) ds_x \int_{\tau_k} g_m^{\lambda,i}(y) ds_y \quad \text{für } \tau_k \in \omega_i^\lambda, \tau_l \in \omega_j^\lambda.$$

Es sind also für alle Randelemente $\tau_k \in \omega_i^\lambda$ und $\tau_l \in \omega_j^\lambda$ die Vektoren

$$a_{m,l}^{\lambda,j} := \int_{\tau_l} f_m^{\lambda,j}(x) ds_x \quad \text{und} \quad b_{m,k}^{\lambda,i} := \int_{\tau_k} g_m^{\lambda,i}(y) ds_y$$

zu berechnen, mit $m = 0, ..., \varrho$, $l = 1, ..., N_j^\lambda$ und $k = 1, ..., N_i^\lambda$, wobei N_i^λ die Anzahl aller Randelemente τ_k im Cluster ω_i^λ bezeichnet. Der Aufwand zur Speicherung und Anwendung der Approximationsmatrix

$$\tilde{V}_h^{\lambda,ij} = \sum_{m=0}^{\varrho} \underline{a}_m^{\lambda,j} (\underline{b}_m^{\lambda,i})^\top$$

beträgt demzufolge

$$(\varrho + 1)(N_i^\lambda + N_j^\lambda).$$

Nun kann eine Approximation $\tilde{V}_h$ der globalen Steifigkeitsmatrix V_h erklärt werden,

$$\tilde{V}_h = \sum_{\lambda=0}^{L} \underbrace{\sum_{j=1}^{4^\lambda} \sum_{i=1}^{4^\lambda}}_{\omega_i^\lambda, \omega_j^\lambda \text{ maximal zulässig}} (P_j^\lambda)^\top \tilde{V}_h^{\lambda,ij} P_i^\lambda + \underbrace{\sum_{j=1}^{4^L} \sum_{i=1}^{4^L}}_{\omega_i^L, \omega_j^L \text{ nicht zulässig}} (P_j^L)^\top V_h^{L,ij} P_i^L. \tag{5.6}$$

Hierbei beschreibt P_j^λ die Zuordnung der Randelemente $\{\tau_l\}_{l=1}^N$ auf das jeweilige Cluster ω_j^λ, $V_h^{\lambda,ij}$ und $\tilde{V}_h^{\lambda,ij}$ sind die durch (5.4) beziehungsweise (5.1) erklärten Matrizen und L bezeichnet das maximale Verfeinerungslevel des Clusterbaums. Anhand entsprechender Skizzen stellt man fest, dass der Gesamtaufwand zur Speicherung und Anwendung der Approximationsmatrix $\tilde{V}_h$ proportional zu

$$(\varrho + 1)(\eta + 1)^2 (L + 1)N + \eta^2 N.$$

ist. Für die Approximation (5.6) lässt sich der Fehler $V_h - \tilde{V}_h$ wie folgt abschätzen.

Lemma 5.1. *Für jedes Paar maximal zulässiger Cluster ω_i^λ und ω_j^λ gelte die Fehlerabschätzung (5.3). Für die durch (5.6) erklärte Approximation $\tilde{V}_h$ der globalen Steifigkeitsmatrix V_h gilt dann die Fehlerabschätzung*

$$|((V_h - \tilde{V}_h)\underline{w}, \underline{v}) \leq c(\eta, \varrho)|\Gamma|\|w_h\|_{L_2(\Gamma)}\|v_h\|_{L_2(\Gamma)}$$

für alle $\underline{w}, \underline{v} \in \mathbb{R}^N \leftrightarrow w_h, v_h \in S_h^0(\Gamma)$.

Beweis. Unter Verwendung von (5.5) folgt zunächst

$$\begin{aligned}\left|((V_h - \tilde{V}_h)\underline{w}, \underline{v})\right| &\leq \sum_{\lambda=0}^{L} \underbrace{\sum_{j=1}^{4^\lambda}\sum_{i=1}^{4^\lambda}}_{\omega_i^\lambda,\omega_j^\lambda \text{ maximal zulässig}} \sum_{\tau_k\in\omega_i^\lambda}\sum_{\tau_l\in\omega_j^\lambda} \left|V_h^{\lambda,ij}[l,k] - \tilde{V}_h^{\lambda,ij}[l,k]\right| |w_k|\,|v_l| \\ &\leq c(\eta,\varrho)\sum_{\lambda=0}^{L} \underbrace{\sum_{j=1}^{4^\lambda}\sum_{i=1}^{4^\lambda}}_{\omega_i^\lambda,\omega_j^\lambda \text{ maximal zulässig}} \sum_{\tau_k\in\omega_i^\lambda}\sum_{\tau_l\in\omega_j^\lambda} \Delta_k\,|w_k|\,\Delta_l\,|v_l|\,.\end{aligned}$$

Aufgrund der maximalen Zulässigkeit der Cluster ω_i^λ und ω_j^λ tritt jede Kombination von Randelementen τ_k und τ_l höchstens einmal auf. Somit gilt

$$\left|((V_h - \tilde{V}_h)\underline{w}, \underline{v})\right| \leq c(\eta,\varrho)\sum_{k=1}^{N}\sum_{l=1}^{N} \Delta_k\,|w_k|\,\Delta_l\,|v_l|\,.$$

Mit der Hölder-Ungleichung folgt

$$\sum_{k=1}^{N}\Delta_k\,|w_k| \leq \left(\sum_{k=1}^{N}\Delta_k\right)^{1/2}\left(\sum_{k=1}^{N}\Delta_k w_k^2\right)^{1/2} = |\Gamma|^{1/2}\,\|w_h\|_{L_2(\Gamma)},$$

und daraus ergibt sich die Behauptung. □

Aus Lemma 5.1 folgt bei geeigneter Wahl von η und ϱ die positive Definitheit von $\tilde{V}_h$ sowie eine Abschätzung des Fehlers der zu berechnenden Näherungslösung.

5.2 Darstellung mit Taylor-Reihen

Im folgenden wird die Darstellung (5.2) über die Taylor-Entwicklung der Fundamentallösung hinsichtlich der Integrationsvariablen $y_i \in \omega_i^\lambda$ anhand des Beispiels $U^*(x,y) = -\frac{1}{2\pi}\log|x-y|$ hergeleitet. Zunächst wird die Taylor-Entwicklung einer skalaren Funktion f betrachtet. Für $p \in \mathbb{N}$ ist

$$f(1) = f(0) + \sum_{n=1}^{p}\frac{1}{n!}\frac{d^n}{dt^n}f(t)_{|t=0} + \frac{1}{p!}\int_0^1 (1-s)^p\frac{d^{p+1}}{ds^{p+1}}f(s)ds.$$

Sei nun y_i^λ das Zentrum des Clusters ω_i^λ. Für beliebiges $y \in \omega_i^\lambda$ und $t \in [0,1]$ sei

$$f(t) := U^*(x, y_i^\lambda + t(y - y_i^\lambda)).$$

Es ist also

$$f(t) = -\frac{1}{2\pi}\log\sqrt{(x_1 - y_{i1}^\lambda - t(y_1 - y_{i1}^\lambda))^2 + (x_2 - y_{i2}^\lambda - t(y_2 - y_{i2}^\lambda))^2}.$$

Durch einmaliges Ableiten erhält man somit

$$\frac{d}{dt}f(t) = \sum_{j=1}^{2}(y_j - y_{i,j}^\lambda)\frac{\partial}{\partial z_j}U^*(x,z)_{|z=y_i^\lambda + t(y-y_i^\lambda)},$$

und durch rekursive Anwendung folgt daraus für $1 \leq n \leq p$

$$\frac{d^n}{dt^n}f(t) = \sum_{|\alpha|=n}\frac{n!}{\alpha!}(y - y_i^\lambda)^\alpha D_z^\alpha U^*(x,z)_{|z=y_i^\lambda + t(y-y_i^\lambda)},$$

mit Multiindizees $\alpha \in \mathbb{N}_0^2$. Entwickelt man $U^*(x,y)$ nun um das Clusterzentrum y_i^λ, so ergibt sich die Darstellung

$$U^*(x,y) = U_\varrho^*(x,y) + R_p(x,y),$$

mit

$$U_\varrho^*(x,y) = U^*(x,y_i^\lambda) + \sum_{n=1}^{p}\sum_{|\alpha|=n}\frac{1}{\alpha!}(y - y_i^\lambda)^\alpha D_z^\alpha U^*(x,z)_{|z=y_i} \tag{5.7}$$

als Approximation der Fundamentallösung. Die Darstellung (5.2) ergibt sich also mit

$$f_0^{\lambda,j}(x) := U^*(x,y_i^\lambda), \quad g_0^{\lambda,i}(y) := 1$$

und

$$f_{n,\alpha}^{\lambda,j}(x) := D_z^\alpha U^*(x,z)_{|z=y_i^\lambda}, \quad g_{n,\alpha}^{\lambda,i}(y) := \frac{1}{\alpha!}(y - y_i^\lambda)^\alpha$$

für $n = 1, ..., p$ und $|\alpha| = n$. Für jedes $n \in [1, ...p]$ existieren genau $n+1$ Multiindizees $\alpha \in \mathbb{N}_0^2$ mit $|\alpha| = n$. Also ergibt sich für die Anzahl $\varrho + 1$ der Terme in (5.2)

$$\varrho + 1 = 1 + \sum_{n=1}^{p}(n+1) = \frac{1}{2}(p+1)(p+2).$$

Für die Herleitung der Fehlerabschätzung (5.3) muss nun das Restglied

$$R_p(x,y) = \frac{1}{p!}\int_0^1 (1-s)^p \sum_{|\alpha|=p+1}\frac{(p+1)!}{\alpha!}(y - y_i^\lambda)^\alpha D_z^\alpha U^*(x,z)_{|z=y_i^\lambda + s(y-y_i^\lambda)}ds \tag{5.8}$$

abgeschätzt werden. Dafür werden zunächst die Ableitungen von U^* abgeschätzt.

Lemma 5.2. *Seien ω_i^λ und ω_j^λ maximal zulässig. Für $|\alpha| = p \in \mathbb{N}$ gilt*

$$|D_y^\alpha U^*(x,y)| \leq \frac{1}{2\pi}\frac{3^{p-1}(p-1)!}{|x-y|^p}$$

für alle $(x,y) \in \omega_j^\lambda \times \omega_i^\lambda$.

Beweis. Für die Ableitungen der Funktion $f(x,y) = \log|x-y|$ ist

$$\frac{\partial}{\partial y_i} f(x,y) = \frac{y_i - x_i}{|x-y|^2}, \quad i = 1,2.$$

Für die zweiten Ableitungen ergibt sich

$$\frac{\partial^2}{\partial y_i^2} f(x,y) = \frac{1}{|x-y|^2} - 2\frac{(y_i - x_i)^2}{|x-y|^4}, \quad i = 1,2,$$

und

$$\frac{\partial^2}{\partial y_1 \partial y_2} f(x,y) = -2\frac{(y_1 - x_1)(y_2 - x_2)}{|x-y|^4}.$$

Allgemein gilt für $|\alpha| = \varrho \in \mathbb{N}$

$$D_y^\alpha f(x,y) = \sum_{|\beta| \le \varrho} a_\beta^\varrho \frac{(y-x)^\beta}{|x-y|^{|\beta|+\varrho}} \tag{5.9}$$

mit gewissen Koeffizienten a_β^ϱ. Daraus folgt

$$|D_y^\alpha f(x,y)| \le \sum_{|\beta| \le \varrho} |a_\beta^\varrho| \frac{1}{|x-y|^\varrho} = \frac{c_\varrho}{|x-y|^\varrho}.$$

Ein Vergleich mit den ersten und zweiten Ableitungen der Funktion $f(x,y)$ liefert $c_1 = 1$ und $c_2 = 3$. Eine allgemeine Abschätzung der Konstanten c_ϱ für $\varrho \ge 2$ folgt nun durch vollständige Induktion. Aus (5.9) ergibt sich für $i = 1,2$ und $j \ne i$

$$\begin{aligned}\frac{\partial}{\partial y_i} D_y^\alpha f(x,y) &= \sum_{|\beta| \le \varrho} a_\beta^\varrho \frac{\partial}{\partial y_i} \frac{(y-x)^\beta}{|x-y|^{|\beta|+\varrho}} \\ &= \sum_{|\beta| \le \varrho} a_\beta^\varrho \left[\beta_i \frac{(y_i - x_i)^{\beta_i - 1}(y_j - x_j)^{\beta_j}}{|x-y|^{|\beta|+\varrho}} - (|\beta| + \varrho) \frac{(y_i - x_i)^{\beta_i+1}(y_j - x_j)^{\beta_j}}{|x-y|^{|\beta|+\varrho+2}} \right].\end{aligned}$$

Mit

$$|y_i - x_i| \le |x-y|, \quad \beta_i \le |\beta| \le \varrho, \quad \beta_i + \beta_j \le |\beta| \quad \text{für } i \ne j$$

folgt daraus die Abschätzung

$$\left| \frac{\partial}{\partial y_i} D_y^\alpha f(x,y) \right| \le \frac{3\varrho}{|x-y|^{\varrho+1}} \sum_{|\beta| \le \varrho} |a_\beta^\varrho| = \frac{3\varrho c_\varrho}{|x-y|^{\varrho+1}} = \frac{c_{\varrho+1}}{|x-y|^{\varrho+1}}$$

und somit

$$c_{\varrho+1} = 3\varrho c_\varrho = 3^\varrho \varrho!.$$

Mit $\varrho = p$ ergibt sich dann die Behauptung. □

Damit kann nun über die Abschätzung von (5.8) eine Fehlerabschätzung für die approximierte Fundamentallösung (5.7) gewonnen werden.

Lemma 5.3. *Seien ω_i^λ und ω_j^λ zwei zueinander zulässige Cluster. Für die durch (5.7) erklärte Approximation $U_\varrho^*(x,y)$ der Fundamentallösung $U^*(x,y)$ gilt dann die Fehlerabschätzung*

$$\left|U^*(x,y) - U_\varrho^*(x,y)\right| \leq 6^p \left(\frac{1}{\eta}\right)^{p+1} \quad \text{für alle } (x,y) \in \omega_j^\lambda \times \omega_i^\lambda.$$

Beweis. Mit Lemma 5.2 folgt für $(x,y) \in \omega_j^\lambda \times \omega_i^\lambda$ mit der Cluster-Zulässigkeitsbedingung

$$\begin{aligned}
\left|U^*(x,y) - U_\varrho^*(x,y)\right| &= |R_p(x,y)| \\
&\leq \frac{1}{p!}\left|y - y_i^\lambda\right|^{p+1} \max_{\bar{y}\in\omega_i^\lambda} \sum_{|\alpha|=p+1} \frac{(p+1)!}{\alpha!} \left|D_z^\alpha U^*(x,z)_{|z=\bar{y}}\right| \int_0^1 (1-s)^p ds \\
&\leq \frac{1}{(p+1)!}\left|y - y_i^\lambda\right|^{p+1} \max_{\bar{y}\in\omega_i^\lambda} \sum_{|\alpha|=p+1} \frac{(p+1)!}{\alpha!} \frac{1}{2\pi} \frac{3^p p!}{|x-\bar{y}|^{p+1}} \\
&\leq \frac{3^p p!}{2\pi}\left(\sum_{|\alpha|=p+1} \frac{1}{\alpha!}\right) \frac{[\text{diam } \omega_i^\lambda]^{p+1}}{[\text{dist } (\omega_i^\lambda, \omega_j^\lambda)]^{p+1}} \\
&\leq \frac{3^p p!}{2\pi}\left(\sum_{|\alpha|=p+1} \frac{1}{\alpha!}\right)\left(\frac{1}{\eta}\right)^{p+1} \\
&= \frac{3^p p!}{2\pi}\left(\sum_{a=0}^{p+1} \frac{1}{a!(p+1-a)!}\right)\left(\frac{1}{\eta}\right)^{p+1} \\
&= \frac{3^p p!}{2\pi(p+1)!}\left(\sum_{a=0}^{p+1} \frac{(p+1)!}{a!(p+1-a)!}\right)\left(\frac{1}{\eta}\right)^{p+1} \\
&= \frac{3^p}{2\pi(p+1)}\left(\sum_{a=0}^{p+1} \binom{p+1}{a}\right)\left(\frac{1}{\eta}\right)^{p+1} \\
&= \frac{3^p}{2\pi(p+1)}(1+1)^{p+1}\left(\frac{1}{\eta}\right)^{p+1} \\
&= \frac{3^p 2^{p+1}}{2\pi(p+1)}\left(\frac{1}{\eta}\right)^{p+1} \\
&\leq \frac{3^p 2^p}{p+1}\left(\frac{1}{\eta}\right)^{p+1} \leq 6^p \left(\frac{1}{\eta}\right)^{p+1}
\end{aligned}$$

und somit die Behauptung. □

Damit gilt für $c(\eta,\varrho) = 6^p\left(\frac{1}{\eta}\right)^{p+1}$.

Die Approximation der Fundamentallösung $U^*(x,y)$ mittels der Taylorreihe erfordert die Berechnung aller auftretenden Ableitungen bis zur Ordnung p. Eine andere Möglichkeit ist das Verwenden spezieller Reihenentwicklungen der Fundamentallösung unter Verwendung harmonischer Kugelfunktionen. Beide Zugänge sind jedoch stark problemabhängig. Ein allgemeinerer Zugang beruht auf der Approximation der Fundamentallösung mit Tschebyscheff-Polynomen.

Kapitel 6

Hierarchische Matrizen und Eigenwertprobleme

Jan Jung

In diesem Teil soll auf die Anwendung von hierarchischen Matrizen und den dadurch entstehenden Vorteil für den Aufwand der benutzten Algorithmen bei der Lösung von Eigenwertproblemen eingegangen werden. Dies wird anhand eines Algorithmus [12, 13] zur iterativen Berechnung der Eigenpaare des Laplace-Operators in 2D exemplarisch durchgeführt.
In Abschnitt 6.1 sind das Eigenwertproblem und der Algorithmus zu dessen Lösung aufgeführt. Abschnitt 6.2 beschreibt dann die im Algorithmus verwendeten Matrix-Faktorisierungen in $\mathcal{H}$-Arithmetik, woraufhin Abschnitt 6.3 diese dann auf den Algorithmus anwendet, den dadurch erhaltenen Laufzeitvorteil aufzeigt und mit einer Rückwärtsfehleranalyse schließt.

6.1 Das Eigenwertproblem für den Laplace-Operator in 2D

Die Anwendung von hierarchischen Matrizen auf Eigenwertprobleme großer Dimension soll hier am Beispiel des diskreten Laplace-Operators in 2D veranschaulicht werden.
Die Lösung des kontinuierlichen Eigenwertproblems

$$\begin{aligned} -\Delta u &= \lambda u \qquad \text{in } \Omega = (0,1)^2, \\ u &= 0 \qquad \text{auf } \Gamma = \partial\Omega, \end{aligned} \tag{6.1}$$

ist

$$u(x,y) = \sin(k\pi x)\sin(\ell\pi y), \tag{6.2}$$

mit den zugehörigen Eigenwerten

$$\lambda = \left(k^2 + \ell^2\right)\pi^2 \tag{6.3}$$

für $k, \ell \in \mathbb{N}$. Zur Diskretisierung mit Finiten Elementen wird (6.1) in die Variationsformulierung

$$\int_\Omega \nabla u \nabla v \,\mathrm{d}x = \lambda \int_\Omega u\, v \,\mathrm{d}x, \tag{6.4}$$

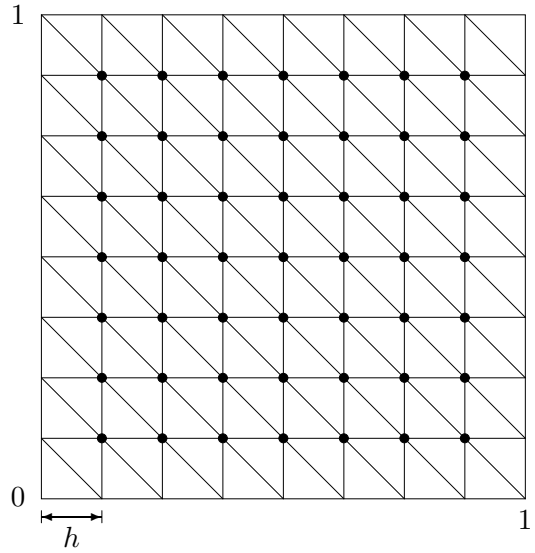


Abbildung 6.1: Reguläre Triangulierung von Ω

mit der Nebenbedingung $u = 0$, $v = 0$ auf Γ, übergeführt. Eine reguläre Triangulierung von Ω mit der Gitterbreite $h = \frac{1}{n+1}$ ergibt, wie in Abbildung 6.1 dargestellt, $N = n^2$ innere Knoten. Stetige stückweise lineare Basisfunktionen auf den Knoten der Dreieckselemente liefern dann das Gleichungssystem

$$A_h \underline{u} = \lambda M_h \underline{u}, \tag{6.5}$$

mit der Steifigkeitsmatrix A_h und der Massematrix M_h. Die symmetrischen und positiv definiten Matrizen $A_h, M_h \in \mathbb{R}^{N \times N}$ sind von folgender Gestalt:

$$A_h = \begin{pmatrix} B & -I & & \\ -I & \ddots & \ddots & \\ & \ddots & \ddots & -I \\ & & -I & B \end{pmatrix} \quad \text{mit} \quad B = \begin{pmatrix} 4 & -1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 4 \end{pmatrix}$$

und

$$M_h = \frac{h^2}{12} \begin{pmatrix} C & D_- & & \\ D_+ & \ddots & \ddots & \\ & \ddots & \ddots & D_- \\ & & D_+ & C \end{pmatrix} \quad \text{mit} \quad C = \begin{pmatrix} 6 & 1 & & \\ 1 & \ddots & \ddots & \\ & \ddots & \ddots & 1 \\ & & 1 & 6 \end{pmatrix},$$

$$D_+ = \begin{pmatrix} 1 & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ & & & 1 \end{pmatrix} \quad \text{und} \quad D_- = \begin{pmatrix} 1 & & & \\ 1 & \ddots & & \\ & \ddots & \ddots & \\ & & 1 & 1 \end{pmatrix},$$

wobei $B, C, D_+, D_- \in \mathbb{R}^{n \times n}$.

6.1.1 Approximation der Eigenpaare von $M_h^{-1}A_h$

Da diese später für Fehlerabschätzungen benötigt werden, sollen nun die Eigenwerte und Eigenvektoren von $M_h^{-1}A_h$ berechnet werden, wofür im Vorfeld zur besseren Analyse die Lösung des verallgemeinerten Eigenwertproblems

$$A_h \underline{\tilde{u}}_{k\ell} = \tilde{\lambda}_{k\ell} M_h \underline{\tilde{u}}_{k\ell}, \quad 1 \leqslant k, \ell \leqslant n, \tag{6.6}$$

näherungsweise bestimmt wird. Hierzu sind zunächst die Eigenwerte von A_h zu ermitteln, für die gilt, dass

$$\lambda_{k\ell} = 4\left(1 - \frac{1}{2}\cos\alpha_k - \frac{1}{2}\cos\alpha_\ell\right) = 4\left(\sin^2\frac{\alpha_k}{2} + \sin^2\frac{\alpha_\ell}{2}\right),$$

mit $\alpha_k = \frac{k\pi}{n+1}$, $\alpha_\ell = \frac{\ell\pi}{n+1}$ für $1 \leqslant k, \ell \leqslant n$. Die zugehörigen normierten Eigenvektoren lauten dann

$$\underline{v}_{k\ell} = \left(\frac{2}{n+1}\sin(i\alpha_k)\sin(j\alpha_\ell)\right)_{1\leqslant i,j\leqslant n}.$$

Die Eigenwerte von $M_h^{-1}A_h$ ergeben sich nun, unter Vernachlässigung von Termen der Ordnung $\mathcal{O}\left(f(k,\ell)\,h^2\right)$, näherungsweise zu

$$\begin{aligned}\tilde{\lambda}_{k\ell} &= \frac{12}{h^2}\frac{2 - \cos\alpha_k - \cos\alpha_\ell}{3 + \cos\alpha_k + \cos\alpha_\ell + \cos\alpha_k\cos\alpha_\ell} \\ &= \frac{1}{h^2}\lambda_{k\ell}\left(1 + \mathcal{O}\left(f(k,\ell)\,h^2\right)\right)\end{aligned} \tag{6.7}$$

mit Eigenvektoren

$$\underline{\tilde{u}}_{k\ell} = \underline{v}_{k\ell} + \mathcal{O}\left(\underline{f}(k,\ell)\,h^2\right).$$

f und $\underline{f}$ bezeichnen hier eine reellwertige bzw. eine vektorwertige Funktion von k und ℓ. Für den approximierten größten Eigenwert $\tilde{\lambda}_N$ von $M_h^{-1}A_h$ für $h \to 0$ gilt dann nach (6.7)

$$\tilde{\lambda}_N = \frac{12\cdot(2+1+1)}{3-1-1+1}\,N = 24N. \tag{6.8}$$

Weiterhin genügen die Eigenwerte $\tilde{\lambda}_{k\ell}$ von $M_h^{-1}A_h$

$$\tilde{\lambda}_{k\ell} \overset{h\to 0}{\longrightarrow} \left(k^2 + \ell^2\right)\pi^2 \tag{6.9}$$

für $k, \ell \geqslant 1$, also konvergieren die meisten Eigenwerte von $M_h^{-1}A_h$ für $h \to 0$ gegen mehrfache Eigenwerte von $-\Delta$.

6.1.2 Algorithmus zur Berechnung der Eigenpaare von $M_h^{-1}A_h$

Wegen (6.9) kann für die Berechnung der Eigenpaare ein so genanntes simultanes Iterationsverfahren zum Einsatz kommen. Hierfür wird nach jeder Iteration zur Orthogonalisierung eine Verallgemeinerung des Rayleigh-Ritz-Verfahren zur Approximation von Eigenvektoren von hermiteschen Matrizen, ein so genannter Schur-Rayleigh-Ritz-Schritt (SRR-Schritt) [18], eingebaut.

Seien also im weiteren m die Vielfachheit des gesuchten Eigenwerts von $M_h^{-1}A_h$ für $h \to 0$ und eine Startmatrix $Q^{(0)} = \left[\underline{q}_1, \ldots, \underline{q}_m\right]$, wobei die orthogonalen Spalten $\underline{q}_j^{(0)} = P\underline{e}_j$ von $Q^{(0)}$ aus dem j-ten Einheitsvektor und einer geeigneten Permutationsmatrix P (siehe Abschnitt 6.3) gebildet werden, gegeben, dann lautet der Algorithmus zur Berechnung der Eigenwerte und Eigenvektoren von $M_h^{-1}A_h$ wie folgt [12, 13]:

Algorithmus 6.1 (Berechnung von Eigenpaaren von $M_h^{-1}A_h$):

Bestimmung des Shifts μ
$Q^{(0)} = P\left[\underline{e}_1, \ldots, \underline{e}_m\right]$
Mindestanzahl $k_{\min}$ der Iterationen
Wähle Toleranz ε
$k = 0$
while $\left(\frac{e_j^{(k)}}{e_j^{(k+1)}} > 1 + \varepsilon \text{ für ein } j \in \{1, \ldots, m\} \text{ or } k < k_{\min}\right)$
  $k = k + 1$
  Löse das System $\left(\mu I - M_h^{-1}A_h\right) Z^{(k)} = Q^{(k-1)}$
  Ermittle QR-Zerlegung von $Z^{(k)}$: $Z^{(k)} = Q^{(k)}R^{(k)}$
  SRR-Schritt:
  Löse das System $\left(\mu I - M_h^{-1}A_h\right) C^{(k)} = Q^{(k)}$
  $B^{(k)} = Q^{(k)\top}C^{(k)}$
  Ermittle Schur-Zerlegung von $B^{(k)}$: $T^{(k)} = Y^{(k)\top}B^{(k)}Y^{(k)}$
  $Q^{(k)} = Q^{(k)}Y^{(k)}$
  Rayleigh-Quotienten und relative Rückwärtsfehler:
  for $j = 1, \ldots, m$
    $\lambda_j^{(k)} = \underline{q}_j^{(k)\top} M_h^{-1}A_h\underline{q}_j^{(k)}$
    $e_j^{(k)} = \frac{\left\|M_h^{-1}A_h\underline{q}_j^{(k)} - \lambda_j^{(k)}\underline{q}_j^{(k)}\right\|_2}{\left\|M_h^{-1}A_h\right\|_2}$
  end
end
Lineare Ausgleichsprobleme und Rayleigh-Quotienten nach ℓ Iterationen:
for $j = 1, \ldots, m$
  $$\left\| M_h^{-1}A_h\left(\underline{q}_j^{(\ell)} + \sum_{\substack{i=1\\ i\neq j}}^{m} \alpha_i \underline{q}_i^{(\ell)}\right) - \lambda_j^{(\ell)}\left(\underline{q}_j^{(\ell)} + \sum_{\substack{i=1\\ i\neq j}}^{m} \alpha_i \underline{q}_i^{(\ell)}\right)\right\|_2 = \min_{\alpha_i}$$
  $\underline{q}_j^{(\ell)} = \underline{q}_j^{(\ell)} + \sum_{\substack{i=1\\ i\neq j}}^{m} \alpha_i \underline{q}_i$
  $\underline{q}_j^{(\ell)} = \frac{\underline{q}_j^{(\ell)}}{\left\|\underline{q}_j^{(\ell)}\right\|_2}$
  $\lambda_j^{(\ell)} = \underline{q}_j^{(\ell)\top} M_h^{-1}A_h\underline{q}_j^{(\ell)}$
end

Im folgenden werden nun die einzelnen Schritte des klassischen Algorithmus mitsamt dem benötigten Aufwand erläutert.
Um festzulegen welche Eigenpaare ermittelt werden sollen, kann man mit der Näherung (6.7)

für die Eigenwerte von $M_h^{-1}A_h$ den Shift

$$\mu_{k\ell} = \frac{12}{h^2}\frac{2 - \cos\alpha_k - \cos\alpha_\ell}{3 + \cos\alpha_k + \cos\alpha_\ell + \cos\alpha_k\cos\alpha_\ell}, \tag{6.10}$$

wie in Zeile `1` geschehen, wählen.
Die linearen Gleichungssysteme aus den Zeilen `8` und `11` lassen sich umschreiben zu

$$(\mu M_h - A_h)\, Z^{(k)} = M_h Q^{(k-1)} \quad \text{bzw.} \quad (\mu M_h - A_h)\, C^{(k)} = M_h Q^{(k)},$$

die sich mit Hilfe einer $LDL^\top$-Zerlegung der symmetrischen Matrix $(\mu M_h - A_h)$, welche nur genau einmal im voraus gebildet werden muss, lösen lassen. Im Algorithmus sind dann nur noch Vorwärts- und Rückwärtssubstitutionen auszuführen. Der Aufwand hierfür liegt dann in $\mathcal{O}(N^3)$.
Die Matrix-Vektor-Produkte $M_h^{-1}A_h\underline{q}_j^{(k)}$ in den Zeilen `17` und `26` benötigen nach einmal erfolgter Cholesky-Zerlegung von M_h nur jeweils eine Matrix-Vektor-Multiplikation und eine Vorwärts- und Rückwärtssubstitution, was also einen Aufwand von $\mathcal{O}(N^3)$ Flops ergibt.
In Zeile `14` wird $Q^{(k)}$ nach einem SRR-Schritt durch $Q^{(k)}Y^{(k)}$ ersetzt, da die Spalten von letzterer Matrix im Allgemeinen die gesuchten Eigenvektoren besser approximieren.
Zur Lösung der linearen Ausgleichsprobleme in den Zeilen `21`–`27` wird eine QR-Zerlegung der $N \times (n-1)$-Matrix $\left(M_h^{-1}A_h - \lambda_j^{(\ell)} I\right) Q_{(j)}^{(\ell)}$ (vgl. Abschnitt 6.3) benötigt, die sich mit einem Aufwand von $\mathcal{O}(n^2 N)$ Operationen ermitteln lässt.
Zusammengefasst beläuft sich der Gesamtaufwand für Algorithmus 6.1 in seiner klassischen Form also auf $\mathcal{O}(N^3)$ Flops.

6.2 Matrix-Faktorisierungen in $\mathcal{H}$-Arithmetik

Wie im vorangegangenen Abschnitt gesehen, werden einige Verfahren zur Zerlegung von Matrizen $A \in \mathbb{R}^{N\times N}$ benötigt, um die Lösung von linearen Gleichungssystemen nach einmal erfolgter Faktorisierung auf Vor- und Rückwärtssubstitutionen zurückzuführen. Durch die Verwendung von $\mathcal{H}$-Arithmetik wird sich der Aufwand für Algorithmus 6.1 erheblich verringern.

6.2.1 Lösen von Block-Dreiecks-Systemen in $\mathcal{H}$-Arithmetik

Für die nachfolgenden Cholesky- und $LDL^\top$-Zerlegungen sowie die Lösung des $\mathcal{H}$-Eigenwertproblems des Laplace-Operators ist es nötig, Blockmatrix-Systeme zu lösen, wobei die exakten Operationen im Gaußschen Algorithmus auf 4×4-Blöcken durch ihre hierarchischen Approximationen ersetzt werden. Die 4×4-Blockstruktur entsteht dadurch, dass das Gitternetz Ω in vier gleichgroße Teile Ω_i, $i = 1, \ldots, 4$, zerlegt wird und die erhaltenen Teilgebiete in folgender Weise nummeriert werden:

Ω_1	Ω_2
Ω_4	Ω_3

Dadurch entstehen in den entsprechenden Matrizen 16 Unterblöcke, welche in drei Typen $\times$, $+$ und $\square$, die in [8] genauer beschrieben sind, eingeteilt werden, die die Nachbarschaftsbeziehenung der entstandenen Teilgebiete beschreiben. Diese Partitionierung wird nun für jedes der vier Teilgebiete wiederholt, so dass eine rekursive Struktur entsteht, die fortgeführt wird solange die entstehenden Unterblöcke mindestens einen Knoten enthalten. Dadurch ergibt sich ein Rekursionsbaum der Tiefe p mit $N = 4^p$.
Zunächst sollen lineare Gleichungssysteme mit hierarchischen unteren Block-Dreiecksmatrizen

$$L = \begin{array}{|c|c|c|c|}\hline L_{11} & & & \\ \hline L_{21} & L_{22} & & \\ \hline L_{31} & L_{32} & L_{33} & \\ \hline L_{41} & L_{42} & L_{43} & L_{44} \\ \hline \end{array},$$

wobei $L_{11}, \ldots, L_{44}$ wiederum untere Block-Dreiecksmatrizen sind, betrachtet werden. Das hierarchische lineare Matrix-Gleichungssystem mit hierarchischen 4×4-Blöcken lautet dann

$$LR_1 = R_2 \quad \text{bzw.} \quad L^\top R_1 = R_2$$

mit Rk-Matrizen, also Matrizen vom Rang k, R_1 und R_2. Somit lassen sich R_1 und R_2 als $R_1 = \underline{x}\underline{y}^\top$ bzw. $R_2 = \underline{a}\,\underline{b}^\top$ schreiben, wobei $\underline{x}, \underline{y}, \underline{a}, \underline{b} \in \mathbb{R}^{N \times k}$, und es ergibt sich für den ersten Fall $L\underline{x}\underline{y}^\top = \underline{a}\,\underline{b}^\top$. Es gilt also $\underline{y} = \underline{b}$ zu setzen und $L\underline{x} = \underline{a}$ nach $\underline{x}$ aufzulösen:

$$\begin{array}{|c|c|c|c|}\hline L_{11} & & & \\ \hline L_{21} & L_{22} & & \\ \hline L_{31} & L_{32} & L_{33} & \\ \hline L_{41} & L_{42} & L_{43} & L_{44} \\ \hline \end{array} \cdot \begin{array}{|c|}\hline \underline{x}_1 \\ \hline \underline{x}_2 \\ \hline \underline{x}_3 \\ \hline \underline{x}_4 \\ \hline \end{array} = \begin{array}{|c|}\hline \underline{a}_1 \\ \hline \underline{a}_2 \\ \hline \underline{a}_3 \\ \hline \underline{a}_4 \\ \hline \end{array} \iff \begin{aligned} L_{11}\underline{x}_1 &= \underline{a}_1 \\ L_{22}\underline{x}_2 &= \underline{a}_2 - L_{21}\underline{x}_1 \\ L_{33}\underline{x}_3 &= \underline{a}_3 - L_{31}\underline{x}_1 - L_{32}\underline{x}_2 \\ L_{44}\underline{x}_4 &= \underline{a}_4 - L_{41}\underline{x}_1 - L_{42}\underline{x}_2 - L_{43}\underline{x}_3. \end{aligned}$$

Für obige rekursive Vorwärtssubstitution benötigt man somit 4 Vorwärtssubstitutionen und 6 Matrix-Vektor-Multiplikationen mit Außerdiagonalblöcken auf dem nächsthöheren Level. Die Lösung von $L^\top R_1 = R_2$, also die Rückwärtssubstitution $L^\top \underline{x} = \underline{a}$, erhält man analog.
Da der Aufwand für die hierarchischen Matrix-Vektor-Multiplikationen mit Außerdiagonalblöcken in $\mathcal{O}(N)$ liegt [8], ergeben sich für $N = 4^p$

$$\mathcal{N}_{\text{Rk-LGS}}(p) = 4\,\mathcal{N}_{\text{Rk-LGS}}(p-1) + \mathcal{O}(N) = \mathcal{O}(pN) = \mathcal{O}(N \log N)$$

Flops für das Lösen eines Rk-Systems auf dem Level p.
Im Nachfolgenden werden nun lineare Matrix-Gleichungssysteme mit Blockmatrizen der drei Typen $\times$, $+$ und $\square$ behandelt.
Beginnend mit den $\times$-Systemen gilt es

$$LX_\times = B_\times \quad \text{bzw.} \quad L^\top X_\times = B_\times$$

zu lösen, wobei $\times$ einen der vier Blocktypen $\nearrow$, $\searrow$, $\swarrow$ oder $\nwarrow$ bezeichnet. Stellvertretend soll hier die Vorgehensweise bei der $\mathcal{H}$-Matrix-Vorwärtssubstitution $LX_{\nwarrow} = B_{\nwarrow}$ aufgezeigt werden. Aus dem Matrix-Gleichungssystem

$$\begin{array}{|c|c|c|c|}\hline L_{11} & & & \\ \hline L_{21} & L_{22} & & \\ \hline L_{31} & L_{32} & L_{33} & \\ \hline L_{41} & L_{42} & L_{43} & L_{44} \\ \hline \end{array} \cdot \begin{array}{|c|c|c|c|}\hline X_{11} & X_{12} & \nwarrow & X_{14} \\ \hline X_{21} & X_{22} & X_{23} & X_{24} \\ \hline X_{31} & X_{32} & X_{33} & X_{34} \\ \hline X_{41} & X_{42} & X_{43} & X_{44} \\ \hline \end{array} = \begin{array}{|c|c|c|c|}\hline R_{11} & R_{12} & \nwarrow & R_{14} \\ \hline R_{21} & R_{22} & R_{23} & R_{24} \\ \hline R_{31} & R_{32} & R_{33} & R_{34} \\ \hline R_{41} & R_{42} & R_{43} & R_{44} \\ \hline \end{array}$$

ergibt sich dann

$$
\begin{aligned}
&L_{11}X_{11} = R_{11}\\
&L_{11}X_{12} = R_{12}\\
&L_{11}\nwarrow\ = \ \nwarrow\\
&L_{11}X_{14} = R_{14}\\
&L_{22}X_{21} = R_{21} - L_{21}X_{11}\\
&L_{22}X_{22} = R_{22} - L_{21}X_{12}\\
&L_{22}X_{23} = R_{23} - L_{21}\nwarrow\\
&L_{22}X_{24} = R_{24} - L_{21}X_{14}\\
&L_{33}X_{31} = R_{31} - L_{31}X_{11} - L_{32}X_{21}\\
&L_{33}X_{32} = R_{32} - L_{31}X_{12} - L_{32}X_{22}\\
&L_{33}X_{33} = R_{33} - L_{31}\nwarrow\ - L_{32}X_{23}\\
&L_{33}X_{34} = R_{34} - L_{31}X_{14} - L_{32}X_{24}\\
&L_{44}X_{41} = R_{41} - L_{41}X_{11} - L_{42}X_{21} - L_{43}X_{31}\\
&L_{44}X_{42} = R_{42} - L_{41}X_{12} - L_{42}X_{22} - L_{43}X_{32}\\
&L_{44}X_{43} = R_{43} - L_{41}\nwarrow\ - L_{42}X_{23} - L_{43}X_{33}\\
&L_{44}X_{44} = R_{44} - L_{41}X_{14} - L_{42}X_{24} - L_{43}X_{34}.
\end{aligned}
$$

Man erhält also 15 Rk- und ein $\nwarrow$-System auf dem nächsthöheren Level. Die Lösung von $L^\top R_\times = R_\times$ erfolgt wiederum analog. Für die $\times$-Systeme ergibt sich dann ein Aufwand von

$$
\begin{aligned}
\mathcal{N}_{\times\text{-LGS}}(p) &= \mathcal{N}_{\times\text{-LGS}}(p-1) + 15\mathcal{N}_{\text{Rk-LGS}}(p-1) + \mathcal{O}(N)\\
&= \mathcal{N}_{\times\text{-LGS}}(p-1) + CpN + \mathcal{O}(N) = \frac{4}{3}CpN + \mathcal{O}(N)\\
&= \mathcal{O}(pN) = \mathcal{O}(N\log N)
\end{aligned}
$$

Flops.
Als nächstes erfolgt nun die Behandlung der linearen Gleichungssysteme des +-Typs, also

$$LX_+ = B_+ \quad \text{bzw.} \quad L^\top X_+ = B_+,$$

mit den Block-Formaten $\uparrow$, $\rightarrow$, $\downarrow$ und $\leftarrow$. Wie oben ist das Verfahren wiederum nur für eines der Formate aufgeführt, hier $LX_\leftarrow = B_\leftarrow$. Das System

$$
\begin{array}{|c|c|c|c|}\hline L_{11} & & & \\ \hline L_{21} & L_{22} & & \\ \hline L_{31} & L_{32} & L_{33} & \\ \hline L_{41} & L_{42} & L_{43} & L_{44} \\ \hline \end{array}
\cdot
\begin{array}{|c|c|c|c|}\hline X_{11} & \leftarrow & \swarrow & X_{14} \\ \hline X_{21} & X_{22} & X_{23} & X_{24} \\ \hline X_{31} & X_{32} & X_{33} & X_{34} \\ \hline X_{41} & \nwarrow & \leftarrow & X_{44} \\ \hline \end{array}
=
\begin{array}{|c|c|c|c|}\hline R_{11} & \leftarrow & \swarrow & R_{14} \\ \hline R_{21} & R_{22} & R_{23} & R_{24} \\ \hline R_{31} & R_{32} & R_{33} & R_{34} \\ \hline R_{41} & \nwarrow & \leftarrow & R_{44} \\ \hline \end{array}
$$

wird dann durch die Rückführung auf 12 Rk-, ein $\nwarrow$-, ein $\swarrow$- und zwei $\leftarrow$-Systeme auf dem nächsthöheren Level mit einem analog zu oben berechneten Aufwand von

$$\mathcal{N}_{+\text{-LGS}}(p) = \mathcal{O}(N\log N)$$

Operationen gelöst.
Mit Hilfe der beiden gerade aufgezeigten Lösungsverfahren können nun die $\square$-Systeme

$$LX_{\square} = B_{\square} \quad \text{bzw.} \quad L^{\top} X_{\square} = B_{\square},$$

und auch hier wiederum exemplarisch nur für ersteres Gleichungssystem

$$\begin{array}{|c|c|c|c|}\hline L_{11} & & & \\ \hline L_{21} & L_{22} & & \\ \hline L_{31} & L_{32} & L_{33} & \\ \hline L_{41} & L_{42} & L_{43} & L_{44} \\ \hline \end{array} \cdot \begin{array}{|c|c|c|c|}\hline \square & \rightarrow & \searrow & \downarrow \\ \hline \leftarrow & \square & \downarrow & \swarrow \\ \hline \nwarrow & \uparrow & \square & \leftarrow \\ \hline \uparrow & \nearrow & \rightarrow & \square \\ \hline \end{array} = \begin{array}{|c|c|c|c|}\hline \square & \rightarrow & \searrow & \downarrow \\ \hline \leftarrow & \square & \downarrow & \swarrow \\ \hline \nwarrow & \uparrow & \square & \leftarrow \\ \hline \uparrow & \nearrow & \rightarrow & \square \\ \hline \end{array},$$

auf 4 $\square$-Systeme, je 2 $\uparrow$-, $\rightarrow$-, $\downarrow$- und $\leftarrow$-Systeme und je ein $\nearrow$-, $\searrow$-, $\swarrow$- und $\nwarrow$-System auf dem nächsthöheren Level zurückgeführt werden, was sich mit einem Aufwand von

$$\begin{aligned} \mathcal{N}_{\square\text{-LGS}}(p) &= 4\mathcal{N}_{\square\text{-LGS}}(p-1) + 8\mathcal{N}_{+\text{-LGS}}(p-1) + 4\mathcal{N}_{\times\text{-LGS}}(p-1) + \mathcal{O}(pN) \\ &= 4\mathcal{N}_{\square\text{-LGS}}(p-1) + \mathcal{O}(pN) = \mathcal{O}(p^2 N) \\ &= \mathcal{O}(N \log^2 N) \end{aligned}$$

Flops bewerkstelligen lässt.

6.2.2 Cholesky-Zerlegung in $\mathcal{H}$-Arithmetik

Für eine symmetrische und positiv definite $\mathcal{H}$-Matrix $A = (A_{ij})_{1 \leqslant i,j \leqslant 4} \in \mathbb{R}^{N \times N}$ ist die Cholesky-Zerlegung $A = L_{\mathcal{H}} L_{\mathcal{H}}^{\top}$ in $\mathcal{H}$-Arithmetik zu berechnen, wobei deren exakte Cholesky-Zerlegung $A = LL^{\top}$ lautet. Es muss also die $\mathcal{H}$-Approximation $L_{\mathcal{H}}$ der unteren Dreiecksmatrix L ermittelt werden.
Seien

$$A = \begin{array}{|c|c|c|c|}\hline A_{11} & A_{21}^{\top} & A_{31}^{\top} & A_{41}^{\top} \\ \hline A_{21} & A_{22} & A_{32}^{\top} & A_{42}^{\top} \\ \hline A_{31} & A_{32} & A_{33} & A_{43}^{\top} \\ \hline A_{41} & A_{42} & A_{43} & A_{44} \\ \hline \end{array} \quad \text{und} \quad L = \begin{array}{|c|c|c|c|}\hline L_{11} & & & \\ \hline L_{21} & L_{22} & & \\ \hline L_{31} & L_{32} & L_{33} & \\ \hline L_{41} & L_{42} & L_{43} & L_{44} \\ \hline \end{array},$$

dann ergeben sich die zu bestimmenden Blöcke von L aus

$$\begin{aligned} L_{11} L_{11}^{\top} &= A_{11} \\ L_{21} L_{11}^{\top} &= A_{21} \\ L_{31} L_{11}^{\top} &= A_{31} \\ L_{41} L_{11}^{\top} &= A_{41} \\ L_{22} L_{22}^{\top} &= A_{22} - L_{21} L_{21}^{\top} \\ L_{32} L_{22}^{\top} &= A_{32} - L_{31} L_{21}^{\top} \\ L_{42} L_{22}^{\top} &= A_{42} - L_{41} L_{21}^{\top} \\ L_{33} L_{33}^{\top} &= A_{33} - L_{31} L_{31}^{\top} - L_{32} L_{32}^{\top} \\ L_{43} L_{33}^{\top} &= A_{43} - L_{41} L_{31}^{\top} - L_{42} L_{32}^{\top} \\ L_{44} L_{44}^{\top} &= A_{44} - L_{41} L_{41}^{\top} - L_{42} L_{42}^{\top} - L_{43} L_{43}^{\top} \end{aligned}$$

rekursiv durch 4 $\mathcal{H}$-Cholesky-Zerlegungen und 6 $\mathcal{H}$-Matrix-Vorwärtssubstitutionen mit Außerdiagonalblöcken als rechte Seiten auf dem nächsthöheren Level.
Nach Abschnitt 6.2.1 benötigt man für eine $\mathcal{H}$-Matrix-Vorwärts- bzw. Rückwärtssubstitution bei einem Außerdiagonalblock $\mathcal{O}(N \log N)$ und bei einem Diagonalblock $\mathcal{O}(N \log^2 N)$ Operationen. Für die $\mathcal{H}$-Cholesky-Zerlegung ergibt sich somit ein Aufwand von

$$\mathcal{N}_{\text{Cholesky}}(p) = \mathcal{O}(N \log^2 N)$$

Flops.

6.2.3 $LDL^\top$-Zerlegung in $\mathcal{H}$-Arithmetik

Die Bestimmung $\mathcal{H}$-$LDL^\top$-Zerlegung einer symmetrischen $\mathcal{H}$-Matrix $A \in \mathbb{R}^{N \times N}$ erfolgt analog zur $\mathcal{H}$-Cholesky-Zerlegung.
Es ist $A = LDL^\top = L \cdot DL^\top$ mit einer unteren Dreiecksmatrix L und einer Diagonalmatrix D, also

$$\begin{array}{|cccc|}\hline A_{11} & A_{21}^\top & A_{31}^\top & A_{41}^\top \\ A_{21} & A_{22} & A_{32}^\top & A_{42}^\top \\ A_{31} & A_{32} & A_{33} & A_{43}^\top \\ A_{41} & A_{42} & A_{43} & A_{44} \\ \hline \end{array}$$

$$= \begin{array}{|cccc|}\hline L_{11} & & & \\ L_{21} & L_{22} & & \\ L_{31} & L_{32} & L_{33} & \\ L_{41} & L_{42} & L_{43} & L_{44} \\ \hline \end{array} \cdot \begin{array}{|cccc|}\hline D_{11} & & & \\ & D_{22} & & \\ & & D_{33} & \\ & & & D_{44} \\ \hline \end{array} \cdot \begin{array}{|cccc|}\hline L_{11}^\top & L_{21}^\top & L_{31}^\top & L_{41}^\top \\ & L_{22}^\top & L_{32}^\top & L_{42}^\top \\ & & L_{33}^\top & L_{43}^\top \\ & & & L_{44}^\top \\ \hline \end{array}$$

$$= \begin{array}{|cccc|}\hline L_{11} & & & \\ L_{21} & L_{22} & & \\ L_{31} & L_{32} & L_{33} & \\ L_{41} & L_{42} & L_{43} & L_{44} \\ \hline \end{array} \cdot \begin{array}{|cccc|}\hline D_{11}L_{11}^\top & D_{11}L_{21}^\top & D_{11}L_{31}^\top & D_{11}L_{41}^\top \\ & D_{22}L_{22}^\top & D_{22}L_{32}^\top & D_{22}L_{42}^\top \\ & & D_{33}L_{33}^\top & D_{33}L_{43}^\top \\ & & & D_{44}L_{44}^\top \\ \hline \end{array},$$

und somit

$$\begin{aligned}
L_{11}D_{11}L_{11}^\top &= A_{11} && \rightarrow \quad D_{11},\ L_{11} \\
L_{21}D_{11}L_{11}^\top &= A_{21} && \rightarrow \quad L_{21} \\
L_{31}D_{11}L_{11}^\top &= A_{31} && \rightarrow \quad L_{31} \\
L_{41}D_{11}L_{11}^\top &= A_{41} && \rightarrow \quad L_{41} \\
L_{22}D_{22}L_{22}^\top &= A_{22} - L_{21}D_{11}L_{21}^\top && \rightarrow \quad D_{22},\ L_{22} \\
L_{32}D_{22}L_{22}^\top &= A_{32} - L_{31}D_{11}L_{21}^\top && \rightarrow \quad L_{32} \\
L_{42}D_{22}L_{22}^\top &= A_{42} - L_{41}D_{11}L_{21}^\top && \rightarrow \quad L_{42} \\
L_{33}D_{33}L_{33}^\top &= A_{33} - L_{31}D_{11}L_{31}^\top - L_{32}D_{22}L_{32}^\top && \rightarrow \quad D_{33},\ L_{33} \\
L_{43}D_{33}L_{33}^\top &= A_{43} - L_{41}D_{11}L_{31}^\top - L_{42}D_{22}L_{32}^\top && \rightarrow \quad L_{43} \\
L_{44}D_{44}L_{44}^\top &= A_{44} - L_{41}D_{11}L_{41}^\top - L_{42}D_{22}L_{42}^\top - L_{43}D_{33}L_{43}^\top && \rightarrow \quad D_{44},\ L_{44}.
\end{aligned}$$

In völliger Analogie zur $\mathcal{H}$-Cholesky-Zerlegung erhält man für den Aufwand der $\mathcal{H}$-$LDL^\top$-Zerlegung

$$\mathcal{N}_{LDL^\top}(p) = \mathcal{O}(N \log^2 N)$$

Operationen.

6.2.4 QR-Zerlegung in $\mathcal{H}$-Arithmetik

Für die $\mathcal{H}$-QR-Zerlegung $A = QR$ einer nichtsingulären $\mathcal{H}$-Matrix $A \in \mathbb{R}^{N\times N}$ kann die bereits vorgestellte $\mathcal{H}$-Cholesky-Zerlegung verwendet werden, denn mit der Cholesky-Zerlegung $A^\top A = LL^\top$ von $A^\top A$ gilt $R = L^\top$. Diese Beobachtung führt auf den folgenden Algorithmus zur Berechnung der $\mathcal{H}$-Approximationen $Q_\mathcal{H}$ und $R_\mathcal{H}$ von Q bzw. R:

Algorithmus 6.2 (Berechnung QR-Zerlegung in $\mathcal{H}$-Arithmetik):

1 Berechne des Matrixprodukts $B = A^\top A$ in $\mathcal{H}$-Arithmetik
2 Bestimme die $\mathcal{H}$-Cholesky-Zerlegung $B = L_\mathcal{H} L_\mathcal{H}^\top$ von $A^\top A$
3 $R_\mathcal{H} = L_\mathcal{H}^\top$
4 Löse $Q_\mathcal{H} R_\mathcal{H} = A$ durch $\mathcal{H}$-Matrix-Vorwärtssubstitution nach $Q_\mathcal{H}$

Für obigen Algorithmus ist eine $\mathcal{H}$-Matrix-Multiplikation, eine $\mathcal{H}$-Cholesky-Zerlegung und eine $\mathcal{H}$-Matrix-Vorwärtssubstitution nötig, woraus sich ergibt, dass der Aufwand für den Algorithmus also $\mathcal{O}(N \log^2 N)$ Operationen beträgt.

6.2.5 Zusammenfassung

In diesem Abschnitt sind zusammenfassend die Vorteile des Einsatzes der $\mathcal{H}$-Arithmetik aufgeführt. Tabelle 6.1 beinhaltet einen Vergleich der in Algorithmus 6.1 benötigten klassischen Matrix-Operationen mit ihren hierarchischen Pendants.

	klassische Form	$\mathcal{H}$-Arithmetik
Matrix-Vektor-Multiplikation	$\mathcal{O}(N^2)$	$\mathcal{O}(N \log N)$
Vor- und Rückwärtssubstitutionen	$\mathcal{O}(N^2)$	$\mathcal{O}(N \log^2 N)$
$LDL^\top$-Zerlegung	$\mathcal{O}(N^3)$	$\mathcal{O}(N \log^2 N)$
Cholesky-Zerlegung	$\mathcal{O}(N^3)$	$\mathcal{O}(N \log^2 N)$
QR-Zerlegung	$\mathcal{O}(N^3)$	$\mathcal{O}(N \log^2 N)$

Tabelle 6.1: Die Laufzeitvorteile durch $\mathcal{H}$-Arithmetik

6.3 Das Eigenwertproblem in $\mathcal{H}$-Arithmetik

Algorithmus 6.1 soll nun unter Verwendung von hierarchischen Matrizen beschleunigt werden. Hierfür wurden die benötigten Matrix-Faktorisierungen in $\mathcal{H}$-Arithmetik entwickelt.
Zuerst einmal ist zu beachten, dass sich die klassischen zeilenweise von links unten nach rechts oben durchnummerierten FE-Matrizen durch Konjugation mit einer geeigneten Permutationsmatrix P in hierarchische Matrizen überführen lassen [12, 13]. Sei also $C_\mathcal{H}$ eine hierarchische Matrix und C eine Matrix in der klassischen Nummerierung, dann gilt $C_\mathcal{H} = PCP^\top$.

Die im Anschluss an Algorithmus 6.1 erfolgten Aufwandsbetrachtungen werden nun noch einmal unter Verwendung von $\mathcal{H}$-Arithmetik aufgeführt.
Der Aufwand für die einmalig zu erfolgende $LDL^\top$-Zerlegung der symmetrischen Matrix $(\mu M_h - A_h)$ in den beiden linearen Gleichungssystemen

$$(\mu M_h - A_h)\, Z^{(k)} = M_h Q^{(k-1)} \quad \text{bzw.} \quad (\mu M_h - A_h)\, C^{(k)} = M_h Q^{(k)}$$

reduziert sich unter Verwendung von $\mathcal{H}$-Arithmetik auf $\mathcal{O}(N \log^2 N)$ Flops. Die in jeder Iteration auszuführenden Vorwärts- und Rückwärtssubstitutionen benötigen dann $\mathcal{O}(N \log N)$ Operationen.
Die einmal mit $\mathcal{O}(N \log^2 N)$ Operationen zu ermittelnde $\mathcal{H}$-Cholesky-Zerlegung von M_h reduziert die Matrix-Vektor-Produkte $M_h^{-1} A_h \underline{q}_j^{(k)}$, $1 \leqslant j \leqslant m$, $1 \leqslant j \leqslant \ell$, auf eine Matrix-Vektor-Multiplikation und je eine Vorwärts- und Rückwärtssubstitution, deren Aufwand jeweils in $\mathcal{O}(N \log N)$ liegt.
Die nach ℓ Iterationen ermittelten Spalten $\underline{q}_j^{(\ell)}$ für $j = 1, \ldots, m$ von $Q^{(\ell)}$ spannen den invarianten m-dimensionalen Unterraum von $\mathbb{R}^N$ zu den Eigenwerten $\lambda_j^{(\ell)}$, $j = 1, \ldots, m$, auf. Hierbei ist im Allgemeinen nur die erste Spalte $\underline{q}_1^{(\ell)}$ ein Eigenvektor zum Eigenwert $\lambda_1^{(\ell)}$. Daher werden in den Zeilen 21–27 zur Verbesserung der restlichen Eigenvektoren m lineare Ausgleichsprobleme gelöst. Zeile 23 lässt sich hierfür umformen zu

$$\left\| \left(M_h^{-1} A_h - \lambda_j^{(\ell)} I \right) Q_{(j)}^{(\ell)} \alpha_{(j)} - \left(\lambda_j^{(\ell)} I - M_h^{-1} A_h \right) \underline{q}_j^{(\ell)} \right\|_2 = \min_{\alpha_i}, \tag{6.11}$$

wobei $Q_{(j)}^{(\ell)} = \left[\underline{q}_1^{(\ell)}, \ldots, \underline{q}_{j-1}^{(\ell)}, \underline{q}_{j+1}^{(\ell)}, \ldots, \underline{q}_m^{(\ell)} \right]$ ist. Da die $\mathcal{H}$-Cholesky-Zerlegung von M_h schon in den Zeilen 15–20 ermittelt werden musste, lassen sich mit der QR-Zerlegung der $N \times (n-1)$-Matrix $\left(M_h^{-1} A_h - \lambda_j^{(\ell)} I \right) Q_{(j)}^{(\ell)}$ die gesuchten $\alpha_{(j)} = (\alpha_i)_{1 \leqslant i \leqslant m,\, i \neq j} \in \mathbb{R}^{m-1}$, so dass (6.11) erfüllt ist, mit einem Aufwand von nur noch $\mathcal{O}(N \log N)$ Operationen bestimmen.
Insgesamt liegt der Aufwand für Algorithmus 6.1 bei $\mathcal{O}(N \log^2 N)$ Operationen und wurde somit von einer kubischen Komplexität für den Algorithmus mit klassischen Matrix-Operationen auf fast linearen, also linear bis auf logarithmische Faktoren, Aufwand drastisch reduziert.

6.3.1 Rückwärtsfehleranalyse des $\mathcal{H}$-Eigenwertproblems

Es sei $(\lambda, \underline{u})$ das fehlerbehaftete Eigenpaar von $M_h^{-1} A_h$ aus Algorithmus 6.1 mit $\|\underline{u}\|_2 = 1$. Dies führt zum Residuum $\underline{r} = \lambda \underline{u} - M_h^{-1} A_h \underline{u}$.
Für die Rang-1-Matrix $\Delta\big(M_h^{-1} A_h\big) = \underline{r}\, \underline{u}^\top$ gilt dann

$$\big(M_h^{-1} A_h + \Delta\big(M_h^{-1} A_h\big)\big)\, \underline{u} = \lambda \underline{u}$$

und für den Rückwärtsfehler $\Delta\big(M_h^{-1} A_h\big)$

$$\big\|\Delta\big(M_h^{-1} A_h\big)\big\|_2 = \max_{\|\underline{w}\|_2 = 1} \big\|\Delta\big(M_h^{-1} A_h\big)\, \underline{w}\big\|_2 = \max_{\|\underline{w}\|_2 = 1} \left\| \underline{r} \underline{u}^\top \underline{w} \right\|_2 = \|\underline{r}\|_2.$$

Der relative Rückwärtsfehler ergibt sich nun zu

$$\frac{\big\|\Delta\big(M_h^{-1} A_h\big)\big\|_2}{\big\|M_h^{-1} A_h\big\|_2} = \frac{\|\underline{r}\|_2}{\|M_h^{-1} A_h\|_2} \tag{6.12}$$

und man erhält das Abbruchkriterium

$$\frac{e_j^{(k)}}{e_j^{(k+1)}} \leqslant 1 + \varepsilon \text{ für alle } j \in \{1, \ldots, m\}$$

mit $\varepsilon > 0$ und

$$e_j^{(k)} = \frac{\left\| M_h^{-1} A_h \underline{q}_j^{(k)} - \lambda_j^{(k)} \underline{q}_j^{(k)} \right\|_2}{\left\| M_h^{-1} A_h \right\|_2},$$

wobei $\left\| M_h^{-1} A_h \right\|_2$ durch $24N$ aus (6.8) approximiert wird.

6.4 Zusammenfassung

Dieses Kapitel stellte einen Algorithmus zur simultanen iterativen Approximationen von Eigenwerten und Eigenvektoren vor. Die hierfür benötigten Operationen mit den auftretenden Matrizen, insbesondere deren Faktorisierungen, wurden zunächst auf klassische Weise ausgeführt, woraus ein Gesamtaufwand von $\mathcal{O}(N^3)$ Operationen für den gesamten Algorithmus resultiert. Die anschließende Entwicklung der Matrix-Zerlegungen in $\mathcal{H}$-Arithmetik brachte erhebliche Einsparungen, der Aufwand reduzierte sich auf fast lineare Komplexität.

Kapitel 7

$\mathcal{H}$-Matrizen in der FEM

Alexander Weiß

7.1 Einleitung

Wie wir im vorherigen Kapitel gesehen haben, lässt sich die Finite-Element-Steifigkeitsmatrix A wegen ihrer Bandstruktur auf einfache Weise durch eine $\mathcal{H}$-Matrix darstellen.
Mit dem in Kapitel 3 vorgestellten Algorithmus zur Invertierung einer $\mathcal{H}$-Matrix kann somit auf einfache Weise eine Näherungslösung des Gleichungssystems $Ax = b$ bestimmt werden. Jedoch kann man keine Aussagen machen, wie groß der maximale Rang sein muss um eine vorgegebene Genauigkeit zu erreichen.
In diesem Vortrag werden wir zeigen, dass zu einer gegebenen Genauigkeit stets eine $\mathcal{H}$-Matrix-Inverse mit beschränktem maximalen Rang existiert.

7.2 Voraussetzungen und Definitionen

Wir betrachten zunächst den Laplace-Operator in einem Gebiet $\Omega \subset \mathbb{R}^d$ mit Dirichlet-Randbedingungen:

$$Lu = -\triangle u = f \quad \text{in } \Omega\,, \tag{7.1}$$

$$u = 0 \quad \text{auf } \partial\Omega\,. \tag{7.2}$$

Wir werden uns auf den Fall $d = 3$ beschränken, andere Dimensionen lassen sich jedoch vollkommen analog behandeln.
Wir betrachten eine Partitionierung in Teilgebiete Ω_t, Ω_s durch die folgende Zulässigkeitsbedingung:

$$\operatorname{dist}(\Omega_t, \Omega_s) \geq \max\{\operatorname{diam}(\Omega_t), \operatorname{diam}(\Omega_s)\}\,. \tag{ZB}$$

Da wir einerseits die Lösung $u \in H_0^1(\Omega) =: V$ suchen, andererseits jedoch die Näherungsfunktion $u_h \in V_h$ als Vektor $\underline{u} \in \mathbb{R}^n$ darstellen (und hierfür eine Näherungslösung suchen), sind folgende Transferoperatoren hilfreich:

Definition 7.1. Der Operator J und sein adjungierter Operator J^* sind definiert durch

$$J : \mathbb{R}^n \to V_h, \quad \underline{x} \mapsto \sum_{i \in \mathcal{I}} x_i \phi_i \,, \tag{7.3}$$

$$J^* : L^2(\Omega) \to \mathbb{R}^n, \quad g \mapsto ((g, \phi_i)_{L^2})_{i=1\ldots n} \,. \tag{7.4}$$

Wie wir wissen, gilt für quasi-uniforme und formreguläre Triangulierungen, dass es Konstanten $0 < c_{J,1} \leq c_{J,2}$ gibt so dass

$$c_{J,1} \|x\|_h \leq \|Jx\|_{L^2(\Omega)} \leq c_{J,2} \|x\|_h \,. \tag{7.5}$$

Dabei ist $\|\cdot\|_h$ eine durch die Maschenweite h gewichtete Euklidische Vektornorm.
Außerdem gilt für den Träger Ω_i einer Basisfunktion ϕ_i

$$\mathrm{vol}(\Omega_i) \geq c_v h^d \,. \tag{7.6}$$

Zusätzlich fordern wir, dass jedes Dreieck zum Träger von nur einer beschränkten Anzahl von Basisfunktionen gehört:

$$\mathrm{vol}(\Omega_t) \geq c_M \sum_{i \in \mathcal{T}} \mathrm{vol}(\Omega_i) \,. \tag{7.7}$$

Definition 7.2. Die Steifigkeitsmatrix A ist definiert durch

$$A_{ij} := \int_\Omega \nabla\phi_j(x) \nabla\phi_i(x) dx \,.$$

Definition 7.3. Die Massenmatrix $M = (M_{ij})_{i,j \in \mathcal{I}}$ ist definiert durch

$$M_{ij} := \int_\Omega \phi_i(x) \phi_j(x) dx \,.$$

Man bemerkt, dass $M = J^*J$ gilt. Damit folgt sofort

Korollar 7.1. *Für die Massenmatrix M gelten*

$$\mu_{\max} = \|M\|_2 = \|J^*J\|_2 \leq \|J\|^2 \leq c_{J,2}^2 \,, \tag{7.8}$$

$$\mu_{\min} = 1/\left\|M^{-1}\right\|_2 \geq c_{J,1}^2 \,, \tag{7.9}$$

$$\mathrm{cond}_2(M) = \mu_{\max}/\mu_{\min} \leq (c_{J,2}/c_{J,1})^2 \,. \tag{7.10}$$

Durch Nachrechnen erhält man mit diesen Definitionen:

Korollar 7.2. *Seien $\underline{a}, \underline{b} \in \mathbb{R}^n$ und $C \in \mathbb{R}^{n \times n}$. Dann gilt*

$$(M\underline{a}, \underline{b}) = \langle J\underline{a}, J\underline{b} \rangle \,, \tag{7.11}$$

$$(MCM\underline{a}, \underline{b}) = \langle (JCJ^*)J\underline{a}, J\underline{b} \rangle \,. \tag{7.12}$$

Die exakte Lösung ist durch die Green-Funktion G gegeben, es gilt

Satz 7.3. *Der inverse Differentialoperator L^{-1} ist mit Hilfe der Green-Funktion als Integral-Operator darstellbar durch*

$$(L^{-1}f)(x) := \int_\Omega G(x,y) f(y) dy \,.$$

Definition 7.4. Zur Greenfunktion G und dem zugehörigen Integraloperator L^{-1} definiert man die Matrix $B = J^*L^{-1}J$. Die Einträge lauten damit

$$B_{ij} = \int_\Omega \int_\Omega \phi_j(x) G(x,y) \phi_i(y) dxdy\,.$$

Um den Finiten-Element-Fehler abzuschätzen, definieren wir die L^2-Projektion Q_h und die Ritz-Projektion P_h.

Definition 7.5. Die L^2- und Ritz-Projektion sind definiert als

$$Q_h : L^2(\Omega) \to V_h, Q_h := JJ^{-1}J^{-*}J^* = JM^{-1}J^*\,, \tag{7.13}$$

$$P_h : V \to V_h, P_h := JA^{-1}J^*L\,. \tag{7.14}$$

Als Fehler definiert man

$$e_h^Q(u) := \|u - Q_h u\|_{L^2(\Omega)}\,, \tag{7.15}$$

$$e_h^P(u) := \|u - P_h u\|_{L^2(\Omega)}\,. \tag{7.16}$$

Wie man weiß, lässt sich der Finite-Element-Diskretisierungsfehler wie folgt abschätzen:

Satz 7.4. *Für den Fehler* $e_h^P(u) := \|u - P_h u\|_{L^2(\Omega)}$ *gilt:*

$$e_h^P(u) \le \varepsilon_h \|f\|_{L^2(\Omega)}\,,$$

wobei $\varepsilon_h \to 0$ *für* $h \to 0$ *und* $f = Lu$.

7.3 Darstellung der inversen Steifigkeitsmatrix

Wir suchen eine Approximation der Inversen der Steifigkeitsmatrix A, die die diskrete Form des Differentialoperators L darstellt. Es liegt also nahe, dass sich A^{-1} durch die diskrete Form des inversen Operators L^{-1} darstellen lässt, also durch die Green-Matrix B.
Dies gilt bis auf den Approximationsfehler, wie folgendes Lemma zeigt:

Lemma 7.5. *Es gilt:*

$$\left\|MA^{-1}M - B\right\|_2 \le 2c_{J,2}^2 \varepsilon_h\,.$$

Beweis. Seien $\underline{x}, \underline{y} \in \mathbb{R}^n$. Mit $B = J^*L^{-1}J$ und (7.12) gilt

$$\begin{aligned}\left((MA^{-1}M - B)\underline{x}, \underline{y}\right) &= \left((MA^{-1}M - MM^{-1}J^*L^{-1}JM^{-1}M)\underline{x}, \underline{y}\right)\\ &= \left\langle (JA^{-1}J^* - JM^{-1}J^*L^{-1}JM^{-1}J^*)J\underline{x}, J\underline{y}\right\rangle.\end{aligned}$$

Setzt man die Definitionen von P_h, Q_h und $f_h := J\underline{x} \in V_h$, $g_h := J\underline{y} \in V_h$ ein, und verwendet die Projektionseigenschaft $Q_h f_h = f_h$, so erhält man:

$$\begin{aligned}\left((MA^{-1}M - B)\underline{x}, \underline{y}\right) &= \left\langle P_h L^{-1} f_h - Q_h L^{-1} Q_h f_h, g_h\right\rangle\\ &= \left\langle P_h L^{-1} f_h - Q_h L^{-1} f_h, g_h\right\rangle\\ &= \left\langle (P_h L^{-1} f_h - L^{-1} f_h) + (L^{-1} f_h - Q_h L^{-1}) f_h, g_h\right\rangle\\ &\le \left(e_h^P(L^{-1}f_h) + e_h^Q(L^{-1}f_h)\right)\|g_h\|_{L^2(\Omega)}\end{aligned}$$

Wegen $e_h^Q \leq e_h^P$ und Satz 7.4 folgt mit (7.5):

$$\begin{aligned}\left((MA^{-1}M - B)\underline{x}, \underline{y}\right) &\leq 2e_h^P(L^{-1}f_h)\|g_h\|_{L^2(\Omega)} \\ &\leq 2\varepsilon_h\|h_f\|_{L^2(\Omega)}\|g_h\|_{L^2(\Omega)} \\ &\leq 2c_{J,2}^2\varepsilon_h\|x\|\|y\|\,.\end{aligned}$$

Insgesamt folgt also $\left\|MA^{-1}M - B\right\|_2 \leq 2c_{J,2}^2\varepsilon_h$. □

Satz 7.6. *Es gilt:*

$$\left\|A^{-1} - M^{-1}BM^{-1}\right\|_2 \leq C_{J,1}^{-4}c_{J,2}^2\varepsilon_h\,.$$

Beweis. Dies folgt sofort aus (7.8). □

7.4 Inverse Massenmatrix

Wie wir im vorherigen Abschnitt gesehen haben, lässt sich die inverse Steifigkeitsmatrix (näherungsweise) als Produkt von Green-Matrix und inverser Massenmatrix darstellen. Wir möchten nun die $\mathcal{H}$-Matrix-Approximierbarkeit der inversen Massenmatrix nachweisen.
Zunächst benötigen wir eine komponentenweise Abschätzung der inversen Massenmatrix, die aus folgendem Lemma folgt:

Lemma 7.7. *Sei $d_{ij} = \operatorname{dist}(\operatorname{supp}\phi_i, \operatorname{supp}\phi_j)$, h_{min} die minimale Seitenlänge aller Elemente $t \in \mathcal{T}$, $r = \mu_{max}/\mu_{min}$ und $C := \frac{r-1}{2r}$, $q = \frac{\sqrt{r}-1}{\sqrt{r}+1} \in (0,1)$. Dann erfüllen die Einträge der Inversen der Massenmatrix M^{-1} folgende Ungleichung:*

$$\left|(M^{-1})_{ij}\right| \leq C\left\|M^{-1}\right\|_2 q^{d_{ij}/h_{min}}\,.$$

Schließlich benötigen wir noch eine Abschätzung der Spektralnorm durch Untermatrizen. Es gilt:

Lemma 7.8. *Sei P eine Partition und $L = \mathcal{O}(\log n)$ maximale Tiefe des Cluster-Baumes. Dann gibt es eine Konstante C_{sp}, so dass für alle Matrizen $M \in \mathbb{R}^{n\times n}$ gilt*

$$\|M\|_2 \leq C_{sp}\sum_{\ell=0}^{L}\max_{t\times s\in P_\ell}\|M|_{t\times s}\|_2\,.$$

Auf die Beweise verzichten wir hier, man kann sie in [3] nachlesen.
Mit Hilfe dieser Lemmata lässt sich eine $\mathcal{H}$-Matrix-Inverse der Massenmatrix angeben:

Satz 7.9. *Zu beliebigem $\varepsilon > 0$ gibt es $N_{\mathcal{H}} \in \mathcal{H}(P, k_\varepsilon)$, so dass*

$$\left\|M^{-1} - N_{\mathcal{H}}\right\|_2 \leq \varepsilon\left\|M^{-1}\right\|_2$$

und $k_\varepsilon \in \mathcal{O}(\log^d(\frac{L}{\varepsilon}))$.

Beweis. Sei zunächst $k \in \mathbb{N}$ beliebig. Wir definieren $N_{\mathcal{H}}$ explizit durch:

$$N_{\mathcal{H}}|_{t\times s} := \begin{cases} M^{-1}|_{t\times s} & \#t\#s \leq k^2 \\ 0 & \text{sonst}\end{cases}$$

Sei $E := M^{-1} - N_{\mathcal{H}}$ die Fehlermatrix. Nach Lemma 7.8 reicht es aus, die Norm von $E_{t\times s} = M^{-1}_{t\times s}$ für $\#t\#s > k^2$ abzuschätzen.
Sei also $\#t\#s > k^2$. Für $i \in t, j \in s$ suchen wir eine Abschätzung von E_{ij}. Zunächst bemerkt man $\operatorname{diam}(\Omega_t) \geq \tilde{C} \sqrt[d]{\operatorname{vol}(\Omega_t)}$. Aus der Zulässigkeitsbedingung (ZB) und (7.6), (7.7) folgt:

$$\begin{aligned} d_{ij} = \operatorname{dist}(\Omega_i, \Omega_j) &\geq \operatorname{dist}(\Omega_t, \Omega_s) \\ &\geq \operatorname{diam}(\Omega_t) \\ &\geq \tilde{C} \sqrt[d]{\operatorname{vol}(\Omega_t)} \\ &\geq \tilde{C} \sqrt[d]{c_M \sum_{i\in t} \operatorname{vol}(\Omega_i)} \\ &\geq \tilde{C} \sqrt[d]{c_M c_v} h \sqrt[d]{\#t} \end{aligned}$$

Entsprechendes gilt für $\#s$, so dass man insgesamt $d_{ij}/h \geq C' \sqrt[2d]{\#t\#s}$ erhält. Mit Lemma 7.7 folgt damit

$$|E_{ij}| \leq C \left\|M^{-1}\right\|_2 q^{C' \sqrt[2d]{\#t\#s}} .$$

Durch Abschätzung der Spektralnorm erhält man schließlich

$$\|E|_{t\times s}\|_2 \leq \sqrt{\#t\#s} \max_{i\in t, j\in s} |E_{ij}| \leq C\sqrt{\#t\#s} \left\|M^{-1}\right\|_2 q^{C' \sqrt[2d]{\#t\#s}} .$$

Für großes ℓ gilt $C\ell q^{C' \sqrt[d]{\ell}} \leq q^{C'' \sqrt[d]{\ell}}$ und somit nach Lemma 7.8

$$\|E\|_2 \leq (L+1) C_{sp} \left\|M^{-1}\right\| q^{C'' \sqrt[d]{k}} .$$

Für die Wahl von $k = k_\varepsilon$ so, dass $\|E\|_2 \leq \varepsilon \left\|M^{-1}\right\|_2$ ist, gilt also $k_\varepsilon \in \mathcal{O}(\log^d(\frac{L}{\varepsilon}))$. □

7.5 Green-Funktion

Um eine $\mathcal{H}$-Matrix-Darstellung von B zu bestimmen, benötigen wir eine Approximation der Green-Funktion. Da wir den Laplace-Operator betrachten, können wir diese Approximation leicht durch die Taylor-Entwicklung erhalten.
Wir geben dennoch einen allgemeinen Satz an, der für beliebige elliptische Differentialoperatoren mit L^∞-Koeffizienten gültig ist, wir werden später auf den Beweis für den allgemeinen Fall eingehen.

Satz 7.10. *Sei $D_1, D_2 \subset \Omega$, D_2 konvex und*

$$\operatorname{dist}(D_1, D_2) \geq \operatorname{diam}(D_2) > 0 .$$

Zu beliebigem $\varepsilon \in (0,1)$ und geeignetem $k_\varepsilon \sim \mathcal{O}(\log^{d+1}(\frac{1}{\varepsilon}))$ gibt es eine Approximation

$$G_k(x,y) = \sum_{i=1}^{k} u_i(x) v_i(y) ,$$

so dass $k \leq k_\varepsilon$ und für alle $x \in D_1$ gilt:

$$\|G(x,\cdot) - G_k(x,\cdot)\|_{L^2(D_2)} \leq \varepsilon \|G(x,\cdot)\|_{L^2(D)} ,$$

wobei $D := \{y \in \Omega : \operatorname{dist}(y, D_2) \leq \frac{1}{2}\operatorname{diam}(D_2)\}$.

7.6 Green-Matrix

Mit Hilfe der approximierten Green-Funktion lässt sich jetzt eine $\mathcal{H}$-Matrix-Approximation der Green-Matrix konstruieren. Es gilt:

Satz 7.11. *Sei Ω_s konvex für alle $s \in T_{\mathcal{I}}$. Zu $\varepsilon \in (0,1)$ und $k_\varepsilon \in \mathbb{N}$ mit $k_\varepsilon \sim \mathcal{O}(\log^{d+1}(1/\varepsilon))$ wie in Satz 7.10. Dann gibt es zu $k \geq k_\varepsilon$ eine Matrix $B_{\mathcal{H}} \in \mathcal{H}(P,k)$, so dass gilt*

$$\|B - B_{\mathcal{H}}\|_2 \leq \varepsilon \frac{c}{\lambda_{min}} L\,,$$

wobei c nur von κ_C und $\operatorname{diam}(\Omega)$ abhängt und $L = \mathcal{O}(\log n)$ das maximale Level ist.

Beweis. Sei $t \times s \in P$ so, dass (ZB) erfüllt ist. Nach Satz 7.10 mit $D_1 = \Omega_t, D_2 = \Omega_s$ und $D = \{x \in \Omega : \operatorname{dist}(x, \Omega_s) \leq 1/2 \operatorname{diam}(\Omega_s)\}$ gibt es eine Approximierende $G^b = \sum_{i=1}^{k_\varepsilon} u_i^b(x) v_i^b(y)$, so dass

$$\left\| G - G^b \right\|_{L^2(\Omega_t \times \Omega_s)} \leq \varepsilon \|G\|_{L^2(\Omega_t \times D)}\,.$$

Man setzt u_i^b, v_i^b durch 0 fort und definiert die Matrizen B^b und $B_{\mathcal{H}}$ als

$$(B^b)_{ij} := \int_\Omega \int_\Omega \phi_j(x) G^b(x,y) \phi_i(y) dx dy \tag{7.17}$$

$$B_{\mathcal{H}}|_b := \begin{cases} B^b|_b & \text{falls (ZB) erfüllt} \\ B|_b & \text{sonst}\,. \end{cases} \tag{7.18}$$

Man sieht, dass der Rang von $B_{\mathcal{H}}$ durch k_ε beschränkt ist, da jeder Term $u_i^b(x) v_i^b(y)$ eine Rang-1-Matrix erzeugt.
Sei $E := B - B_{\mathcal{H}}$ die Fehlermatrix. Nach Lemma 7.8 reicht es aus, die Norm von $E|_b = B|_b - B^b|_b$ für denn Fall, dass (ZB) erfüllt ist, abzuschätzen. Nach der Definition von B und B^b gilt für Vektoren $\underline{x} = (x_j)_{j \in s}, \underline{y} = (y_i)_{i \in t}$

$$\begin{aligned} \left|((B|_b - B_{\mathcal{H}}|_b)\underline{y}, \underline{x})\right| &\leq \left\| G - G^b \right\|_{L^2(\Omega_t \times \Omega_s)} \|J_t y\|_{L^2(\Omega_s)} \|J_s x\|_{L^2(\Omega_t)} \\ &\leq \varepsilon \|G\|_{L^2(\Omega_t \times D)} \|J_t y\|_{L^2(\Omega_s)} \|J_s x\|_{L^2(\Omega_t)} \\ &\leq \varepsilon c_{J,2}^2 \|G\|_{L^2(\Omega_t \times D)} \|y\| \|x\|\,. \end{aligned}$$

Also folgt $\|B|_b - B_{\mathcal{H}}|_b\|_2 \leq \varepsilon c_{J,2}^2 \|G\|_{L^2(\Omega_t \times D)}$.
Die Abschätzung von G führen wir hier nur für die Fundamentallösung des Laplace-Operators durch, sie lässt sich jedoch auf allgemeine Differentialoperatoren erweitern. Man erhält dann die gesuchte Abschätzung

$$\|G\|_{L^2(\Omega_t \times D)} \leq \frac{c}{\lambda_{\min}} \delta^2\,.$$

Mit $G(x,y) = \frac{1}{4\pi} \frac{1}{|x-y|}$ und $\delta := \operatorname{dist}(\Omega_t, D) = 1/2 \operatorname{dist} \Omega_t, \Omega_s \geq 1/2 \operatorname{diam}(\Omega_t)$ (analog: $\delta \geq 1/2 \operatorname{diam} \Omega_s$) folgt

$$\|G\|_{L^2(\Omega_t \times D)} \leq \frac{1}{4\pi\delta} \sqrt{\operatorname{vol}(\Omega_t) \operatorname{vol}(D)}\,.$$

Wegen $\operatorname{vol}(D), \operatorname{vol}(\Omega_t) \leq C\delta^3$ gilt damit

$$\|G\|_{L^2(\Omega_t \times D)} \leq \frac{C}{4\pi} \delta^2\,,$$

und mit $\delta \leq \operatorname{diam}(\Omega)$ und Lemma 7.8 erhält man die Behauptung. □

7.7 Inverse Steifigkeitsmatrix

Nun können wir die Ergebnisse aus den vorherigen Kapiteln zusammenfassen und eine $\mathcal{H}$-Matrix-Approximation der Inversen Steifigkeitsmatrix angeben. Es gilt:

Satz 7.12. *Sei $\varepsilon \in (0,1)$ und $k_\varepsilon \in \mathcal{O}(\log^{d+1}(\frac{L}{\varepsilon}))$, $k := C_{id}^2 C_{sp}^2 (L+1) k_\varepsilon$ aus der Rangabschätzung bei der Multiplikation von $\mathcal{H}$-Matrizen. Sei weiterhin ε_h wie in Satz 7.4. Dann gibt es $C_\mathcal{H} \in \mathcal{H}(P,k)$, so dass*

$$\left\|A^{-1} - C_\mathcal{H}\right\|_2 \leq C_1 \varepsilon_h + C_2 \varepsilon \,.$$

Beweis. Nach Satz 7.9 und 7.11 gibt es $N_\mathcal{H} \in \mathcal{H}(P,k_\varepsilon)$ und $B_\mathcal{H} \in \mathcal{H}(P,k_\varepsilon)$, die die Matrizen M^{-1} und B approximieren.
Wir definieren $C_\mathcal{H}$ als

$$C_\mathcal{H} := N_\mathcal{H} B_\mathcal{H} N_\mathcal{H} \,.$$

Damit gilt $C_\mathcal{H} \in \mathcal{H}(P,k)$. Wegen

$$M^{-1}BM^{-1} - N_\mathcal{H} B_\mathcal{H} N_\mathcal{H} = (M^{-1} - N_\mathcal{H})BM^{-1} + N_\mathcal{H}(B - B_\mathcal{H})M^{-1} + N_\mathcal{H} B_\mathcal{H}(M^{-1} - N_\mathcal{H})$$

und der Beschränktheit von $\|B\|_2$, $\left\|M^{-1}\right\|_2$ und damit auch von $\|B_\mathcal{H}\|_2$, $\|N_\mathcal{H}\|$ erhält man

$$\left\|M^{-1}BM^{-1} - C_\mathcal{H}\right\| \leq C_2 \varepsilon \,.$$

Mit Satz 7.6 folgt

$$\left\|A^{-1} - C_\mathcal{H}\right\| \leq C_1 \varepsilon_h + C_2 \varepsilon \,.$$

□

Falls $\varepsilon_h \in \mathcal{O}(h^\beta)$, so kann man daraus folgern, dass es $C_\mathcal{H} \in \mathcal{H}(P,k_C)$ mit $k_C \in \mathcal{O}(\log^{d+3}(n))$ gibt, so dass

$$\left\|A^{-1} - C_\mathcal{H}\right\| \leq C \varepsilon_h \,.$$

Wir haben also gesehen, dass man im Fall des Laplace-Operators wegen der Glattheit der Green-Funktion leicht eine $\mathcal{H}$-Matrix-Approximation mit beschränktem Rang angeben kann. Wir werden das Ergebnis nun auf Differentialoperatoren mit Koeffizienten aus $L^\infty(\Omega)$ verallgemeinern

7.8 Allgemeine Differentialoperatoren

Wir betrachten einen elliptischen Differentialoperator in einem Gebiet $\Omega \subset \mathbb{R}^d$ mit Dirichlet-Randbedingungen:

$$Lu = -\sum_{i,j=1}^{d} \partial_j (c_{ij} \partial_i u) = f \quad \text{in } \Omega \,, \tag{7.19}$$

$$u = 0 \quad \text{auf } \partial\Omega \,, \tag{7.20}$$

wobei $c_{ij} \in L^\infty(\Omega)$ und $C = (c_{ij})$ symmetrisch mit $0 < \lambda_{\min} \leq \lambda \leq \lambda_{\max}$.
Wir können die Inverse Steifigkeitsmatrix A^{-1} wieder durch $M^{-1}BM^{-1}$ approximieren, sämtliche Sätze aus den vorherigen Abschnitten lassen sich übertragen. Das Problem ist jedoch,

dass die Green-Funktion nicht mehr glatt ist, weshalb sich Satz 7.10 nicht mehr durch Taylor-Entwicklung zeigen lässt.
Der Beweis von Satz 7.10 ist für den allgemeinen Fall jedoch recht umfangreich, weshalb wir hier nur die Vorgehensweise skizzieren werden. Den ausführlichen Beweis kann man in [3] finden.
Die Grundidee des Beweises ist, dass wir zu einem abgeschlossenen Unterraum von $L^2(D)$ einen k-dimensionalen Unterraum finden können, der eine Approximationseigenschaft erfüllt:

Lemma 7.13. *Sei $D \subset \mathbb{R}^d$ konvex und X abgeschlossener Unterraum von $L^2(D)$. Zu $k \in \mathbb{N}$ gibt es einen Unterraum $V_k \subseteq X$ mit $\dim V_k \leq k$ so dass*

$$\operatorname{dist}_{L^2(\Omega)}(u, V_k) \leq c_{appr} \frac{\operatorname{diam}(D)}{\sqrt[d]{k}} \|\nabla u\|_{L^2(\Omega)} \quad \forall u \in X \cup H^1(D)\,.$$

Der Beweis beruht auf der Poincaré-Friedrichs-Ungleichung.
Wenn wir also einen abgeschlossenen Unterraum $X(D)$ finden, in dem $G(x,\cdot)$ liegt und $\|\nabla u\|_{L^2(\Omega)}$ in diesem Raum durch $\|u\|_{L^2(\Omega)}$ abschätzen können, so finden wir damit eine endlich-dimensionale Approximation der Green-Funktion.
Wie es sich herausstellt erfüllt der Raum der L-harmonischen Funktionen

$$X(D) := \{u \in H_0^1(D) : a(u,\phi) = 0 \ \forall \phi \in C_0^\infty(D)\} \tag{7.21}$$

für $D \subseteq \Omega$ diese Eigenschaften. Die Abgeschlossenheit und die Abschätzung von $\|\nabla u\|_{L^2(\Omega)}$ folgen aus der Caccioppoli-Ungleichung:

Lemma 7.14. *Sei $K \subset D \subseteq \Omega$ mit $\operatorname{dist}(K, \partial D) > 0$, $\kappa_C := \lambda_{\max}/\lambda_{\min}$. Dann gilt:*

$$\|\nabla u\|_{L^2(K)} \leq \frac{4\sqrt{\kappa_C}}{\operatorname{dist}(K, \partial D)} \quad \forall u \in X(D)\,.$$

Somit kann man folgenden Satz zeigen:

Satz 7.15. *Es gelten die gleichen Voraussetzungen wie in Lemma 7.14. Weiter sei $D_2 \subset D$ mit*

$$\operatorname{dist}(D_2, \partial D) \geq \operatorname{diam}(D_2) > 0\,.$$

Dann gibt es zu $M > 1$ einen Unterraum $W \subset X(D_2)$ so dass

$$\operatorname{dist}_{L^2(D)}(u, W) \leq \frac{1}{M} \|u\|_{L^2(D)} \quad \forall u \in X(D)$$

und $\dim W \in \mathcal{O}(\log^{d+1} M)$.

Setzt man $M = 1/\varepsilon$ und verwendet, dass $G(x,\cdot) \in X(D)$, so folgt daraus Satz 7.10.

7.9 Zusammenfassung und Realisierung

Wir haben gesehen, dass zur inversen Steifigkeitsmatrix A^{-1} eine $\mathcal{H}$-Matrix-Approximation $C_{\mathcal{H}}$ mit beschränktem maximalen Rang existiert. Dies ist jedoch lediglich ein Existenzbeweis, da er auf der im allgemeinen unbekannte Green-Funktion basiert.
In der Praxis wird deshalb mit dem in Vortrag 3 vorgestellten Algorithmus die Steifigkeitsmatrix invertiert und daraus die Näherungslösung u_h bestimmt. Dieses Verfahren hat zwar

bei beschränktem Rang der Matrix einen optimalen Aufwand, jedoch kann man den Fehler der so konstruierten inversen Matrix nicht mehr abschätzen.
Das Problem, ein Verfahren zu finden, dass eine $\mathcal{H}$-Matrix-Approximation der inversen Massenmatrix berechnet, die die in diesem Kapitel gezeigten Eigenschaften aufweist, bleibt also nach wie vor offen.

Kapitel 8

Adaptive Cross Approximation

Gregor Gassner

8.1 Einleitung

Bei den Randelementmethoden treten vollbesetzte Matrizen der Form

$$a_{ij} := \int_{\Gamma} \kappa(x, y_i)\phi_j(x)ds_x \tag{8.1}$$

beim Kollokationsverfahren, beziehungsweise

$$a_{ij} := \kappa(x_j, y_i) \tag{8.2}$$

beim Nyström-Verfahren, oder

$$a_{ij} := \int_{\Gamma}\int_{\Gamma} \kappa(x, y)\phi_j(x)\phi_i(y)ds_x ds_y \tag{8.3}$$

beim Galerkin-Verfahren auf.
Das Aufstellen solcher Matrizen ist mit einem großen Speicher- und Rechenaufwand verbunden. In diesem Vortrag soll ein Verfahren aufgezeigt werden, welches Matrizen der Art (8.1), (8.2) und (8.3) mit möglichst wenig Rechenaufwand und möglichst wenig Speicherplatzbedarf generiert. Die bei der Randelementmethode auftretenden Kerne κ sind für $x = y$ singulär und damit im sogenannten Nahfeld nicht glatt. Zum Beispiel lautet der Kern für die dreidimensionale Potentialgleichung

$$\kappa(x, y) = \frac{1}{4\pi}\frac{1}{|x - y|}.$$

Im Fernfeld sind solche Kerne glatt und können dort mit verschiedenen Methoden approximiert werden. Das Verfahren basiert also auf einer Partitionierung der Matrix A in Blöcke B des Nahfeldes und des Fernfeldes, wobei wir für diesen Vortrag nur Blöcke des Fernfeldes betrachten, sogenannte η-zulässige Blöcke. Blöcke des Nahfeldes werden hier nicht weiter behandelt und sind demnach voll besetzt und haben vollen Rang.

Definition 8.1. Ein Paar (D_1, D_2) beschränkter Mengen $D_1, D_2 \subset \mathbb{R}^d$ mit

$$diam(D_2) \leq \eta \cdot dist(D_1, D_2)$$

heißen η-zulässig.

Für solche η-zulässigen Blöcke mit vollem Rang, werden wir einen Algorithmus zur Niedrigrangapproximation konstruieren. Die resultierenden Matrizen werden auch als hierarchische Matrizen, bzw. $\mathcal{H}$-Matrizen bezeichnet. Das hier vorgestellte Verfahren benötigt dazu im Allgemeinen alle Einträge der Originalmatrix, um den Approximationsfehler zu garantieren. Für spezielle Kerne sind aber nur wenige Einträge nötig. Um diese Eigenschaft zu garantieren, beschränken wir uns also auf eine bestimmte Klasse von Kernfunktionen κ, nämlich Kernfunktionen, die asymptotisch glatt bezüglich y sind.

Definition 8.2. Eine Funktion $f : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ wird als asymptotisch glatt (bezüglich y) bezeichnet, falls $f(x, y)$ für $y \neq x$ bezüglich y unendlich oft differenzierbar ist und für jeden Multiindex $\alpha \in \mathbb{N}_0^d$ gilt

$$\left|D_y^\alpha f(x, y)\right| \leq c_p \left|x - y\right|^{g-p}, \quad p := |\alpha|$$

wobei c_p nur von p abhängt und g die Singularität von f beschreibt.

Im Fall (8.1) und (8.2) entstehen die Blockmatrizen $B \in \mathbb{R}^{m \times n}$ durch Auswertung von linearen Abbildungen $\mathcal{L}_1\kappa, ..., \mathcal{L}_n\kappa$ an Punkten $\{y_1, .., y_m\} \subset D_2$, d.h.

$$b_{ij} = (\mathcal{L}_j\kappa)(y_i) \quad i = 1, ..., m, j = 1, ..., n.$$

Für die Nyströmmethode ist

$$(\mathcal{L}_j\kappa)(y) := \kappa(x_j, y), \quad j = 1, ..., n,$$

ein Operator der Punktauswertung, wobei $x_j \in D_1$, $j = 1, ..., n$.
Für das Kollokationsverfahren ist

$$(\mathcal{L}_j\kappa)(y) := \int_{D_1} \kappa(x, y)\phi_j(x)ds_x, \quad j = 1, ..., n.$$

ein Integraloperator. Das Galerkin-Verfahren ist auf diese Weise nicht behandelbar, deshalb beschränken wir uns hier auf Matrizen der Art (8.1) und (8.2).
Das Ziel des nächsten Abschnittes ist es, die Abbildung $\mathcal{L}_j\kappa$ folgendermaßen zu zerlegen:

$$(\mathcal{L}_j\kappa)(y) = ([\mathcal{L}]_k\kappa)(y)^T G_k(\mathcal{L}_j\kappa)([\zeta]_k) + R_k(y), \quad y \in D_2. \tag{8.4}$$

Dabei ist G_k eine $k \times k$ Matrix und R_k der Approximationsfehler.
In (8.4) werden die Schreibweisen

$$f([\zeta]_k) = \begin{pmatrix} f(\zeta_{i_1}) \\ \vdots \\ f(\zeta_{i_k}) \end{pmatrix} \quad \text{und} \quad ([\mathcal{L}]_k\kappa)(y) = \begin{pmatrix} (\mathcal{L}_{j_1}\kappa)(y) \\ \vdots \\ (\mathcal{L}_{j_k}\kappa)(y) \end{pmatrix}$$

mit Punkten $\zeta_{i_l} \in \mathbb{R}^d$, $l = 1, ..., k$ verwendet.

Wichtig ist hierbei, dass zur Approximation von $\mathcal{L}_j\kappa$ nicht eine spezielle zum Kern κ passende Reihe verwendet wird, sondern einige ausgewählte Abbildungen $\mathcal{L}_{j_1}\kappa, ..., \mathcal{L}_{j_k}\kappa$.

8.2 Das analytische Problem

Der Algorithmus soll schrittweise eine Approximation

$$\kappa(x,y) = \sum_{i=1}^{k} u_i(y)v_i(x) + r_k(x,y) = s_k(x,y) + r_k(x,y) \quad \text{mit} \quad x \in D_1, y \in D_2$$

aufbauen, wobei s_k die Approximation und r_k das zugehörige Residuum ist.Die η-Zulässigkeit des Paares (D_1, D_2) benötigen wir, um den Approximationsfehler R_k zu kontrollieren. Wir betrachten nun die Konstruktion der Funktionenfolgen $\{s_k\}$ und $\{r_k\}$. Dazu definieren wir uns eine weitere Menge von Punkten $\zeta_i \in D_2, i = 1, ..., m_p$. Die Wahl von m_p und den Punkten ζ_i wird erst später diskutiert.

Algorithmus 8.1. Sei $i_0 = 0$, $r_0(x,y) = \kappa(x,y)$ und $s_0(x,y) = 0$.
Im k-ten Schritt sei i_k das erste Element in $\{i_{k-1}+1, ..., m_p\}$, so dass ein $j \in \{1, ..., n\}$ mit $(\mathcal{L}_j r_{k-1})(\zeta_{ik}) \neq 0$ existiert.
Läßt sich kein solches j finden, endet die Konstruktion hier.
Im anderen Fall können wir aus allen solchen j nach einem Kriterium, dass wir noch angeben werden, einen Index j_k auswählen.
Bereche $\gamma_k^{-1} = (\mathcal{L}_{j_k} r_{k-1})(\zeta_{i_k})$ und damit

$$r_k(x,y) = r_{k-1}(x,y) - \gamma_k(\mathcal{L}_{j_k} r_{k-1})(y) r_{k-1}(x, \zeta_{i_k}),$$
$$s_k(x,y) = s_{k-1}(x,y) + \gamma_k(\mathcal{L}_{j_k} r_{k-1})(y) r_{k-1}(x, \zeta_{i_k}).$$

Man stellt fest, dass die Funktionen $\mathcal{L}_j r_k$ Nullstellen ansammeln. Es gilt nämlich

Lemma 8.1. *Für $k \geq 0$ und alle $1 \leq i < i_{k+1}$ gilt*

$$(\mathcal{L}_j r_k)(\zeta_i) = 0, \quad j = 1, ..., n$$

und für $1 \leq l \leq k$

$$(\mathcal{L}_{j_l} r_k)(y) = 0 \quad \textit{für alle} \quad y \in \mathbb{R} \setminus D_1.$$

Beweis. Siehe [2, S. 80]. □

Wegen $\mathcal{L}_j \kappa = \mathcal{L}_j s_k + \mathcal{L}_j r_k$ interpoliert $\mathcal{L}_j s_k$ deshalb mehr und mehr $\mathcal{L}_j \kappa$.

Definition 8.3. Sei $\widehat{B}_k \in \mathbb{R}^{k \times k}$ mit

$$\widehat{B}_k = \begin{pmatrix} (\mathcal{L}_{j_1}\kappa)(\zeta_{i_1}) & \cdots & (\mathcal{L}_{j_k}\kappa)(\zeta_{i_1}) \\ \vdots & & \vdots \\ (\mathcal{L}_{j_1}\kappa)(\zeta_{i_k}) & \cdots & (\mathcal{L}_{j_k}\kappa)(\zeta_{i_k}) \end{pmatrix}$$

und $\widehat{B}_k(l,j) \in \mathbb{R}^{k \times k}$ die Matrix, die aus $\widehat{B}_k$ durch Ersetzen der l-ten Spalte durch den Vektor $(\mathcal{L}_j \kappa)([\zeta]_k)$ entsteht. Insbesondere gilt dann $\widehat{B}_k = \widehat{B}_k(l, j_l)$.

Lemma 8.2. *Für $1 \leq l \leq k$ gilt*

$$det\widehat{B}_k(l,j) = \gamma_k^{-1} det\widehat{B}_{k-1}(l,j) - (\mathcal{L}_j r_{k-1})(\zeta_{i_k}) det\widehat{B}_{k-1}(l, j_k)$$

und

$$\begin{aligned} det\widehat{B}_1(1,j) &= (\mathcal{L}_j\kappa)(\zeta_{i_1}), \\ det\widehat{B}_k(k,j) &= (\mathcal{L}_j r_{k-1})(\zeta_{i_k}) det\widehat{B}_{k-1}, \quad k \geq 1. \end{aligned}$$

Insbesondere gilt

$$det\widehat{B}_k = \prod_{l=1}^{k} \gamma_l^{-1}.$$

Beweis. Wie man leicht sieht (z.B. [17, S. 326]), gibt es Funktionen $\alpha_\nu^{(k-1)}, \nu = 1, ..., k-1$, so dass für alle $x \in D_1$ und $y \in \mathbb{R}^d \setminus D_1$ gilt

$$r_{k-1}(x,y) = \kappa(x,y) - \sum_{\nu=1}^{k-1} \alpha_\nu^{(k-1)}(y)\kappa(x,\zeta_{i_\nu}).$$

Sei $1 \leq l < k$. Wir subtrahieren von der letzten Zeile von $\widehat{B}_k(l,j)$ eine Linearkombination der übrigen Zeilen. Dabei benutzen wir als Koeffizienten $\alpha_\nu^{(k-1)}(\zeta_{i_k}), \nu = 1, ..., k-1$. Die Determinante der neuen Matrix $\widetilde{B}_k(l,j)$ stimmt mit der von $\widehat{B}_k(l,j)$ überein. Weil nun für $s \in \{1, ..., l-1, l+1, ..., k\}$

$$(\widetilde{B}_k(l,j))_{ks} = (\mathcal{L}_{j_s}\kappa)(\zeta_{i_k}) - \sum_{\nu=1}^{k-1} \alpha_\nu^{(k-1)}(\zeta_{i_k})(\mathcal{L}_{j_s}\kappa)(\zeta_{i_\nu}),$$

folgt wegen Lemma 8.1 im Fall $s \in \{1, ..., l-1, l+1, ..., k-1\}$

$$(\widetilde{B}_k(l,j))_{ks} = 0.$$

Die letzte Zeile von $\widetilde{B}_k(l,j)$ hat also nur zwei Einträge, die möglicherweise nicht verschwinden. Dies sind

$$(\widetilde{B}_k(l,j))_{kl} = (\mathcal{L}_j r_{k-1})(\zeta_{i_k})$$

und

$$(\widetilde{B}_k(l,j))_{kk} = (\mathcal{L}_{j_k} r_{k-1})(\zeta_{i_k}) = \gamma_k^{-1}.$$

Mit Hilfe des Laplaceschen Entwicklungssatzes folgt die Behauptung. □

Dieses Lemma garantiert nicht nur, dass $\widehat{B}_k$ nicht singulär ist, wir sehen auch, dass die Determinante einen umso größeren Wert besitzt, je größer das Produkt der Pivot-Elemente γ_k^{-1} ist. Man kann nun zeigen, dass die Zerlegung von κ in s_k und r_k vom Typ (8.4) ist.

Lemma 8.3. *Für die generierten Folgen $\{s_k\}_k$ und $\{r_k\}_k$ gilt*

$$s_k(x,y) + r_k(x,y) = \kappa(x,y),$$

wobei für $k \geq 1$

$$s_k(x,y) = ([\mathcal{L}]_k\kappa)(y)^T \widehat{B}_k^{-1} \kappa(x, [\zeta]_k).$$

Beweis. Im Fall $k = 1$ ist das Lemma offenbar richtig. Wir fahren per Induktion von $k-1$ nach k fort. Aus der Definition von r_k und s_k sehen wir, dass

$$s_k(x,y) + r_k(x,y) = s_{k-1}(x,y) + r_{k-1}(x,y) = ... = s_0(x,y) + r_0(x,y) = \kappa(x,y).$$

Wir setzen der Einfachheit halber

$$a_k = \widehat{B}_{k-1}^{-1}(\mathcal{L}_{j_k}\kappa)([\zeta]_{k-1}) \quad \text{und} \quad b_k = \widehat{B}_{k-1}^{-T}([\mathcal{L}]_{k-1}\kappa)(\zeta_{i_k}).$$

Wegen

$$\begin{aligned} s_k(x,y) &= s_{k-1}(x,y) + \gamma_k(\mathcal{L}_{j_k}r_{k-1})(y)r_{k-1}(x,\zeta_{i_k}) \\ &= ([\mathcal{L}]_k\kappa)(y)^T \begin{pmatrix} \widehat{B}_{k-1}^{-1} + \gamma_k a_k b_k^T & -\gamma_k a_k \\ -\gamma_k b_k^T & \gamma_k \end{pmatrix} \kappa(x,[\zeta]_k) \end{aligned}$$

und

$$\widehat{B}_k \begin{pmatrix} \widehat{B}_{k-1}^{-1} + \gamma_k a_k b_k^T & -\gamma_k a_k \\ -\gamma_k b_k^T & \gamma_k \end{pmatrix} = I_k$$

hat mit s_{k-1} auch s_k die Form

$$s_k(x,y) = ([\mathcal{L}]_k\kappa)(y)^T \widehat{B}_k^{-1}\kappa(x,[\zeta]_k).$$

□

Bezeichnen wir nun mit $M_k(l,y)$ die Matrix, bei der in $\widehat{B}_k$ die l-te Zeile durch den Vektor $([\mathcal{L}]_k\kappa)(y)$ ersetzt ist, d.h.

$$M_k(l,y) = \begin{pmatrix} (\mathcal{L}_{j_1}\kappa)(\zeta_{i_1}) & \cdots & (\mathcal{L}_{j_k}\kappa)(\zeta_{i_1}) \\ \vdots & & \vdots \\ (\mathcal{L}_{j_1}\kappa)(y) & \cdots & (\mathcal{L}_{j_k}\kappa)(y) \\ \vdots & & \vdots \\ (\mathcal{L}_{j_1}\kappa)(\zeta_{i_k}) & \cdots & (\mathcal{L}_{j_k}\kappa)(\zeta_{i_k}) \end{pmatrix} \leftarrow l,$$

dann sieht man mit der Cramerschen Regel, dass

$$\left(([\mathcal{L}]_k\kappa)(y)^T\widehat{B}_k^{-1}\right)_l = \frac{detM_k(l,y)}{det\widehat{B}_k}.$$

Also folgt

$$(\mathcal{L}_j s_k)(y) = \sum_{l=1}^{k} \frac{detM_k(l,y)}{det\widehat{B}_k}(\mathcal{L}_j\kappa)(\zeta_{i_l}). \tag{8.5}$$

Die Darstellung (8.5) zeigt, dass $\mathcal{L}_j s_k$ die eindeutig bestimmte Interpolierende von $\mathcal{L}_j\kappa$ in den Punkten ζ_{i_l} in der linearen Hülle der Funktionen $\mathcal{L}_{j_l}\kappa, l = 1, ..., k$, ist. Das bedeutet, dass (8.5) die Lagrange-Interpolation mit den Lagrange-Funktionen $\mathcal{L}_j\kappa$ ist.

8.3 Interpolationsprobleme

Zur Kontrolle des Fehlers der Kernapproximation benötigen wir den Fehler der Interpolation (8.5). Da ein Fehler im Funktionensystem $\mathcal{L}_{j_l}\kappa, l = 1, ..., k$ nicht bekannt ist, werden wir daher den Fehler $\mathcal{L}_j r_k$ mit dem Fehler eines anderen Funktionensystems in Beziehung bringen.
Sei $\Psi_1, ..., \Psi_{m_p}$ ein weiteres Funktionensystem mit

$$det([\Psi_j(\zeta_i)]_{ij}) \neq 0.$$

Sei $L^{\Psi}_{m_p}[\mathcal{L}_j\kappa]$ die Interpolierende von $\mathcal{L}_j\kappa$ im Funktionenraum Ψ, dann ist für dieses System der Interpolationsfehler wie folgt definiert,

$$E^{\Psi}_p[\mathcal{L}_j\kappa] := \mathcal{L}_j\kappa - L^{\Psi}_{m_p}[\mathcal{L}_j\kappa].$$

Nun sind wir in der Lage, den Rest $\mathcal{L}_j r_k$ der Appromimation mit dem Restglied E^{Ψ}_p der Interpolation im Ψ-System in Beziehung zu setzen.

Lemma 8.4. *Seien $\zeta_{i_1}, ..., \zeta_{i_k}$ die bei der Konstruktion der Folgen $\{r_k\}_k$ und $\{s_k\}_k$ verwendeten Pivot-Punkte der Punktmenge $\zeta_1, ..., \zeta_{m_p}$. Dann gilt für die Funktionen $\mathcal{L}_j r_k$, dass*

$$(\mathcal{L}_j r_k)(y) = E^{\Psi}_p[\mathcal{L}_j\kappa](y) - \sum_{l=1}^{k} \frac{det\widehat{B}_k(l,j)}{det\widehat{B}_k} E^{\Psi}_p[\mathcal{L}_{j_l}\kappa](y), \quad y \in \mathbb{R}^d \setminus D_1.$$

Beweis. Siehe [2, S. 85]. □

Als nächstes werden wir die Pivot-Elemente γ_k auswählen. Das Ziel ist es, die Koeffizienten $\frac{det\widehat{B}_k(l,j)}{det\widehat{B}_k}$ in Lemma 8.4 zu kontrollieren.

8.4 Wahl der Pivot-Elemente

Lemma 8.5. *Sei j_k in jedem Schritt so gewählt, dass*

$$|(\mathcal{L}_{j_k} r_{k-1})(\zeta_{i_k})| \geq |(\mathcal{L}_j r_{k-1})(\zeta_{i_k})| \quad f\ddot{u}r \quad 1 \leq j \leq n \tag{8.6}$$

gilt. Dann folgt für $1 \leq l \leq k$ und $j = 1, ..., n$ die Abschätzung

$$\left|det\widehat{B}_k(l,j)\right| \leq 2^{k-l} \left|det\widehat{B}_k\right|.$$

Beweis. Wir benützen $\gamma_k^{-1} = (\mathcal{L}_{j_k} r_{k-1})(\zeta_{i_k})$. Damit ergibt sich für $1 \leq l \leq k$ (siehe Lemma 8.2)

$$\begin{aligned} det\widehat{B}_k(l,j) &= \gamma_k^{-1} det\widehat{B}_{k-1}(l,j) - (\mathcal{L}_j r_{k-1})(\zeta_{i_k}) det\widehat{B}_{k-1}(l,j_k) \\ &= (\mathcal{L}_{j_k} r_{k-1})(\zeta_{i_k}) det\widehat{B}_{k-1}(l,j) - (\mathcal{L}_j r_{k-1})(\zeta_{i_k}) det\widehat{B}_{k-1}(l,j_k) \end{aligned}$$

$$det\widehat{B}_k = \prod_{l=1}^{k} \gamma_l^{-1} = \gamma_k^{-1} det\widehat{B}_{k-1} = (\mathcal{L}_{j_k} r_{k-1})(\zeta_{i_k}) det\widehat{B}_{k-1}.$$

$$det\widehat{B}_k(k,j) = (\mathcal{L}_j r_{k-1})(\zeta_{i_k}) det\widehat{B}_{k-1}.$$

Bildet man nun die folgenden Quotienten,

$$\frac{det\widehat{B}_k(l,j)}{det\widehat{B}_k} = \frac{det\widehat{B}_{k-1}(l,j)}{det\widehat{B}_{k-1}} - \frac{(\mathcal{L}_j r_{k-1})(\zeta_{i_k})}{(\mathcal{L}_{j_k} r_{k-1})(\zeta_{i_k})} \frac{det\widehat{B}_{k-1}(l,j_k)}{det\widehat{B}_{k-1}}$$

und

$$\frac{det\widehat{B}_k(k,j)}{det\widehat{B}_k} = \frac{(\mathcal{L}_j r_{k-1})(\zeta_{i_k})}{(\mathcal{L}_{j_k} r_{k-1})(\zeta_{i_k})} \overset{(8.6)}{\leq} 1, \tag{8.7}$$

dann erhalten wir für $1 \leq l \leq k$

$$\begin{aligned}
\max_{j=1,\ldots,n} \frac{\left|det\widehat{B}_k(l,j)\right|}{\left|det\widehat{B}_k\right|} &\leq 2 \max_{j=1,\ldots,n} \frac{\left|det\widehat{B}_{k-1}(l,j)\right|}{\left|det\widehat{B}_{k-1}\right|} \\
&\leq 2^{k-l} \max_{j=1,\ldots,n} \frac{\left|det\widehat{B}_l(l,j)\right|}{\left|det\widehat{B}_l\right|} \\
&\overset{(8.7)}{\leq} 2^{k-l}.
\end{aligned}$$

Hieraus folgt die Behauptung. □

Wir wählen also in jedem Schritt j_k so, dass γ_k^{-1} das betragsmäßig größte Element aller j mit $(\mathcal{L}_j r_k)(\zeta_{i_k}) \neq 0$ ist. Mit dieser Strategie erreichen wir zwar nicht die Untermatrix mit der betragsmäßig größten Determinante, wir können jedoch mit Lemma 8.4 sehen, dass diese maximale Element-Strategie die bestmögliche Wahl in Hinsicht auf größte Determinanten ist, falls wir alle bisher gewählten Indizes $j_1, ..., j_{k-1}$ beibehalten.
Mit Lemma (8.4) und Lemma (8.5) ergibt sich nun folgende Abschätzung.
Für $y \in \mathbb{R}^d \setminus D_1$ und $j = 1, ..., n$ gilt

$$|(\mathcal{L}_j r_k)(y)| \leq (1 + 2^k) \max_{j=1,\ldots,n} \left|E_p^{\Psi}[\mathcal{L}_j \kappa](y)\right|. \tag{8.8}$$

Im Gegensatz zur eindimensionalen Lagrange-Interpolation weist die Interpolationsaufgabe in mehreren Raumdimensionen einige Schwierigkeiten auf. Insbesondere ist die Eindeutigkeit des Interpolationspolynoms abhängig von der Wahl der Interpolationsknoten. Mit dieser Problematik werden wir uns im Abschnitt 8.6 weiter beschäftigen und verweisen hier auf [16].

8.5 Der Algorithmus

Wir werden nun einen ersten Algorithmus zur Approximation der Matrix B mit

$$b_{ij} = (\mathcal{L}_j \kappa)(y_i) \quad 1 \leq i \leq m, 1 \leq j \leq n$$

durch eine Niedrigrang-Matrix formulieren. Dieser Algorithmus wird noch alle Einträge der Matrix benötigen.
Die Folgen $\{\widehat{R}_k\}$ und $\{\widehat{S}_k\}$ von $(m + m_p) \times n$ Matrizen werden induktiv konstruiert. Dazu definieren wir uns

$$z_i := \begin{cases} \zeta_i, & 1 \leq i \leq m_p \\ y_{i-m_p}, & m_p \leq i \leq m + m_p \end{cases}.$$

Algorithmus 8.2. (**vollständig pivotisierter ACA**).
Setze $i_0 = 0$, $\widehat{S}_0 = 0$ und

$$(\widehat{R}_0)_{ij} = (\mathcal{L}_j\kappa)(z_i), \quad 1 \le i \le m + m_p, 1 \le j \le n.$$

Für k=1,2,... finde den kleinsten Index $i_k \in \{i_{k-1} + 1, ..., m_p\}$, so dass $\underline{e}_{i_k}^T \widehat{R}_{k-1} \neq 0$.
Wähle

$$\gamma_k^{-1} = sign(\underline{e}_{i_k}^T \widehat{R}_{k-1}\underline{e}_j) \max_{1\le j\le n} \left|\underline{e}_{i_k}^T \widehat{R}_{k-1}\underline{e}_j\right|$$

und bezeichne den zugehörigen Index mit j_k.
Setze

$$\widehat{R}_k = \widehat{R}_{k-1} - \gamma_k \widehat{R}_{k-1}\underline{e}_{j_k}\underline{e}_{i_k}^T \widehat{R}_{k-1},$$
$$\widehat{S}_k = \widehat{S}_{k-1} + \gamma_k \widehat{R}_{k-1}\underline{e}_{j_k}\underline{e}_{i_k}^T \widehat{R}_{k-1}.$$

Der Abbruch erfolgt, wenn der relative Fehler

$$\frac{\|R_k\|_F}{\|R_0\|_F} \le \varepsilon$$

erfüllt. Die Vektoren $\underline{e}_i$ sind dabei die kanonischen Vektoren im jeweiligen Raum.

Die Erweiterung m_p bei der Berechnung des Approximanten kann nach der Generierung verworfen werden.
Von der Matrix $\widehat{S}_{k-1}$ wird eine Dyade aus den Vektoren $\widehat{R}_{k-1}\underline{e}_{j_k}$ und $\widehat{R}_{k-1}^T\underline{e}_{i_k}$ addiert. Dadurch hat $\widehat{S}_k$ höchstens den Rang k.
Es wird nun eine Verbindung der Einträge von $\widehat{R}_k$ mit den generierten Folgen $\{r_k\}_k$ hergestellt.

Lemma 8.6. *Für $i = 1, ...m + m_p$ und $j = 1, ..., n$ gilt*

$$(\widehat{R}_k)_{ij} = (\mathcal{L}_j r_k)(z_i).$$

Beweis. Für $k = 0$ ist die Aussage offensichtlich richtig. Wir überprüfen sie noch in den übrigen Fällen. Angenommen, die Aussage sei für ein $k - 1$ gezeigt, dann sind sowohl die Kriterien zur Wahl der Indizes i_k und j_k als auch die Größe γ_k identisch. Außerdem gilt

$$\begin{aligned}(\widehat{R}_k)_{ij} &= (\widehat{R}_{k-1})_{ij} - \gamma_k(\widehat{R}_{k-1})_{ij_k}(\widehat{R}_{k-1})_{i_kj} \\ &= (\mathcal{L}_j r_{k-1})(z_i) - \gamma_k(\mathcal{L}_{j_k} r_{k-1})(z_i)(\mathcal{L}_j r_{k-1})(\zeta_{i_k}) \\ &= (\mathcal{L}_j r_k)(z_i)\end{aligned}$$

für alle $j = 1, ..., n$. □

Der Algorithmus (8.2) entspricht also der diskreten Version der Generierung der Folgen $\{\mathcal{L}_j r_k\}_k$ und $\{\mathcal{L}_j s_k\}_k$.
Man kann nun den Algorithmus (8.2) in eine effiziente endgültige Form bringen, die nur einen Teil der Matrixeinträge benötigt.

Algorithmus 8.3. (**partiell pivotisierter ACA**).
Setze $i_0 = 0$. Für $k = 1, 2, ...$ finde den kleinsten Index i, $i_{k-1} \leq i \leq m_p$, so dass der Vektor

$$(\widetilde{v}_k)_j = (\mathcal{L}_j\kappa)(\zeta_i) - \sum_{l=1}^{k-1}(\widehat{u}_l)_i(v_l)_j, \quad j = 1, ..., n$$

nicht trivial ist. Ist dies nicht möglich, so endet der Algorithmus. Im anderen Fall setze $i_k = i$ und finde den betragsmäßig größten Eintrag im Vektor $\underline{\widetilde{v}}_k$.
Den zugehörigen Index bezeichne mit j_k.
Setze

$$\underline{v}_k = (\widetilde{v}_k)_{j_k}^{-1}\underline{\widetilde{v}}_k$$

und

$$(\widehat{u}_k)_i = (\mathcal{L}_{j_k}\kappa)(z_i) - \sum_{l=1}^{k-1}(\widehat{u}_l)_i(v_l)_{j_k}, \quad i = 1, ..., m + m_p.$$

Das Abbruchskriterium beziehungsweise der Approximationsfehler wird mit einem Fehlerschätzer kontrolliert (siehe Abschnitt 8.7).

Wir sehen, dass für die Approximation in jedem Schritt nur die Funktionsauswertungen $(\mathcal{L}_j\kappa)(\zeta_{i_k})$, $j = 1, ..., n$ und $(\mathcal{L}_{j_k}\kappa)(z_i)$, $i = 1, ..., m + m_p$, sowie algebraische Transformationen nötig sind.

Lemma 8.7. *Für den Approximanten $\widehat{S}_k$ gilt dann*

$$\widehat{S}_k = \sum_{l=1}^{k}\underline{\widehat{u}}_l\underline{v}_l^T.$$

Beweis. Für $k = 1$ stimmt die Aussage. Wir nehmen an, sie gilt für $k - 1$. Dann hat man nach Lemma (8.6) im k-ten Schritt für $j = 1, ..., n$

$$\begin{aligned}(\widetilde{v}_k)_j &= (\mathcal{L}_j\kappa)(\zeta_{i_k}) - \sum_{l=1}^{k-1}(\widehat{u}_l)_{i_k}(v_l)_j \\ &= (\widehat{R}_0)_{i_kj} - (\widehat{S}_{k-1})_{i_kj} = (\widehat{R}_{k-1})_{i_kj} \\ &= (\underline{e}_{i_k}^T\widehat{R}_{k-1})_j = (\mathcal{L}_j r_{k-1})(\zeta_{i_k}).\end{aligned}$$

Insbesondere stimmen die beiden Kriterien für die Wahl der Indizes i_k und j_k überein, und $(\widetilde{v}_k)_{j_k} = (\mathcal{L}_{j_k}r_{k-1})(\zeta_{i_k}) = \gamma_k^{-1}$. Für $i = 1, ..., m + m_p$ sieht man außerdem

$$\begin{aligned}(\widehat{u}_k)_i &= (\mathcal{L}_{j_k}\kappa)(z_i) - \sum_{l=1}^{k-1}(\widehat{u}_l)_i(v_l)_{j_k} \\ &= (\widehat{R}_0)_{ij_k} - (\widehat{S}_{k-1})_{ij_k} \\ &= (\widehat{R}_{k-1})_{ij_k} \\ &= (\widehat{R}_{k-1}\underline{e}_{j_k})_i.\end{aligned}$$

Also haben wir

$$\widehat{S}_k = \widehat{S}_{k-1} + \gamma_k\widehat{R}_{k-1}\underline{e}_{j_k}\underline{e}_{i_k}^T\widehat{R}_{k-1} = \sum_{l=1}^{k-1}\underline{\widehat{u}}_l\underline{v}_l^T + \underline{\widehat{u}}_k\underline{v}_k^T = \sum_{l=1}^{k}\underline{\widehat{u}}_l\underline{v}_l^T.$$

Damit ist die Behauptung bewiesen. □

Insgesamt benötigt man also zur Berechnung von $\widehat{S}_k$ höchstens $(m+m_p+n)k$ Speichereinheiten und höchstens

$i_k n + k(m+m_p)$	Auswertungen von $\mathcal{L}_j\kappa$,
$2ki_k n + k^2(m+m_p)$	Additionen und Multiplikationen,
kn	Divisionen.

Unter der Annahme, dass jede Auswertung $(\mathcal{L}_j\kappa)(y)$ in $O(1)$ Aufwand durchgeführt werden kann, sind dies insgesamt $O((m+m_p+n)i_k^2)$ Operationen für die Generierung des Approximanten $\widehat{S}_k$.

Nehmen wir kurz an, dass $i_l = j_l = l$, $l = 1, ..., k$ gilt, dann ergibt sich

$$\widehat{R}_k = (I - \gamma_k \widehat{R}_{k-1}\underline{e}_k\underline{e}_k^T)\widehat{R}_{k-1} =: L_k\widehat{R}_{k-1}.$$

Wobei L_k eine $(m_p+m)\times(m_p+m)$ Matrix ist, mit

$$L_k = \begin{pmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & 1 & & & & \\ & & & 0 & & & \\ & & & -\frac{\underline{e}_{k+1}^T\widehat{R}_{k-1}\underline{e}_k}{\underline{e}_k^T\widehat{R}_{k-1}\underline{e}_k} & 1 & & \\ & & & \vdots & & \ddots & \\ & & & -\frac{\underline{e}_{m+m_p}^T\widehat{R}_{k-1}\underline{e}_k}{\underline{e}_k^T\widehat{R}_{k-1}\underline{e}_k} & & & 1 \end{pmatrix}$$

Wir sehen, dass L_k sich nur an der Position (k,k) von einer Gauss-Matrix unterscheidet. Im gewissen Sinn produziert der Algorithmus also eine Spalten-pivotisierte LU-Zerlegung für die Approximation. Es ist bekannt, dass die Elemente während einer Spalten-pivotisierten LU-Zerlegung wachsen können. Der Term $(1+2^k)$ in (8.8) ist also kein Produkt ungenauer Abschätzungen.

Beispiel 8.1. Wir werden nun zwei Schritte des Algorithmus 8.2 (fully pivoted ACA) mit einer 4×4-Matrix vorrechnen. Sei

$$B = \begin{pmatrix} 3 & 3 & 4 & 5 \\ 2 & 5 & 9 & 9 \\ 2 & 2 & 9 & 8 \\ 7 & 7 & 6 & 6 \end{pmatrix}.$$

Dann beginnen wir den Algorithmus mit der Initialisierung und setzen
$i_0 = 0, S_0 = 0, R_0 = B$.
k=1:
Es ergibt sich $i_1 = 1$, $j_1 = 4$ und damit $\gamma_1^{-1} = 5$. Damit können wir nun die Hilfsvektoren $\underline{u}_1$ und $\underline{v}_1$ angeben.

$$\underline{u}_1^T := (R_0\underline{e}_4)^T = \begin{pmatrix} 5 & 9 & 8 & 6 \end{pmatrix}, \quad \underline{v}_1^T := \underline{e}_1^T R_0 = \begin{pmatrix} 3 & 3 & 4 & 5 \end{pmatrix}$$

Mit diesen Vektoren können wir nun die Dyade für die Approximation berechnen. Es ist

$$D_1 := \gamma_1 \underline{u}_1 \underline{v}_1^T = \begin{pmatrix} 3 & 3 & 4 & 5 \\ 5.4 & 5.4 & 7.2 & 9 \\ 4.8 & 4.8 & 6.4 & 8 \\ 3.6 & 3.6 & 4.8 & 6 \end{pmatrix}.$$

Damit ergibt sich

$$R_1 = R_0 - D_1 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ -3.4 & -0.4 & 1.8 & 0 \\ -2.8 & -2.8 & 2.6 & 0 \\ 3.4 & 3.4 & 1.2 & 0 \end{pmatrix}$$

und

$$S_1 = D_1.$$

Für den Fehler der Approximation gilt $\|R_1\|_F = 7.871467$ oder $\frac{\|R_1\|_F}{\|B\|_F} = 0.32883$.

k=2:

Es ergibt sich: $i_2 = 2$, $j_2 = 1$ und $\gamma_2^{-1} = -3.4$,

$$\underline{u}_2^T := (R_1 \underline{e}_1)^T = \begin{pmatrix} 0 & -3.4 & -2.8 & 3.4 \end{pmatrix}, \quad \underline{v}_2^T := \underline{e}_2^T R_1 = \begin{pmatrix} -3.4 & -0.4 & 1.8 & 0 \end{pmatrix},$$

$$D_2 := \gamma_2 \underline{u}_2 \underline{v}_2^T = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 3.4 & 0.4 & -1.8 & 0 \\ 2.8 & 0.33 & -1.49 & 0 \\ -3.4 & -0.4 & 1.8 & 0 \end{pmatrix},$$

$$R_2 = R_1 - D_2 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & -2.47 & 1.12 & 0 \\ 0 & 3 & 3 & 0 \end{pmatrix},$$

$$S_2 = S_1 + D_2 = \begin{pmatrix} 3 & 3 & 4 & 5 \\ 2 & 5 & 9 & 9 \\ 2 & 4.47 & 7.88 & 8 \\ 7 & 4 & 3 & 6 \end{pmatrix}.$$

Der Fehler im zweiten Schritt ist $\|R_2\|_F = 5.03517$ oder $\frac{\|R_2\|_F}{\|B\|_F} = 0.210347$.

8.6 Fehlerabschätzungen

Kennt man nicht alle Einträge der Matrix, ist der exakte Approximationsfehler nicht bekannt. Um den Fehler unserer Approximation der Matrix-Einträge abschätzen zu können, werden wir diesen mit der Folge $\{r_k\}_k$ in Beziehung bringen und dann unter Benutzung der Eigenschaft des Kerns κ die Größe des Fehlers abschätzen. Dazu müssen wir für das Funktionensystem $\Psi_1, ..., \Psi_{m_p}$ aus Abschnitt 8.3 ein Funktionensystem wählen, für das der Interpolationsfehler bekannt ist. Daher verwenden wir

$$\Psi_i(x) = x^i = x_1^{i_1} \cdots x_d^{i_d}, \quad i \in \mathbb{N}_0^d \text{ mit } \|i\|_\infty \leq p - 1$$

die Produkte der Monome. Entsprechend wählen wir

$$m_p := p^d$$

als die Dimension des aufgespannten Raums Ψ. Bei dieser Wahl überträgt sich die Eindeutigkeit der Interpolation aus dem Eindimensionalen, die Bedingung $det[\psi_j(\zeta_i)]_{ij} \neq 0$ wird also erfüllt. Dabei verwenden wir für die bei der Konstruktion der Folge $\{s_k\}_k$ eingeführte Punktmenge $\{\zeta_1, ..., \zeta_{m_p}\}$ das Tensor-Produkt

$$\zeta_i = (\zeta_{i_1}, ..., \zeta_{i_d}), \quad \mathbf{i} \in \mathbb{N}^d \quad \text{mit} \quad \|i\|_\infty \leq p$$

der Nullstellen der Tschebeyscheff-Polynome in $[-a, a]$

$$\xi_\nu = a \, cos\left(\frac{\pi}{2}\frac{2\nu - 1}{p}\right), \quad \nu = 1, ..., p.$$

Nun können wir die Interpolation $\mathbf{L}_p[\mathcal{L}_j\kappa](y)$ und deren Fehler bestimmen und den Rest $\mathcal{L}_j r_k$ der Approximation mit dem Rest der polynomialen Interpolation

$$E_p[\mathcal{L}_j\kappa](y) = \mathcal{L}_j\kappa(y) - \mathbf{L}_p[\mathcal{L}_j\kappa](y)$$

in Beziehung setzen. Damit sind wir imstande, Konvergenzaussagen für den Algorithmus anzugeben.

Lemma 8.8. *Sei (D_1, D_2) ein η-zulässiges Paar von Mengen und κ ein asymptotisch glatter Kern.*
Im Fall der Punkteauswertungsmatrizen

$$a_{ij} = \kappa(x_j, y_i), \qquad i = 1, ..., m, j = 1, ..., n$$

mit $x_j \in D_1$ und $y_j \in D_2$ gilt für $p \geq g$, dass

$$|(R_k)_{ij}| \leq C_p dist^g(D_1, D_2)\eta^p, \quad 0 \leq \eta \leq \frac{1}{4}\sqrt{d}.$$

Im Fall der Kollokationsmatrizen

$$a_{ij} = \int_D \kappa(x, y_i)\phi_j(x)ds_x \qquad i = 1, ..., m, j = 1, ..., n$$

mit $supp(\phi_j) \subseteq D_1, \|\phi_j\|_\infty = 1$ und $y_j \in D_2$ gilt für $p \geq g$, dass

$$|(R_k)_{ij}| \leq C_p dist^g(D_1, D_2)\mu(D_1)\eta^p, \quad 0 \leq \eta \leq \frac{1}{4}\sqrt{d}.$$

Dabei ist $R_k := A - S_k$.

Beweis. Siehe [2, S. 92-96] □

8.7 Ein Fehlerschätzer

Da wir die ursprüngliche Matrix nicht berechnen, können wir den Fehler nicht exakt bestimmen. Wir können deshalb nur einen Fehlerschätzer verwenden, um die Genauigkeit zu kontrollieren. Wegen

$$R_{m_{p+1}} = R_{m_p} - \sum_{l=m_p+1}^{m_{p+1}} \underline{u}_l \underline{v}_l^T$$

und weil wir erwarten, dass $\left\|R_{m_{p+1}}\right\|_F$ um eine Ordnung kleiner ist als die Norm $\left\|R_{m_p}\right\|_F$, können wir den Wert

$$\left\| \sum_{l=m_p+1}^{m_{p+1}} \underline{u}_l \underline{v}_l^T \right\|_F$$

als eine Approximation an $\left\|R_{m_p}\right\|_F$ verwenden.
Die Berechnung ergibt:

$$\left\| \sum_{l=m_p+1}^{m_{p+1}} \underline{u}_l \underline{v}_l^T \right\|_F^2 = \sum_{l,k=m_p+1}^{m_{p+1}} \left(\sum_{i=1}^{m} (u_l)_i (u_k)_i \right) \left(\sum_{j=1}^{n} (v_l)_j (v_k)_j \right).$$

Die Auswertung benötigt $(m+n)(m_{p+1} - m_p)^2$ Operationen.

8.8 Der Gesamtfehler der Approximation

Im folgenden werden wir den Gesamtfehler der Approximation der Matrix $A \in \mathbb{R}^{N \times N}$ abschätzen und die Gesamtkomplexität des Algorithmus 8.3 angeben.
Sei $\mathcal{M}_\eta$ die Menge aller η-zulässigen Blöcke. Sei $M \in \mathcal{M}_\eta$ eine $n \times m$ Matrix. Dann gilt für den Fehler (vergleiche Lemma 8.8)

$$\left\| M - \widetilde{M} \right\|_F \leq \sqrt{mn}\, C_p\, dist^g(D_1, D_2) \eta^p.$$

Für den Gesamtfehler der Approximation erhalten wir damit

$$\left\| A - \widetilde{A} \right\|_F^2 = \sum_{M \in \mathcal{M}_\eta} \left\| M - \widetilde{M} \right\|_F^2 \leq \sum_{M \in \mathcal{M}_\eta} mn\, C_p^2\, dist^{2g}(D_1, D_2) \eta^{2p}.$$

Mit geometrischen Überlegungen (siehe [2]) ergibt sich der Zusammenhang

$$dist^g(D_1, D_2) \leq c_1 N^{-c_2 g}, \quad g \in \mathbb{R}$$

und damit die Abschätzung

$$\left\| A - \widetilde{A} \right\|_F \leq c_1 C_p N^{1 - c_2 g} \eta^p.$$

Wir wählen $\eta > 0$ also so, dass

$$\eta^p = \frac{\varepsilon}{c_1 C_p} N^{1 - c_2 g}.$$

Dies garantiert uns, dass $\left\| A - \widehat{A} \right\|_F \leq \varepsilon$. Man kann noch zeigen (siehe [2]) das die Gesamtkomplexität des Algorithmus 8.3, für beliebig kleine Zahlen $\alpha > 0$, $O(N^{1+\alpha} \varepsilon^{-\alpha})$ ist.

Kapitel 9

$\mathcal{H}^2$-Matrizen

Eugen Rempel

Für lineare Gleichungssysteme mit dünnbesetzten $n \times n$-Matrizen sind Iterationsmethoden bekannt, deren Aufwand bei geeigneter Vorkonditionierung $\mathcal{O}(n)$ ist. Für vollbesetzte Matrizen liegt ein anderer Fall vor. Standardverfahren weisen einen Speicheraufwand von $\mathcal{O}(n^2)$ auf und benötigen $\mathcal{O}(n^2)$ Operationen für die Matrix-Vektor-Multiplikation. Die Matrix-Matrix-Multiplikation ist sogar von der $\mathcal{O}(n^3)$-Komplexität.
Vollbesetzte Matrizen entstehen einerseits bei der Diskretisierung von Randintegralgleichungen in der BEM, andererseits bei der Invertierung von dünnbesetzten FEM-Matrizen. In beiden Fällen sind die so entstandenen Matrizen M mit einem Diskretisierungsfehler F_d behaftet, somit kann man die Matrix M durch eine "günstigere" Matrix $\widetilde{M}$ ersetzen, falls der Fehler $\|M - \widetilde{M}\|$ kleiner als F_d ist.
Die hierarchischen Matrizen (abgekürzte Schreibweise: $\mathcal{H}$-Matrizen) definieren eine Menge der Approximationsmatrizen $\widetilde{M}$, die oben besprochen wurden. Die $\mathcal{H}$-Matrizen haben folgende Eigenschaften:

- Der Speicheraufwand ist von $\mathcal{O}(n \log^\alpha n)$-Komplexität.
- Die Matrix-Vektor-Multiplikation ist von $\mathcal{O}(n \log^\alpha n)$-Komplexität.
- In der Regel sind Produkte und Summen von $\mathcal{H}$-Matrizen keine $\mathcal{H}$-Matrizen mehr, können aber mit einem Aufwand von $\mathcal{O}(n \log^\alpha n)$ durch solche approximiert werden.
- Das Gleiche gilt für die Inverse einer $\mathcal{H}$-Matrix.

Der logarithmische Faktor kann durch eine weitere Verbesserung vermieden werden, die zu den so genannten $\mathcal{H}^2$-Matrizen führt, die in diesem Vortrag untersucht werden (siehe [9]). Die Ziffer 2 in der Bezeichnung weist auf zwei hierarchische Strukturen hin: einerseits die Hierarchie der Matrizen und andererseits die Hierarchie der zugrundeliegenden Basiselemente.

9.1 Definition der $\mathcal{H}^2$-Matrizen

Man betrachtet weiterhin den Clusterbaum T, welcher bei der Definition der $\mathcal{H}$-Matrizen eingeführt wurde. Zusätzlich zu der Hierarchie auf den Clustern, die durch den Clusterbaum

T auferlegt wurde, führt man eine zweite hierarchische Struktur ein, die durch die Vektoren $\underline{a}_i$ und $\underline{c}_i$ der Darstellung der Niedrig-Rang-Matrizen $\mathcal{R}$

$$\sum_{i=1}^{k} \underline{a}_i \, \underline{c}_i^\top \tag{9.1}$$

bestimmt wird.

9.1.1 Hierarchische Basen für Reihen- bzw. Zeilenvektoren der $\mathcal{H}^2$-Matrizen

Bis jetzt konnte eine Niedrig-Rang-Matrix durch beliebige Vektoren $\underline{a}_i$ und $\underline{c}_i$ gebildet werden. Nun hält man sie für den Block b der Partition P_2 der Indexmenge $\mathcal{I} \times \mathcal{I}$ fest. Ein beliebiger Block b hat die Form $b = \tau \times \sigma$ mit Clustern $\tau, \sigma \in T$. Man verlangt, dass die Vektoren $(\underline{a}_i)_i$ nur vom Cluster τ abhängen. Entsprechend hängen die Vektoren $(\underline{c}_i)_i$ nur vom Cluster σ ab. Mit $\mathcal{V}_{\underline{a}}(\tau)$ (analog $\mathcal{V}_{\underline{c}}(\sigma)$) wird der Aufspann der Vektoren $\underline{a}_i^\tau$ bezeichnet:

$$\mathcal{V}_{\underline{a}}(\tau) = \operatorname{span}\{\underline{a}_i^\tau : 1 \le i \le k(\tau)\}, \tag{9.2}$$

$$\mathcal{V}_{\underline{c}}(\sigma) = \operatorname{span}\{\underline{c}_i^\sigma : 1 \le i \le k(\sigma)\}. \tag{9.3}$$

Dabei ist $k : P_2 \to \mathbb{N}$ eine Funktion, die jedem Block $b \in P_2$ seinen Rang $k(b)$ zuordnet. Ferner definiert man für Cluster τ und σ, die zum gleichen Level gehören, $k(\tau) = k(\sigma) := k(b)$.
Die zugehörigen Niedrig-Rang-Matrizen $\mathcal{R}$ sind Elemente des Tensorraumes

$$\mathcal{V}(b) = \mathcal{V}_{\underline{a}}(\tau) \times \mathcal{V}_{\underline{c}}(\sigma) \tag{9.4}$$

für $b = \tau \times \sigma$. Die Speicheranforderungen für $\mathcal{R}$-Matrizen sind kleiner als bei den $\mathcal{H}$-Matrizen. Da die Vektoren $\underline{a}_i, \underline{c}_i$ fest sind, braucht man nur die Koeffizienten bzgl. der Basis $\{\underline{a}_i^\tau \underline{c}_j^{\sigma,\top}\}$ von $\mathcal{V}(b)$ zu speichern.

Bemerkung 9.1. Seien die Räume (9.2) und (9.3) gegeben. Dann braucht man $k(\tau) \cdot k(\sigma)$ Koeffizienten $\xi_{i,j}$, um die $\mathcal{R}$-Matrix $\sum_{i,j} \xi_{i,j} \underline{a}_i^\tau \underline{c}_j^{\sigma,\top}$ zu bilden.

9.1.2 Restriktionen

Man betrachte einen Cluster $\tau \in T$, welcher kein Blatt ist. Wir bezeichnen mit τ', τ'' die Söhne von τ. Die Zerlegung $\tau = \tau' \cup \tau''$ liefert eine Blockpartitionierung des Vektors $\underline{a}_i^\tau$ in Vektoren

$$(\underline{a}_{i,\nu}^\tau)_{\nu \in \tau'} = R_{\underline{a}}^{\tau',\tau} \underline{a}_i^\tau, \quad (\underline{a}_{i,\nu}^\tau)_{\nu \in \tau''} = R_{\underline{a}}^{\tau'',\tau} \underline{a}_i^\tau. \tag{9.5}$$

Der Restriktionsoperator $R_{\underline{a}}^{\tau',\tau}$ bildet den vollen Vektor auf Blockvektoren ab. Wenn man die Einträge von $\underline{a}_i^\tau$ so anordnet, dass die ersten Einträge gerade die Einträge von $\underline{a}_i^{\tau'}$ sind, kann man schreiben

$$\underline{a}_i^\tau = \begin{pmatrix} R_{\underline{a}}^{\tau',\tau} \underline{a}_i^\tau \\ R_{\underline{a}}^{\tau'',\tau} \underline{a}_i^\tau \end{pmatrix}. \tag{9.6}$$

Die Restriktionen $R_{\underline{c}}^{\sigma',\sigma}$ ist analog definiert und liefert

$$\underline{c}_j^\sigma = \begin{pmatrix} R_{\underline{c}}^{\sigma',\sigma} \underline{c}_j^\sigma \\ R_{\underline{c}}^{\sigma'',\sigma} \underline{c}_j^\sigma \end{pmatrix}. \tag{9.7}$$

9.1.3 Konsistenzbedingungen

Seien $\tau, \tau', \tau'' \in T$ wie vorher definiert. Die Konsistenzbedingungen zwischen den Räumen $\mathcal{V}_{\underline{a}}(\tau)$ und $\mathcal{V}_{\underline{a}}(\tau'), \mathcal{V}_{\underline{a}}(\tau'')$ lauten

$$\mathcal{V}_{\underline{a}}(\tau') = R_{\underline{a}}^{\tau',\tau}\mathcal{V}_{\underline{a}}(\tau), \quad \mathcal{V}_{\underline{a}}(\tau'') = R_{\underline{a}}^{\tau'',\tau}\mathcal{V}_{\underline{a}}(\tau). \tag{9.8}$$

Eine ähnliche Relation gilt auch für die Räume $\mathcal{V}_{\underline{c}}(\sigma), \mathcal{V}_{\underline{c}}(\sigma'), \mathcal{V}_{\underline{c}}(\sigma'')$. Eine wichtige Schlussfolgerung aus $\mathcal{V}_{\underline{a}}(\tau') \supseteq R_{\underline{a}}^{\tau',\tau}\mathcal{V}_{\underline{a}}(\tau)$ ist in der folgenden Bemerkung erläutert.

Bemerkung 9.2. Man braucht die Vektoren $\underline{a}_i^\tau$ nicht explizit zu speichern. Auf Grund der Darstellung für $1 \le i \le k(\tau)$

$$(\underline{a}_{i,\nu}^\tau)_{\nu\in\tau'} = R_{\underline{a}}^{\tau',\tau}\underline{a}_i^\tau = \sum_{j=1}^{k(\tau')} \alpha_{ij}^{\tau,\tau'}\underline{a}_j^{\tau'} \tag{9.9}$$

reicht es aus, die Koeffizienten $\alpha_{ij}^{\tau,\tau'}$ und die analog definierten $\alpha_{ij}^{\tau,\tau''}$ zu speichern.

Die andere Richtung $\mathcal{V}_{\underline{a}}(\tau') \subseteq R_{\underline{a}}^{\tau',\tau}\mathcal{V}_{\underline{a}}(\tau)$ impliziert die nächste Folgerung.

Bemerkung 9.3. Die Abbildung k ist monoton bzgl. der Knoten, d.h. ist τ' ein Sohn von τ, so gilt $k(\tau') \le k(\tau)$. Hängt k nur vom Level ab, so gilt $k_{l+1} \le k_l$.

9.1.4 Normalform

Es ist möglich, aus den Vektoren $\{\underline{a}_i^\tau : 1 \le i \le k(\tau), \tau \in T\}$ die Basen folgendermaßen zu wählen, so dass gilt:

- Die Summe in der Darstellung (9.9) der Basis eines gröberen Levels läuft nur über $j = 1, \ldots, \min(i, k(\tau'))$, d.h.

$$R_{\underline{a}}^{\tau',\tau}\underline{a}_i^\tau = \sum_{j=1}^{\min(i,k(\tau'))} \alpha_{ij}^{\tau,\tau'}\underline{a}_j^{\tau'}. \tag{9.10}$$

  Analog gilt

$$R_{\underline{c}}^{\sigma',\sigma}\underline{c}_j^\sigma = \sum_{i=1}^{\min(j,k(\sigma'))} \gamma_{ji}^{\sigma,\sigma'}\underline{c}_i^{\sigma'} \quad \text{für } 1 \le j \le k(\sigma). \tag{9.11}$$

- Die Vektoren können ferner orthonormal gewählt werden, d.h. es gilt $\left(\underline{a}_i^\tau, \underline{a}_j^\tau\right) = \delta_{i,j}$ für $1 \le i, j \le k(\tau)$ mit dem Kronecker-Symbol $\delta_{i,j}$.
- Es ist sogar besser, wenn die Basen $\{\underline{a}_i^\tau\}$ und $\{\underline{c}_j^\tau\}$ von $\mathcal{V}_{\underline{a}}^\tau$ und $\mathcal{V}_{\underline{c}}^\tau$, welche verschieden sein können, bi-orthonormal gewählt werden, d.h.

$$\left(\underline{a}_i^\tau, \underline{c}_j^\tau\right) = \delta_{i,j}. \tag{9.12}$$

- Für die Blätter gilt $k(\tau) = 1$ und $\underline{a}_1^\tau = \underline{c}_1^\tau = (1)$.

9.1.5 Konstanter Rang

Sei der Rang $k(\tau) = k_{const}$ konstant für alle Cluster τ. Da $\{\underline{a}_i^\tau : 1 \le i \le k_{const}\}$ eine Basis ist, gilt $\dim \mathcal{V}_{\underline{a}} = k_{const}$ und deshalb muß für die Anzahl der Elemente im Cluster $\#\tau \ge k(\tau)$ gelten. Somit kann $k(\tau) = k_{const}$ nicht für kleinere Blöcke des Levels l erfüllt werden, denn es ist $\#\tau = 2^{p-l} < k_{const}$ für genügend große l. Man muss also für $\tau \in T$ verlangen

$$k(\tau) = \min\{k_{const}, 2^{p-level(\tau)}\}. \tag{9.13}$$

9.1.6 Nichtkonstanter Rang

Die Formel (9.13) legt nahe, für kleine Blöcke einen kleinen Rang $k(\tau)$ zu wählen. Es stellt sich als nützlich heraus, den Rang $k(\tau)$ nach einer Regel zu wählen, etwa

$$k(\tau) = \alpha(p - level(\tau)) + \beta, \tag{9.14}$$

mit $\alpha, \beta \ge 1$.
Wie es sich herausstellen wird, verschlechtert diese Wahl nicht die Approximationseigenschaft. Ferner impliziert ein kleiner Rang auch weniger Kosten für die Speicherung der Koeffizienten und für verschiedene arithmetische Operationen (vgl. (9.9)). Wichtig dabei ist folgende Relation, die besagt, dass die Summe aller mit $k(\tau)^\gamma$ gewichteter Knoten für alle $\gamma \in \mathbb{N}$ linear beschränkt in n ist,

$$\sum_{\tau \in T} k(\tau)^\gamma = \sum_{l=0}^{p} 2^l (\alpha(p-l) + \beta)^\gamma \sim n. \tag{9.15}$$

Als Beispiel führen wir diese Rechnung für $\gamma = 1$ durch:

$$\begin{aligned}
\sum_{l=0}^{p} 2^l(\alpha(p-l)+\beta) &= \alpha \sum_{l=0}^{p} 2^l(p-l) + \beta \sum_{l=0}^{p} 2^l = \alpha p \sum_{l=0}^{p} 2^l - \alpha \sum_{l=0}^{p} 2^l l + \beta \sum_{l=0}^{p} 2^l \\
&= \alpha p(2^{p+1}-1) - \alpha(2^{p+1}(p-1)+2) + \beta(2^{p+1}-1) \\
&= \alpha 2^{p+1} + \beta 2^{p+1} - 2\alpha - \beta = 2n\alpha + 2n\beta - 2\alpha - \beta \\
&= (2\alpha + 2\beta)n - 2\alpha - \beta.
\end{aligned}$$

9.2 Komplexitätsabschätzungen

Wir beweisen zuerst, dass die Speicherung von $\mathcal{O}(n)$-Komplexität ist. Anschließend zeigen wir, dass auch die Matrix-Vektor-Multiplikation in $\mathcal{O}(n)$ Schritten durchgeführt werden kann.

9.2.1 Speicheranforderungen

Nach der Bemerkung 9.2 sind folgende Matrizen

$$A^{\tau,\tau'} := (\alpha_{ij}^{\tau,\tau'})_{1 \le i \le k(\tau), 1 \le j \le k(\tau')} \quad \text{für } \tau, \tau' \in T, \tau' \text{ Sohn von } \tau, \tag{9.16}$$

$$C^{\sigma,\sigma'} := (\gamma_{ji}^{\sigma,\sigma'})_{1 \le j \le k(\sigma), 1 \le i \le k(\sigma')} \quad \text{für } \sigma, \sigma' \in T, \sigma' \text{ Sohn von } \tau, \tag{9.17}$$

von der Größe $k_l \times k_{l+1}$ zu speichern, wobei $l = level(\tau) = level(\sigma) \in \{0, 1, \ldots, p-1\}$ ist. Es gibt 2^{l+1} verschiedene Paare von τ, τ' mit $l = level(\tau)$, wobei τ' ein Sohn von τ ist. Mit $k_l \leq \alpha(p-l) + \beta$ für $\alpha, \beta \geq 1$ gilt für den Speicheraufwand

$$\sum_{l=0}^{p-1} 2^{l+1} k_l k_{l+1} \leq \sum_{l=0}^{p-1} 2^{l+1} (\alpha(p-l) + \beta)(\alpha(p-l-1) + \beta). \tag{9.18}$$

Mit der Relation (9.15) erhält man die nächste Schlussfolgerung.

Bemerkung 9.4. Der Speicherplatz für die Transfermatrizen $A^{\tau,\tau'}, C^{\sigma,\sigma'}$ ist proportional zu n.

Man betrachte ferner die Koeffizientenmatrix

$$Z^b = (\xi_{ij}^b)_{1 \leq i,j \leq k(b)} \quad \text{für } b = \tau \times \sigma \in P_2 \tag{9.19}$$

der Blockmatrix $M^b = \sum_{i,j} \xi_{i,j} \underline{a}_i^\tau \underline{c}_j^{\sigma,\top}$ (vgl. Bemerkung 9.1). Es gilt folgendes Ergebnis.

Lemma 9.1. *Der Speicherplatz für die Koeffizientenmatrizen $Z^b, b \in P_2$ ist durch $\mathcal{O}(n)$ beschränkt.*

Beweis. Man bemerkt zuerst, dass die Matrix $Z^b, b \in P_2$ insgesamt $k(b)^2$ Einträge hat. Somit sind $\sum_{b \in P_2} k(b)^2$ Koeffizienten zu speichern. Mit der Definition $P_2(l) := \{b \in P_2 : level(b) = l\}$ zeigt sich, dass für die Anzahl der Elemente auf dem Level l folgende Relation $\#P_2(l) \sim 2^l$ gilt [7]. Entsprechend (9.15) ist somit $\sum_{b \in P_2} k(b)^2 = \sum_{l=0}^{p} 2^l (\alpha(p-l) + \beta)^2 \sim n$. □

9.2.2 Algorithmus für die schnelle Matrix-Vektor-Multiplikation

Die schnelle Matrix-Vektor-Multiplikation erfolgt in drei Schritten:

- Vorwärtstransformation,
- Block-Multiplikation,
- Rückwärtstransformation.

Wie wir sehen werden, brauchen alle drei Schritte $\mathcal{O}(n)$ Operationen. Somit besitzt dieser Algorithmus für die Matrix-Vektor-Multiplikation lineare Komplexität.

Blockmatrix-Vektor-Multiplikation

Man betrachtet die Multiplikation eines Blocks $M^b, b = \tau \times \sigma \in P_2$ mit dem Vektor $\underline{\hat{x}}_\sigma \in \mathcal{V}_{\underline{a}}(\sigma)$. Man bezeichne mit $\underline{\hat{\mathbf{x}}}_\sigma$ den Koeffizientenvektor bzgl. der Basis $\{\underline{a}_i^\sigma : 1 \leq i \leq k(\sigma)\}$, d.h.

$$\underline{\hat{x}}_\sigma = \sum_{i=1}^{k(\sigma)} \underline{\hat{\mathbf{x}}}_{\sigma,i} \underline{a}_i^\sigma. \tag{9.20}$$

Lemma 9.2. *(a) Sei $b = \tau \times \sigma \in P_2, M^b = \sum_{i,j} \xi_{i,j} \underline{a}_i^\tau \underline{c}_j^{\sigma,\top}$ mit $Z^b = (\xi_{ij}^b)_{1 \leq i,j \leq k(b)}$ und $\underline{\hat{x}}_\sigma \in \mathcal{V}_{\underline{a}}(\sigma)$. Dann hat $\underline{y}_\tau = M^b \underline{\hat{x}}_\sigma$ den Koeffizientenvektor $\underline{\mathbf{y}}_\tau = Z^b \underline{\hat{\mathbf{x}}}_\sigma$ bzgl. der Basis $\{\underline{a}_i^\tau : 1 \leq i \leq k(\tau)\}$.*
(b) $\underline{x}_\sigma$ besitze die Zerlegung $\underline{x}_\sigma = \underline{\hat{x}}_\sigma + \underline{x}_\sigma^\perp$ mit $\underline{\hat{x}}_\sigma \in \mathcal{V}_{\underline{a}}(\sigma)$ und $\underline{x}_\sigma^\perp \perp \mathcal{V}_{\underline{c}}(\sigma)$. Dann ist $M^b \underline{x}_\sigma = M^b \underline{\hat{x}}_\sigma$ und Teilbehauptung (a) gilt für $M^b \underline{\hat{x}}_\sigma$.

Beweis. (a): Es gilt

$$M^b\hat{\underline{x}}_\sigma = \left(\sum_{i,j} \xi_{i,j}\underline{a}_i^\tau \underline{c}_j^{\sigma,\top}\right)\left(\sum_{l=1}^{k(\sigma)} \hat{\underline{\mathbf{x}}}_{\sigma,l}\underline{a}_l^\sigma\right) = \sum_{i,j} \xi_{ij}^b \hat{\underline{\mathbf{x}}}_{\sigma,j}\underline{a}_i^\tau, \tag{9.21}$$

denn es ist auf Grund der Bi-Orthogonalität $\left(\underline{c}_j^\sigma, \sum_l \hat{\underline{\mathbf{x}}}_{\sigma,l}\underline{a}_l^\sigma\right) = \hat{\underline{\mathbf{x}}}_{\sigma,j}$.
(b): Folgt mit (a). □

Vorwärtstransformation

Sei ein Vektor $\underline{x}$ gegeben. Man möchte das Produkt $M\underline{x}$ ausrechnen, wobei M eine $\mathcal{H}^2$-Matrix ist. Um Lemma 9.2 anwenden zu können, stellt man den Blockvektor $\underline{x}_\sigma = (\underline{x}_i)_{i\in\sigma}$ als die Summe $\underline{x}_\sigma = \hat{\underline{x}}_\sigma + \underline{x}_\sigma^\perp$ dar, wobei der Koeffizientenvektor $\hat{\underline{\mathbf{x}}}_\sigma$ von $\hat{\underline{x}}_\sigma$ zur Verfügung stehen muß. Da die Matrix M Blöcke von allen Levels enthält, braucht man die Koeffizientenvektoren $\hat{\underline{\mathbf{x}}}_\sigma$ für alle $\sigma \in T$.
Man führt die Schreibweise $T(l) := \{\tau \in T : level(\tau) = l\}$ für alle $0 \le l \le p$ ein.
Folgende Rechenvorschrift beginnt beim Level $l = p$ an und geht bis zur Wurzel ($l = 0$):

- Beginn beim Level $l = p$. Sei $\sigma = \{s\} \in T(p)$. Der eindimensionale Blockvektor $\underline{x}_\sigma = (\underline{x}_i)_{i\in\sigma} = (\underline{x}_s)$ ist zu dem Koeffizientenvektor $\underline{\mathbf{x}}_\sigma$ identisch, da die Basis nur den Einheitsvektor enthält. Somit ist $\hat{\underline{\mathbf{x}}}_\sigma$ bekannt, und es gilt $\underline{x}_\sigma^\perp = \underline{0}$.

- Rekursion $l+1 \to l$ für $0 \le l < p$. Man nimmt an, dass die Koeffizientenvektoren $\hat{\underline{\mathbf{x}}}_\tau$ des ersten Summanden in $\underline{x}_\tau = \hat{\underline{x}}_\tau + \underline{x}_\tau^\perp$ für alle $\tau \in T(l+1)$ bekannt sind. Man konstruiert den Vektor $\hat{\underline{\mathbf{x}}}_\sigma$ für $\sigma \in T(l)$ folgendermaßen. Seien $\sigma', \sigma'' \in T(l+1)$ die Söhne von σ. Die Zerlegung vom Level $l+1$ liefert

  $$\underline{x}_\sigma = \underline{x}_\sigma^I + \underline{x}_\sigma^{II} \quad \text{mit } \underline{x}_\sigma^I := \begin{bmatrix}\hat{\underline{x}}_{\sigma'}\\ \hat{\underline{x}}_{\sigma''}\end{bmatrix} \text{ und } \underline{x}_\sigma^{II} := \begin{bmatrix}\underline{x}_{\sigma'}^\perp\\ \underline{x}_{\sigma''}^\perp\end{bmatrix}. \tag{9.22}$$

  Auf Grund von

  $$\left(\underline{c}_j^\sigma, \underline{x}_\sigma^{II}\right) = \left(R_{\underline{c}}^{\sigma',\sigma}\underline{c}_j^\sigma, \underline{x}_{\sigma'}^\perp\right) + \left(R_{\underline{c}}^{\sigma'',\sigma}\underline{c}_j^\sigma, \underline{x}_{\sigma''}^\perp\right) = 0$$

  gilt, dass der zweite Summand $\underline{x}_\sigma^{II}$ in (9.22) orthogonal zu $\mathcal{V}_{\underline{c}}(\sigma)$ ist. Der erste Summand muß weiter in $\underline{x}_\sigma^I = \hat{\underline{x}}_\sigma + \underline{x}_\sigma^{III}$ zerlegt werden, wobei $\hat{\underline{x}}_\sigma \in \mathcal{V}_{\underline{a}}(\sigma)$ und $\underline{x}_\sigma^{III} \perp \mathcal{V}_{\underline{c}}(\sigma)$ sind. Dann gilt mit $\underline{x}_\sigma^\perp := \underline{x}_\sigma^{II} + \underline{x}_\sigma^{III}$ die gewünschte Zerlegung $\underline{x}_\sigma = \hat{\underline{x}}_\sigma + \underline{x}_\sigma^\perp$.

  Die Einträge des Koeffizientenvektors $\hat{\underline{\mathbf{x}}}_\sigma$ sind durch $\hat{\underline{\mathbf{x}}}_{\sigma,j} = \left(\underline{c}_j^\sigma, \underline{x}_\sigma^I\right) = \left(\underline{c}_j^\sigma, \hat{\underline{x}}_\sigma\right)$ bestimmt. Unter Benutzung der Konstruktionsvorschrift von $\underline{c}_j^\sigma$ in (9.7) erhält man

  $$\hat{\underline{\mathbf{x}}}_{\sigma,j} = \left(\underline{c}_j^\sigma, \underline{x}_\sigma^I\right) = \left(\begin{bmatrix}R_{\underline{c}}^{\sigma',\sigma}\underline{c}_j^\sigma\\ R_{\underline{c}}^{\sigma'',\sigma}\underline{c}_j^\sigma\end{bmatrix}, \begin{bmatrix}\hat{\underline{x}}_{\sigma'}\\ \hat{\underline{x}}_{\sigma''}\end{bmatrix}\right) = \left(R_{\underline{c}}^{\sigma',\sigma}\underline{c}_j^\sigma, \hat{\underline{x}}_{\sigma'}\right) + \left(R_{\underline{c}}^{\sigma'',\sigma}\underline{c}_j^\sigma, \hat{\underline{x}}_{\sigma''}\right).$$

  Mit der Formel (9.11) für $R_{\underline{c}}^{\sigma',\sigma}\underline{c}_j^\sigma$ und der Darstellung $\hat{\underline{x}}_{\sigma'} = \sum_t \hat{\underline{\mathbf{x}}}_{\sigma',t}\underline{a}_t^{\sigma'}$ gilt nun

  $$\left(R_{\underline{c}}^{\sigma',\sigma}\underline{c}_j^\sigma, \hat{\underline{x}}_{\sigma'}\right) = \Big(\sum_{i=1}^{\min(j,k(\sigma'))} \gamma_{ji}^{\sigma,\sigma'}\underline{c}_i^{\sigma'}, \sum_{t=1}^{k(\sigma')} \hat{\underline{\mathbf{x}}}_{\sigma',t}\underline{a}_t^{\sigma'}\Big) = \sum_{i=1}^{\min(j,k(\sigma'))} \gamma_{ji}^{\sigma,\sigma'}\hat{\underline{\mathbf{x}}}_{\sigma',i} = (C^{\sigma,\sigma'}\hat{\underline{\mathbf{x}}}_{\sigma'})_j,$$

mit der Matrix $C^{\sigma,\sigma'}$, die in (9.16) definiert wurde. Da der zweite Summand eine analoge Darstellung besitzt, gilt nun

$$\underline{\hat{\mathbf{x}}}_\sigma = C^{\sigma,\sigma'}\underline{\hat{\mathbf{x}}}_{\sigma'} + C^{\sigma,\sigma''}\underline{\hat{\mathbf{x}}}_{\sigma''}. \tag{9.23}$$

Nach Voraussetzung sind die Koeffizientenvektoren $\underline{\hat{\mathbf{x}}}_{\sigma'}, \underline{\hat{\mathbf{x}}}_{\sigma''}$ bekannt. Somit braucht man nur zwei Matrix-Vektor-Multiplikationen durchzuführen, um den Koeffizientenvektor $\underline{\hat{\mathbf{x}}}_\sigma$ für $\sigma \in T(l)$ auszurechnen. Dabei ist die Anzahl der Operationen in (9.23) proportional zu der Anzahl der Einträge der $k_l \times k_{l+1}$-Matrizen $C^{\sigma,\sigma'}$ und $C^{\sigma,\sigma''}$.

Bemerkung 9.4 impliziert somit die nächste Schlussfolgerung.

Bemerkung 9.5. Die Rechenvorschrift (9.23) für alle $\sigma \in T(l), l = p-1, \ldots, 0$ umfasst $\mathcal{O}(n)$ Operationen und liefert den Koeffizientenvektor $\underline{\hat{\mathbf{x}}}_\sigma$ für alle $\sigma \in T$.

Block-Multiplikation

Für alle Blöcke $M^b, b = \tau \times \sigma \in P_2$ sind die Zwischenresultate $\underline{y}^b_\tau = M^b \underline{x}_\sigma$ auszurechnen, d.h. entsprechend der Bemerkung 9.2 sind die Koeffizientenvektoren $\underline{\mathbf{y}}^b_\tau = Z^b\underline{\hat{\mathbf{x}}}$ von $\underline{y}^b_\tau$ auszurechnen.
Die Anzahl der Operationen für alle Produkte $\underline{\mathbf{y}}^b_\tau = Z^b\underline{\hat{\mathbf{x}}}, b = \tau \times \sigma \in P_2$ ist wiederum zu der Anzahl der Einträge der Matrizen Z^b proportional. Somit gilt nach Lemma 9.1 folgendes Ergebnis.

Bemerkung 9.6. Die Matrix-Vektor-Multiplikationen $\underline{\mathbf{y}}^b_\tau = Z^b\underline{\hat{\mathbf{x}}}$ verlangen für alle Blöcke der Form $b = \tau \times \sigma \in P_2$ insgesamt $\mathcal{O}(n)$ Operationen.

Rückwärtstransformation

Zuletzt fasst man alle in der vorhergehenden Phase erhaltenen Teilergebnisse zusammen. Dabei benutzt man die Rückwärtstransformation, die am Level $l = 0$ beginnt und bis zum Level p läuft. Für jedes Level l berechnet man $\underline{\mathbf{y}}_\tau$ für alle $\tau \in T(l)$, wobei $\underline{\mathbf{y}}_\tau$ der Koeffizientenvektor für $\underline{y}_\tau$ ist, der durch

$$\underline{y}_{\tau,i} := \sum_{b'=\tau'\times\sigma' \in P_2 \text{ mit } \tau' \supseteq \tau} (\underline{y}^{b'}_{\tau'})_i \quad \text{für } i \in \tau$$

definiert ist. Man beachte, dass alle $\tau' \supseteq \tau$ zu einem $T(l')$ mit $l' \leq l$ gehören. Wie zuvor bezeichnet man mit $P_2(l)$ die Menge $\{b \in P_2 \text{ mit } level(b) = l\}$. Im Detail wird die Rückwärtstransformation wie folgt durchgeführt:

- Beginn beim Level $l = 0$. Der einzige Block auf dem Level $l = 0$, $\mathcal{I}\times\mathcal{I}$, ist nicht zulässig. Somit ist der Beginn durch
$$\underline{\mathbf{y}}_\mathcal{I} := \underline{0}$$
initiiert.

- Rekursion $l \to l+1$ für $0 \leq l < p$. Man setzt voraus, dass die Koeffizientenvektoren $\underline{\mathbf{y}}_\tau$ für alle $\tau \in T(l)$ bereits bekannt sind. Seien $\tau', \tau'' \in T(l+1)$ die Söhne eines bestimmten Clusters $\tau \in T(l)$. Der dem Koeffizientenvektor $\underline{\mathbf{y}}_\tau$ entsprechende Vektor $\underline{y}_\tau$ ist durch
$$\underline{y}_\tau = \sum_i \underline{\mathbf{y}}_{\tau,i}\underline{a}^\tau_i = \begin{bmatrix} R^{\tau',\tau}_{\underline{a}}\underline{y}_\tau \\ R^{\tau'',\tau}_{\underline{a}}\underline{y}_\tau \end{bmatrix}$$

nach der Definition von $R_{\underline{a}}^{\tau',\tau}$ und $R_{\underline{a}}^{\tau'',\tau}$ gegeben. Die Koeffizientenvektoren $\hat{\underline{\mathbf{y}}}_{\tau'}$ und $\hat{\underline{\mathbf{y}}}_{\tau''}$ von $R_{\underline{a}}^{\tau',\tau}\underline{y}_\tau$ und $R_{\underline{a}}^{\tau'',\tau}\underline{y}_\tau$ sind durch

$$\hat{\underline{\mathbf{y}}}_{\tau'} = (A^{\tau,\tau'})^\top \underline{\mathbf{y}}_\tau, \qquad \hat{\underline{\mathbf{y}}}_{\tau''} = (A^{\tau,\tau''})^\top \underline{\mathbf{y}}_\tau \tag{9.24}$$

definiert, wie man anhand von

$$R_{\underline{a}}^{\tau',\tau}\underline{y}_\tau = R_{\underline{a}}^{\tau',\tau}\sum_i \underline{\mathbf{y}}_{\tau,i}\underline{a}_i^\tau = \sum_i \underline{\mathbf{y}}_{\tau,i} R_{\underline{a}}^{\tau',\tau}\underline{a}_i^\tau = \sum_i \underline{\mathbf{y}}_{\tau,i}\alpha_{ij}^{\tau,\tau'}\underline{a}_i^{\tau'}$$

nachrechnen kann. Als Nächstes muss man alle Beiträge von Blöcken des Levels $l+1$ addieren

$$\underline{\mathbf{y}}_{\tau'} = \hat{\underline{\mathbf{y}}}_{\tau'} + \sum_{\sigma' \text{ mit } b'=\tau'\times\sigma'\in P_2(l+1)} \underline{\mathbf{y}}_{\tau'}^{b'}. \tag{9.25}$$

Bemerkung 9.7. Entsprechend Lemma 9.1 beträgt die Anzahl der in den Rückwärtstransformationen (9.24) und (9.25) involvierten Operationen

$$2\sum_{l=0}^{p-1} k_l k_{l+1}\#T(l+1) + \sum_{l=0}^{p-1} k_{l+1}\#P_2(l+1) = 2\sum_{l=0}^{p-1} k_l k_{l+1}2^{l+1} + \sum_{l=0}^{p-1} k_{l+1}\#P_2(l+1) \sim n.$$

- Ergebnis beim Level $l = p$. Die resultierenden Koeffizientenvektoren $\underline{\mathbf{y}}_\tau$ für alle einelementigen Cluster $\tau = \{i\} \in T(p)$ stimmen mit den Komponenten y_i von $\underline{y} = M\underline{x}$ überein. Somit ist die Matrix-Vektor-Multiplikation berechnet.

9.2.3 Andere arithmetische Matrixoperationen

Anders als bei allgemeinen $\mathcal{H}$-Matrizen kann die Summe zweier $\mathcal{H}^2$-Matrizen (bei der gleichen Partitionierung und den gleichen hierarchischen Basen) exakt berechnet werden. Da nur die Matrizen Z^b für alle $b \in P_2$ zu addieren sind, beträgt der Aufwand $\mathcal{O}(n)$.
Wir besprechen hier nicht die Matrix-Matrix-Multiplikation. Allerdings kann man anmerken, dass das Produkt zweier Blöcke relativ billig auszurechnen ist, da man die Skalarprodukte der Form $\left(\underline{c}_j^\sigma, \underline{a}_i^\sigma\right)$ zu bestimmen hat. Dies lässt sich nach (9.12) ausrechnen.

9.2.4 Konstanter Rang

Wir präsentieren zuletzt folgendes Ergebnis [9]:

Korollar 9.3. *Sei $k_{const} \in \{1,\dots,n\}$. Der Rang $k(\tau)$ sei nach* (9.13) *gewählt. Der Speicherbedarf für $A^{\tau,\tau'}, C^{\sigma,\sigma'}, Z^b$ (vgl. Bemerkung 9.4 und Lemma 9.1) und der Aufwand für die Matrix-Vektor-Multiplikation betragen $\mathcal{O}(n \cdot k_{const})$.*

9.3 Approximation der Integraloperatoren durch $\mathcal{H}^2$-Matrizen

9.3.1 Approximation des Kerns

Man möchte jetzt das Rüstzeug der $\mathcal{H}^2$-Matrizen am Problem der Approximation von Integraloperatoren anwenden: man versuche, den diskreten Integraloperator $\mathbf{K}^h$ durch eine $\mathcal{H}^2$-Matrix

zu approximieren (siehe [4]). Dabei ist der Integraloperator $\mathcal{K}$ durch

$$\mathcal{K}[u](\underline{x}) := \int_\Gamma \kappa(\underline{x},\underline{y})u(\underline{y})\,d\underline{y}$$

definiert, wobei $\Gamma \subseteq \mathbb{R}^d$ ein Gebiet oder eine Mannigfaltigkeit ist, und $\kappa : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ eine Kernfunktion ist. Ferner diskretisiert man $\mathcal{K}$ nach Galerkin, indem man eine Funktionenbasis $(\Psi_i)_{i\in\mathcal{I}}$ verwendet. Man erhält eine Matrix $\mathbf{K}^h \in \mathbb{R}^{\mathcal{I}\times\mathcal{I}}$ mit

$$\mathbf{K}^h_{ij} := \int_\Gamma \Psi_i(\underline{x}) \int_\Gamma \kappa(\underline{x},\underline{y})\Psi_j(\underline{y})\,d\underline{y}\,d\underline{x} \quad \text{für alle } i,j \in \mathcal{I}.$$

Falls die Kernfunktion $\kappa(\cdot,\cdot)$ folgende Form besitzt

$$\kappa(\underline{x},\underline{y}) = \sum_{\nu=1}^{k}\sum_{\mu=1}^{l} s_{\nu,\mu} f_\nu(\underline{x}) g_\mu(\underline{y}) \tag{9.26}$$

für Funktionen $(f_\nu)_{\nu=1}^k$ und $(g_\mu)_{\mu=1}^l$ und Koeffizienten $(s_{\nu,\mu})_{\nu,\mu=1}^{k,l}$, findet man

$$\mathbf{K}_{ij} = \sum_{\nu=1}^{k}\sum_{\mu=1}^{l} s_{\nu,\mu} \underbrace{\int_\Gamma \Psi_i(\underline{x}) f_\nu(\underline{x})\,d\underline{x}}_{=:F_{i\nu}} \underbrace{\int_\Gamma \Psi_j(\underline{y}) g_\mu(\underline{y})\,d\underline{y}}_{=:G_{j\mu}} = (FSG^\top)_{ij},$$

d.h. die Matrix $\mathbf{K}$ hat die Form $\mathbf{K} = FSG^\top$.
Leider besitzt die typische Kernfunktion eines Randintegraloperators nicht die Form (9.26). Somit kann man die obere Darstellung nicht global verwenden. Allerdings kann man sie lokal benutzen. Die Standardkernfunktionen sind *asymptotisch glatt*, d.h. sie erfüllen eine Abschätzung der Form

$$\left|\partial^\alpha_{\underline{x}}\partial^\beta_{\underline{y}}\kappa(\underline{x},\underline{y})\right| \leq C(\alpha+\beta)!(c_0\|\underline{x}-\underline{y}\|)^{-g-|\alpha|-|\beta|} \tag{9.27}$$

für Konstanten $C, c_0 \in \mathbb{R}^+$, beliebige Multiindizes $\alpha,\beta \in \mathbb{N}_0^d$ und einen Parameter $g \in \mathbb{R}^+$, welcher den Singularitätsgrad auf der Diagonale $\underline{x} = \underline{y}$ charakterisiert. Diese Abschätzung impliziert, dass der Kern in den Gebieten, die von der Diagonale entfernt sind, glatt ist.
Eine glatte Funktion kann durch Polynome approximiert werden, und die Polynome besitzen die Darstellung (9.26). Dann kann man den Kern interpolieren.
Um einen einfachen Algorithmus zu erhalten, kann man die Tensorprodukt-Interpolation benutzen. Dafür braucht man achsenparallele Gebiete. Man erhält jetzt für jeden Cluster $\tau \in T_\mathcal{I}$ einen achsenparallelen Würfel $Q^\tau \in \mathbb{R}^d$ mit $\tau \subseteq Q^\tau$. Q^τ wird als *begrenzender Würfel* bezeichnet.
Um die Kernfunktion auf dem Block $\tau\times\sigma \in P_2$ zu approximieren, benutzt man Interpolation auf dem Gebiet $Q^\tau \times Q^\sigma$. Man muss also sicherstellen, dass der Kern in dem Gebiet genügend glatt ist. Man entscheidet dies anhand der sogenannten Zulässigkeitsbedingung

$$\max\{\operatorname{diam}(Q^\tau),\operatorname{diam}(Q^\sigma)\} \leq \eta \operatorname{dist}(Q^\tau,Q^\sigma). \tag{9.28}$$

Man hält einen Interpolationsoperator $\mathfrak{I}^\tau$ von der Ordnung m^τ für jeden begrenzenden Würfel Q^τ mit dazugehörigen Interpolationspunkten $(\underline{x}^\tau_\nu)_{\nu\in M^\tau}$ mit einer Indexmenge M^τ und den

Lagrange-Polynomen $(\mathcal{L}_\nu^\tau)_{\nu\in M^\tau}$ fest. Für einen Block $\tau\times\sigma$ aus der Menge der zulässigen Blöcke P_{fern} definiert man die Kernapproximation

$$\widetilde{\kappa}^{\tau,\sigma}(\underline{x},\underline{y}) := (\mathfrak{I}^\tau\otimes\mathfrak{I}^\sigma)[\kappa](\underline{x},\underline{y}) = \sum_{\nu\in M^\tau}\sum_{\mu\in M^\sigma}\kappa(\underline{x}_\nu^\tau,\underline{x}_\mu^\sigma)\mathcal{L}_\nu^\tau(\underline{x})\mathcal{L}_\mu^\sigma(\underline{y}).$$

Die Partition P_2 beschreibt eine Zerlegung der Menge $\Gamma\times\Gamma$ in zulässige und unzulässige Bereiche. Man definiert die Approximation der Kernfunktion durch lokale Approximationen in allen Fernfeldbereichen:

$$\widetilde{\kappa}(\underline{x},\underline{y}) := \begin{cases}\widetilde{\kappa}^{\tau,\sigma}(\underline{x},\underline{y}) & \text{, falls } \underline{x}\in\tau,\underline{y}\in\sigma,\tau\times\sigma\in P_{fern}\\ \kappa(\underline{x},\underline{y}) & \text{, sonst.}\end{cases}$$

9.3.2 Approximation der Matrizen

Man erhält die Approximationsmatrix $\widetilde{\mathbf{K}}^h$ durch die Diskretisierung von $\widetilde{\kappa}(\cdot,\cdot)$ mit $P_{nah} := P_2\backslash P_{fern}$:

$$\begin{aligned}\widetilde{\mathbf{K}}_{ij}^h &= \int_\Gamma\Psi_i(\underline{x})\int_\Gamma\widetilde{\kappa}(\underline{x},\underline{y})\Psi_j(\underline{y})\,d\underline{y}\,d\underline{x} = \sum_{\tau\times\sigma\in P_2}\int_\tau\Psi_i(\underline{x})\int_\sigma\widetilde{\kappa}(\underline{x},\underline{y})\Psi_j(\underline{y})\,d\underline{y}\,d\underline{x}\\ &= \sum_{\tau\times\sigma\in P_{fern}}\int_\tau\Psi_i(\underline{x})\int_\sigma\widetilde{\kappa}^{\tau,\sigma}(\underline{x},\underline{y})\Psi_j(\underline{y})\,d\underline{y}\,d\underline{x}\\ &\quad+\sum_{\tau\times\sigma\in P_{nah}}\int_\tau\Psi_i(\underline{x})\int_\sigma\kappa(\underline{x},\underline{y})\Psi_j(\underline{y})\,d\underline{y}\,d\underline{x}.\end{aligned}$$

Somit erhält man

$$\widetilde{\mathbf{K}}^h = \sum_{\tau\times\sigma\in P_{fern}}\mathbf{V}^\tau\mathbf{S}^{\tau,\sigma}\mathbf{V}^{\sigma\top}+\sum_{\tau\times\sigma\in P_{nah}}\hat{\mathbf{S}}^{\tau,\sigma}$$

wobei die Matrizen $\mathbf{V}^\tau\in\mathbb{R}^{\mathcal{I}\times M^\tau},\mathbf{S}^{\tau,\sigma}\in\mathbb{R}^{M^\tau\times M^\sigma}$ und $\hat{\mathbf{S}}^{\tau,\sigma}\in\mathbb{R}^{\mathcal{I}\times\mathcal{I}}$ durch

$$\mathbf{V}_{i\nu}^\tau := \int_\tau\mathcal{L}_\nu^\tau(\underline{x})\Psi_i(\underline{x})\,d\underline{x},\quad \mathbf{S}_{\nu,\mu}^{\tau,\sigma} := \kappa(\underline{x}_\nu^\tau,\underline{x}_\mu^\sigma)\quad\text{und}$$
$$\hat{\mathbf{S}}_{ij}^{\tau,\sigma} := \int_\tau\Psi_i(\underline{x})\int_\sigma\kappa(\underline{x},\underline{y})\Psi_j(\underline{y})\,d\underline{y}\,d\underline{x}$$

definiert sind.
Wir bemerken, dass für alle $i\in\mathcal{I}$ mit $\tau\cap\mathrm{supp}\Psi_i=\emptyset$ die Gleichung $\mathbf{V}_{i\nu}^\tau=0$ für $\nu\in M^\tau$ gilt. Somit ist $\mathbf{V}^\tau$ eine dünnbesetzte Matrix. Desweiteren entspricht der Nahfeldanteil mit den Blöcken $\hat{\mathbf{S}}^{\tau,\sigma}$ einer dünnbesetzten Matrix.
Sei $\tau\in T_\mathcal{I}$ und τ' ein Sohn von τ. Ist die Ordnung $m^\tau\le m^{\tau'}$, so ist $\mathcal{L}_\nu^\tau=\mathfrak{I}^{\tau'}[\mathcal{L}_\nu^\tau]$ und deshalb gilt

$$\begin{aligned}\mathbf{V}_{i\nu}^\tau &= \int_\tau\mathcal{L}_\nu^\tau(\underline{x})\Psi_i(\underline{x})\,d\underline{x} = \sum_{\tau'\text{ Sohn von }\tau}\int_{\tau'}\mathcal{L}_\nu^\tau(\underline{x})\Psi_i(\underline{x})\,d\underline{x}\\ &= \sum_{\tau'\text{ Sohn von }\tau}\mathfrak{I}^{\tau'}[\mathcal{L}_\nu^\tau]\Psi_i(\underline{x})\,d\underline{x} = \sum_{\tau'\text{ Sohn von }\tau}\sum_{\nu'\in M^{\tau'}}\mathcal{L}_\nu^\tau(\underline{x}_{\nu'}^{\tau'})\int_{\tau'}\mathcal{L}_{\nu'}^{\tau'}(\underline{x})\Psi_i(\underline{x})\,d\underline{x}\\ &= \left(\sum_{\tau'\text{ Sohn von }\tau}\mathbf{V}^{\tau'}\mathbf{B}^{\tau',\tau}\right)_{i\nu}\end{aligned}$$

für Matrizen $\mathbf{B}^{\tau',\tau} \in \mathbb{R}^{M^{\tau'} \times M^{\tau}}$, die durch

$$\mathbf{B}^{\tau',\tau}_{\nu'\nu} := \mathcal{L}^{\tau}_{\nu}(\underline{x}^{\tau'}_{\nu'})$$

definiert sind. Somit erfüllt unsere Clusterbasis die Konsistenzbedingung (9.8). Dies impliziert, dass $\widetilde{\mathbf{K}}$ eine $\mathcal{H}^2$-Approximationsmatrix für $\mathbf{K}$ mit Blockrängen $k(\tau) = \#M^{\tau} = (m^{\tau})^d$ ist.

Lemma 9.4. *[9] Durch Kombination der Standardabschätzungen der Interpolation mit der Bedingung der asymptotischen Glattheit* (9.27) *und der Zulässigkeitsbedingung* (9.28) *erhält man folgende Fehlerabschätzung*

$$\begin{aligned}\left|\kappa(\underline{x},\underline{y}) - \widetilde{\kappa}(\underline{x},\underline{y})\right| &\leq C(c_1\eta)^m \min_{\tau\times\sigma\in P_{fern}} dist(Q^{\tau},Q^{\sigma})^{-g} \\ &\leq C(c_1\eta)^m \Big(\frac{\eta}{\min_{\tau\in T_{\mathcal{I}}} diam(Q^{\tau})}\Big)^g\end{aligned}$$

mit Konstanten $C, c_1 \in \mathbb{R}^+$. Dies bedeutet, dass die Approximation durch eine $\mathcal{H}^2$-Matrix in m exponentiell konvergiert, falls $c_1\eta < 1$ gilt.

9.3.3 Interpolation mit verschiedener Ordnung

Falls man die Interpolation mit konstanter Ordnung m benutzt, ist der Fehler der Approximation durch eine $\mathcal{H}^2$-Matrix von der $\mathcal{O}((c_1\eta)^m)$-Komplexität, während der Aufwand für die Matrix-Vektor-Multiplikation $\mathcal{O}(nm^d)$ ist.
Die Zahl n steuert den Diskretisierungsfehler, und um sicherzustellen, dass unser Approximationsfehler von der gleichen Größenordnung ist, wie der Diskretisierungsfehler, muss man m proportional zu $\log(n)$ wählen. Somit ist die Approximation mit der konstanten Ordnung von der $\mathcal{O}(n\log^d n)$-Komplexität. Man möchte dies aber noch weiter verbessern.
Man versucht, auf den größeren Clustern einen größeren Matrizenrang zu verwenden und auf den kleineren Clustern den Rang kleiner zu machen. Dies verursacht ein Problem: Bei der Definition der Approximation durch eine $\mathcal{H}^2$-Matrix braucht man die Ungleichung $m^{\tau} \leq m^{\tau'}$, um die Bedingung (9.8) nachzuweisen. Man muss sich also überlegen, wie man Clusterbasen alternativ definieren kann.
Dabei geht man folgendermaßen vor : Man approximiert Lagrange-Polynome $\mathcal{L}^{\tau}_{\nu}$ durch $\hat{\mathcal{L}}^{\tau}_{\nu}$ so, dass folgende Identität gilt

$$\hat{\mathcal{L}}^{\tau}_{\nu} = \sum_{\nu'\in M^{\tau'}} \mathbf{B}^{\tau',\tau}_{\nu'\nu}\hat{\mathcal{L}}^{\tau'}_{\nu'}.$$

Dies impliziert (9.8). Dafür führt man einen Operator der stückweisen Interpolation $\hat{\mathfrak{I}}^{\tau}$ ein:

$$\hat{\mathfrak{I}}^{\tau}[u](\underline{x}) := \begin{cases} u(\underline{x}) & \text{, falls } \tau \text{ ein Blatt ist} \\ \hat{\mathfrak{I}}^{\tau'}[\mathfrak{I}^{\tau}[u]](\underline{x}) & \text{, falls } x \in \tau', \tau' \text{ein Sohn von } \tau \\ 0 & \text{, sonst.} \end{cases}$$

Man definiert nun die Approximation der Lagrange-Polynome durch

$$\hat{\mathcal{L}}^{\tau}_{\nu} := \hat{\mathfrak{I}}^{\tau}[\mathcal{L}^{\tau}_{\nu}].$$

Nach dieser Definition gilt für alle Blätter $\hat{\mathcal{L}}_\nu^\tau = \mathcal{L}_\nu^\tau$. Für einen Cluster $\tau \in T_\mathcal{I}$, der kein Blatt ist, und seinen Sohn $\tau' \ni \underline{x}$ findet man

$$\begin{aligned}\hat{\mathcal{L}}_\nu^\tau(\underline{x}) = \hat{\mathfrak{I}}^\tau[\mathcal{L}_\nu^\tau](\underline{x}) = \hat{\mathfrak{I}}^{\tau'}[\hat{\mathfrak{I}}^\tau[\mathcal{L}_\nu^\tau]](\underline{x}) &= \sum_{\nu' \in M^{\tau'}} \mathcal{L}_\nu^\tau(\underline{x}_{\nu'}^{\tau'}) \hat{\mathfrak{I}}^{\tau'}[\mathcal{L}_{\nu'}^{\tau'}](\underline{x}) \\ &= \sum_{\nu' \in M^{\tau'}} \mathbf{B}_{\nu'\nu}^{\tau',\tau} \hat{\mathcal{L}}_{\nu'}^{\tau'}(\underline{x}).\end{aligned}$$

Benutzt man diese Approximation der Lagrange-Polynome, so kann man eine Approximation nicht konstanter Ordnung für die Kernfunktion $\kappa(\cdot,\cdot)$ konstruieren. Es lässt sich zeigen [9], dass der Approximationsfehler zum Diskretisierungsfehler proportional ist und der Aufwand für die Matrix-Vektor-Multiplikation $\mathcal{O}(n)$ ist.

Literaturverzeichnis

[1] AMD Athlon Processor x86 Code Optimization Guide, 2002.

[2] M. Bebendorf: Effiziente numerische Lösung von Randintegralgleichungen unter Verwendung von Niedrigrang–Matrizen. Dissertation, Universität des Saarlandes, Saarbrücken, 2000.

[3] M. Bebendorf, W. Hackbusch: Existence of $\mathcal{H}$–matrix approximants to the inverse FE–matrix of elliptic operators with L^∞–coefficients. Numer. Math. 95 (2003) 1–28.

[4] S. Börm: $\mathcal{H}^2$–matrices – Multilevel methods for the approximation of integral operators. Bericht Nr. 7, Max–Planck–Institut für Mathematik in den Naturwissenschaften, Leipzig, 2003.

[5] S. Börm, L. Grasedyck, W. Hackbusch: Hierarchical Matrices. Lecture Notes no. 21, Max–Planck–Institut für Mathematik in den Naturwissenschaften, Leipzig, 2003.

[6] L. Grasedyck: Theorie und Anwendungen Hierarchischer Matrizen. Dissertation, Christian–Albrecht–Universität, Kiel, 2001.

[7] W. Hackbusch: A sparse matrix arithmetic based on $\mathcal{H}$–matrices. Part I: Introduction to $\mathcal{H}$–matrices. Computing 62 (1999) 89–108.

[8] W. Hackbusch, B. N. Khoromskij: A sparse $\mathcal{H}$–matrix arithmetic. Part II: Application to multi–dimensional problems. Computing 64 (2000) 21–47.

[9] W. Hackbusch, B. N. Khoromskij, S. A. Sauter: On $\mathcal{H}^2$–matrices. In: Lectures on Applied Mathematics (H.–J. Bungartz, R. H. W. Hoppe, C. Zenger eds.), Springer, Berlin, pp. 9–29, 2000.

[10] R. Hyde: The Art of Assembly Programming. http://webster.cs.ucr.edu, 1996.

[11] M. Kowarschik, U. Rüde, C. Weiss: Data layout optimizations for variable coefficient multigrid. In: Proceedings of the 2002 International Conference on Computational Science, Lecture Notes in Computer Science, vol. 2331, Springer, Amsterdam, pp. 642–651, 2002.

[12] M. Lintner: Lösung der 2D Wellengleichung mittels hierarchischer Matrizen. Dissertation, TU München, 2002.

[13] M. Lintner: The eigenvalue problem for the 2D Laplacian in $\mathcal{H}$–matrix arithmetic and application to the heat and wave equation. Computing 72 (2004) 293–323.

[14] T. Ottmann, P. Widmayer: Algorithmen und Datenstrukturen. Spektrum Akademischer Verlag, 2002.

[15] W. Press, S. Teukolsky, W. Vetterling, B. Flannery: Numerical Recipes in C. Cambrigde University Press, 1997.

[16] T. Sauer, Y. Xu: On multivariate Lagrange interpolation. Math. Comp. 64 (1995) 1147–1170.

[17] O. Steinbach: Numerische Näherungsverfahren für elliptische Randwertprobleme. Finite Elemente und Randelemente. B. G. Teubner, Stuttgart, Leipzig, Wiesbaden, 2003.

[18] G. W. Stewart: Simultaneous iteration for computing invariant subspaces of non–hermitian matrices. Numer. Math. 25 (1976) 123–136.

[19] S. Zimmer: Numerische Simulation. Vorlesungsskript, Universität Stuttgart, 2004.

Olaf Steinbach (ed.)
Pfaffenwaldring 57
70569 Stuttgart
Germany
E-Mail: steinbach@mathematik.uni-stuttgart.de
WWW: http://www.ians.uni-stuttgart.de/LstAngMath/Steinbach/

Erschienene Preprints ab Nummer 2004/001

Komplette Liste: `http://preprints.ians.uni-stuttgart.de`

2004/001 *Geis, W., Mishuris, G., Sändig, A.-M.*: 3D and 2D asymptotic models for piezoelectric stack actuators with thin metal inclusions

2004/002 *Klimke, A., Wohlmuth, B., Willner, K.*: Computing expensive multivariate functions of fuzzy numbers using sparse grids

2004/003 *Klimke, A., Wohlmuth, B., Willner, K.*: Uncertainty modeling using efficient fuzzy arithmetic based on sparse grids: applications to dynamic systems

2004/004 *Flemisch, B., Mair, M., Wohlmuth, B.*: Nonconforming discretization techniques for overlapping domain decompositions

2004/005 *Sändig, A.-M.*: Vorlesung Mathematik für Informatiker und Softwaretechniker I, WS 2003/2004

2004/006 *Bürger, R., Karlsen, K. H., Towers, J. D.*: Closed-form and finite difference solutions to a population balance model of grinding mills

2004/007 *Berres, S., Bürger, R., Tory, E. M.*: Applications of Polydisperse Sedimentation Models

2004/008 *Bürger, R., Karlsen, K. H., Towers, J. D.*: A model of continuous sedimentation of flocculated suspensions in clarifier-thickener units

2004/009 *Bürger, R., Karlsen, K. H., Towers, J. D.*: Mathematical model and numerical simulation of the dynamics of flocculated suspensions in clarifier-thickeners

2004/010 *Lehrstühle: Wendland, Wohlmuth, Abteilungen: Gekeler. Sändig*: Jahresbericht 2003

2004/011 *Sändig, A.-M. (Hrsg.), Knees, D. (Hrsg.)*: Nichtlineare Funktionalanalysis mit Anwendungen in der Festkörpermechanik

2004/012 *Wendland, W.L.*: Vorlesungsskript Partielle Differentialgleichungen

2004/013 *Steinbach, O. (ed.)*: Seminarbericht: Hierarchische Matrizen