Effiziente Behandlung von Integraloperatoren bei populationsdynamischen Modellen

Dissertation

zur Erlangung des akademischen Grades

doctor rerum naturalium
(Dr. rer. nat.),

genehmigt durch die Fakultät für Mathematik

der Otto-von-Guericke-Universität Magdeburg

von Diplom-Mathematiker Jürgen Koch
geboren am 25.6.1968 in Leipzig

Gutachter:
Prof. Dr. Wolfgang Hackbusch
Prof. Dr. Lutz Tobiska
Prof. Dr. Gerald Warnecke

Eingereicht am: 1. Juli 2005

Verteidigung am: 20. Dezember 2005

Inhaltsverzeichnis

1. Einführung und Übersicht

Populationsdynamische Modelle spielen eine wichtige Rolle in Wissenschaft und Industrie. Die hier betrachteten Modelle werden durch eine Populationsbilanzgleichung dargestellt, der ein System von Integrodifferentialgleichungen zu Grunde liegt. Die auftretenden Differentialoperatoren diskretisiert man meist mit Hilfe von impliziten, die Integraloperatoren hingegen explizit.

Den Integraloperatoren liegen hierbei im Wesentlichen Volterrasche Integralgleichungen 1. Art zugrunde. Nach Diskretisierung, etwa durch das Kollokations- oder das Galerkin-Verfahren, haben wir im 1D-Fall, aufgrund der variablen Grenzen, Matrizen mit Dreiecksstruktur vorliegen, die wir mit einem Vektor von Basiskoeffizienten multiplizieren wollen. Der Aufwand für die numerische Behandlung dieser Matrizen entspricht dabei im Wesentlichen dem Aufwand, der für den Einsatz des zugehörigen populationsdynamischen Modells benötigt wird. Bei naiver Herangehensweise ergibt sich eine Komplexität von $\mathcal{O}(n^2)$ für die Speicherung der Matrizen, sowie für die Ausführung der Matrix-Vektor-Multiplikation, wobei n die Anzahl der Freiheitsgrade bezeichnet. Quadratische Komplexität bedeutet aber einen nicht vertretbaren Aufwand in der Modellierung, wodurch der breite Einsatz populationsdynamischer Modelle in der Praxis bisher verhindert wurde. Die Motivation dieser Arbeit besteht nun darin, neue Methoden für die effiziente numerische Behandlung dieser Integraloperatoren vorzustellen.

In der Literatur wurden dazu eine Reihe von Verfahren vorgeschlagen, die eine Reduzierung des Speicherbedarfs und des Aufwandes bei der Matrix-Vektor-Multiplikation bewirken.

Bei der Wavelet-Methode erreicht man durch eine geschickte Wahl der Ansatzfunktionen, dass ein großer Teil der Matrixeinträge vernachlässigbar klein wird. Damit kann der Aufwand auf $\mathcal{O}(n \log^d n)$, $d > 0$, reduziert werden. Unter Umständen kann man sogar optimale Komplexität beim Speicheraufwand erreichen, siehe dazu etwa Dahmen und Schneider [10] und Schneider [43].

Eine weitere Gruppe von Verfahren macht sich zu Nutze, dass die Kernfunktion an den Stellen, an denen sie glatt ist, durch eine separable Entwicklung ersetzt werden kann. Zu dieser Gruppe gehören u.a. das Multipol-Verfahren, siehe dazu etwa Greengard und Rokhlin [24], [25] sowie Rokhlin [41] und die Panel-Clustering-Methode, siehe etwa Hackbusch und Nowak [28]. Der hier erreichte Aufwand liegt ebenfalls bei $\mathcal{O}(n \log^d n)$, $d > 0$, für Multiplikation und Speicherung.

Das Konzept der $\mathcal{H}$-Matrizen ist eine Weiterentwicklung des Panel-Clustering. Statt einer Partition des Integrationsgebietes benutzt man hier eine Blockzerlegung der Matrix. Man versucht dann die Blöcke als Niedrigrangmatrizen darzustellen, was wiederrum durch separierte Multipol- oder Polynomentwicklung erreicht wird.

Ein rein algebraischer Zugang zur Erzeugung der Blöcke wird als *adaptive cross approximation* in Bebendorf [2] beschrieben. Damit dieses Verfahren effizient ist, muss

allerdings bereits bekannt sein, welche Blöcke eine solche Darstellung ermöglichen.

Wir greifen hier die Ideen auf, die den $\mathcal{H}$-Matrizen zu Grunde liegen und passen sie an unsere Erfordernisse an. Desweiteren nutzen wir den Faltungscharakter eines der Integraloperatoren des Modells aus und kombinieren die Methoden der schnellen Fouriertransformation (FFT) mit dem $\mathcal{H}$-Matrix-Kalkül. Den so dargestellten Matrizen liegen dabei Klassen von Kernfunktionen zur Beschreibung von Kristall- und Tropfenpopulation zugrunde. Dabei konnten wir den Aufwand auf $\mathcal{O}(n)$ bzw. $\mathcal{O}(n \log n)$ reduzieren.

Auf der Grundlage dieser Methoden wurde eine Softwarebibliothek geschrieben, mit der die zu den Kernfunktionen gehörigen diskretisierten Operatoren dargestellt werden können. Diese Bibliothek kann nun zur Lösung der bereits angesprochenen Integrodifferentialgleichungen benutzt werden. Sie stellt eine Erweiterung der Programmbibliothek LIBBEM dar, die ursprünglich von Christian Lage für die Lösung von Randelementproblemen erstellt wurde, siehe Lage [32].

Oft ist das populationsdynamische Modell allein nicht ausreichend zur Beschreibung der Population. Man möchte wissen, wie sich Strömungen innerhalb des Lebensraumes auf die Population auswirken. Im Rahmen eines Gemeinschaftsprojektes mit der Universität Heidelberg wurde ein solches Strömungsmodell von T. Fischer und D. Logashenko auf der Grundlage der Softwarebibliothek UG erstellt, deren Ursprünge von Peter Bastian et.al. stammen, siehe Bastian et.al. [1]. In dieser Arbeit werden wir nicht näher darauf eingehen, wir verweisen hier auf Fischer et.al. [14].

Durch die Kopplung beider Konzepte kann man detailliertere Modelle berechnen, mit welchen noch genauere Vorhersagen möglich sind. Weiterhin stellen die gekoppelten Bibliotheken ein mächtiges Werkzeug bei der Suche nach besseren Modellen bzw. Modellgleichungen sowie nach optimalen Anfangs- und Randbedingungen dar.

Wir werden zunächst eine allgemeine Modellbeschreibung geben und dabei auf die Phänomene eingehen, die innerhalb einer Population stattfinden, sowie auf Phänomene, die sich zwischen Mitgliedern der Population und ihrer Umgebung abspielen. In diesem Zusammenhang werden wir auch die zugehörigen Modellgleichungen repräsentieren und die Frage beantworten, warum es überhaupt sinnvoll ist, nichtlebende Materie, wie Kristalle oder Tropfen, als Populationen zu betrachten. Darüber hinaus werden wir die Funktionsweisen eines Kristallisators bzw. eines Rührkessels kennenlernen. Anschließend werden wir auf die Besonderheiten von Kristall- und Tropfenpopulationen eingehen und spezielle Modellgleichungen angeben.

Das Kapitel 3 gibt zunächst einen Überblick über einige Diskretisierungsverfahren. Danach werden wir die Vorzüge erläutern, welche sich bei der Verwendung separabler Kernfunktionen bieten und geben eine Einführung in den Kalkül der $\mathcal{H}$-Matrizen. Anschließend werden wir zeigen, dass sich dieser Kalkül, mit gewissen Modifikationen, auch auf die Integraloperatoren der populationsdynamischen Modelle anwenden lässt. Dabei gehen wir speziell auch auf den Volterra-Charakter der Integralterme ein. Das Kapitel wird mit der Behandlung des quadratischen Integraloperators abgeschlossen.

Kapitel 4 beginnt mit einer kurzen Einführung in die Theorie objektorientierten Programmierens, welches es dem Anwender ermöglicht, Software flexibel an seine speziellen Bedürfnisse anzupassen. Danach stellen wir die Klassenerweiterungen von LIBBEM vor, die auf der Grundlage der Erkenntnisse aus Kapitel 3 kreiert wurden. Aufgrund

der Beschreibung sollte es dem Anwender möglich sein, eigene Programme zu schreiben oder die Bibliothek in seine eigenen Programme zu integrieren. Die Kopplung mit UG ist nur ein mögliches Einsatzgebiet. Ein kurzes Programmbeispiel soll das Verständnis erleichtern und bildet den Abschluss dieses Kapitels.

Bei der Realisierung der Software haben wir sehr stark die Möglichkeiten ausgenutzt, die uns durch objektorientiertes Programmieren geboten werden. Wir tragen damit den Tatsachen Rechnung, dass man erstens noch keine endgültigen Modellgleichungen zur Verfügung hat und zweitens die Bibliothek gerade der Entdeckung und dem Testen neuer Modellgleichungen dienen soll.

Das Kapitel 5 enthält eine Reihe von numerischen Experimenten, die mit Hilfe unserer Softwarebibliothek erzielt wurden und die unsere Aussagen bzgl. Komplexität und Konvergenz belegen.

In den Anhängen haben wir abschließend noch einmal wichtige Grundlagen aus verschiedenen mathematischen Gebieten zusammengestellt, die für das Verständnis der Arbeit von Nutzen sein können.

Danksagung

An dieser Stelle komme ich der angenehmen Pflicht nach, all denjenigen zu danken, die (direkt oder indirekt) ebenfalls an der Entstehung dieser Arbeit beteiligt waren. Besonders danken möchte ich

- Herrn Prof. Dr. Wolfgang Hackbusch für die Überlassung des Themas sowie für seine Unterstützung,
- Lars Grasedyck und Steffen Börm für zahlreiche Diskussionen und Anregungen sowie für das Korrekturlesen der Arbeit,
- Herrn Prof. Dr.-Ing. Kai Sundmacher für die Möglichkeit am Max-Planck-Institut für Dynamik komplexer technischer Systeme, sozusagen „vor Ort“, meine Arbeit zu verbessern und zu beenden,
- meinen „mathematischen“ und „technischen“ Kollegen aus Leipzig und Magdeburg für die Unterstützung und gute Zusammenarbeit,
- meinen Freunden und meinen Eltern, welche (besonders in der „letzten Phase“ der Arbeit) mich ertragen und leider allzu oft zurückstehen mussten,

allen oben genannten für ihre Geduld.

2. Populationsdynamische Modelle

2.1. Allgemeines

Unter einer Population versteht man im Allgemeinen eine Gesamtheit von Individuen, wie z.B. Bakterien oder Kristallen, die sich in einer bestimmten Umgebung bzw. einem bestimmten Lebensraum aufhalten. Dabei sind die Individuen unter anderem dadurch charakterisiert, dass sie geboren werden, sich vermehren, wachsen und sterben. Außerdem werden Ressourcen verbraucht oder auch erzeugt, die Individuen interagieren mit ihrer Umgebung sowie untereinander. Man versucht diese Eigenschaften in populationsdynamischen Modellen darzustellen, um damit die Gesetzmäßigkeiten innerhalb einer Population besser verstehen zu können bzw. überhaupt erst zu finden. Mit Hilfe dieser Modelle kann man z.B. den Lebensraum der Individuen optimal an bestimmte Erfordernisse anpassen oder gewisse Vorhersagen über die Entwicklung der Population treffen.

In den Modellen interessiert man sich gewöhnlich nicht für das Verhalten eines einzelnen Individuums, sondern für das Verhalten der gesamten Population. Ein Individuum kann durch gewisse charakteristische Größen klassifiziert werden, wie z.B. seine Länge, die Menge seiner gespeicherten Nährstoffe (bei Bakterien) oder seine Form (bei Kristallen). Man beschränkt sich bei der Modellierung auf Wahrscheinlichkeiten, d.h. man arbeitet mit den in Frage kommenden charakteristischen Größen nicht direkt, sondern man betrachtet deren Verteilungs- bzw. Anzahldichtefunktion. Speziell interessiert man sich für die zeitliche Änderung dieser Funktion. Man spricht hier auch allgemein von *Populationsbilanzen*.

Bei den von uns betrachteten Modellen handelt es sich um *disperse Systeme*. Man hat hierbei mindestens zwei Phasen im thermodynamischen Sinne vorliegen und unterscheidet die *disperse* und die *kontinuierliche Phase*. Die disperse Phase stellt die Population dar und die kontinuierliche deren Lebensraum. Man kann sich erstere Phase als kleine, inhomogene, körnige Strukturen vorstellen, die sich feinverteilt innerhalb einer zweiten, homogenen Phase befinden. Wichtig hierbei ist die strukturelle Trennung der Phasen, d.h. zwischen den Phasen ist eine eindeutige Phasengrenze ausgebildet, siehe dazu auch Abb. 2.1. Häufig kommt es zu einem Stoffaustausch zwischen den Phasen. So nehmen z.B. die Individuen Teile der kontinuierlichen Phase auf, diese Teile verschwinden damit aus der kontinuierlichen Phase und gehören nun zur dispersen Phase. Liegen genau zwei Phasen vor, so spricht man von *Zwei-Phasen-Systemen*, bei mehr als zwei Phasen entsprechend von *Mehr-Phasen-Systemen*.

Disperse Systeme stellen eine wichtige Prozessklasse in vielen Fertigungsprozessen dar. Trotz ihrer praktischen Bedeutung sind sie noch unzureichend verstanden und durch Modelle erst in Ansätzen erfasst, Versuche dazu siehe Ramkrishna [38].

In den nachfolgenden Abschnitten werden wir auf die Besonderheiten von Kristalli-

sations- und Emulsionsprozessen eingehen und an ihrem Beispiel die Beschreibung disperser Systeme durch populationsdynamische Modelle vornehmen.

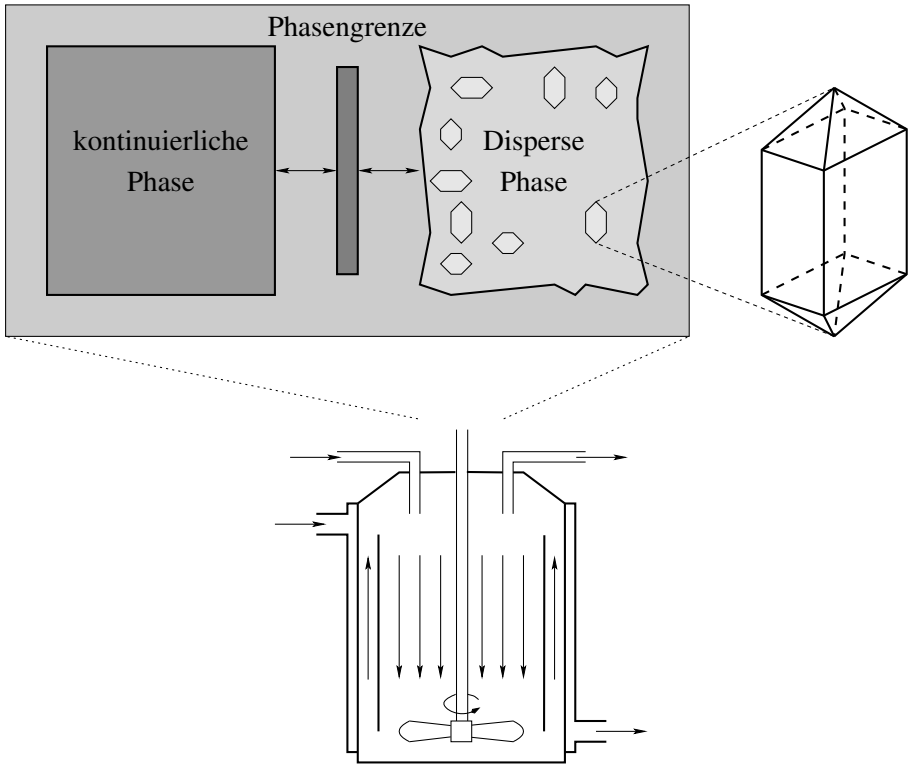


Abbildung 2.1.: Der Kristallisator als disperses System.

2.2. Phänomene bei der Kristallisation

Bei der richtigen Interpretation weisen auch Kristalle die typischen Merkmale einer Population auf.

In unserem Kristallmodell bilden klar erkennbare Kristallstrukturen die disperse Phase, die kontinuierliche Phase wird durch eine so genannte *Nähr-* oder *Mutterlösung* repräsentiert, einer Substanz[1)] die Kristalle in gelöster Form enthält, siehe Abbildung 2.1. Wir haben somit ein Zwei-Phasen-System vorliegen. Für das Modell nehmen wir im Folgenden für die kontinuierliche Phase stets eine Flüssigkeit an, die gelöste Kristalle enthält.

Die zwei Phasen befinden sich in einem speziellen Behälter, dem *Kristallisator*. Bei diesem handelt es sich um einen geschlossenen Metallzylinder, in dessen Inneren sich konzentrisch ein Rohr befindet, das so genannte *Leitrohr*, an seinem Boden befindet sich ein Rührer. Außerdem umgibt den Kristallisator ein Mantel, in dem sich eine

[1)] Der Terminus *Lösung* suggeriert eine Flüssigkeit, was aber nicht notwendig der Fall sein muss, es wird z.B. auch Dampf benutzt.

Kühlflüssigkeit befindet, und es gibt diverse Zu- und Abflüsse. Abbildung 2.2 gibt dazu einen Überblick.

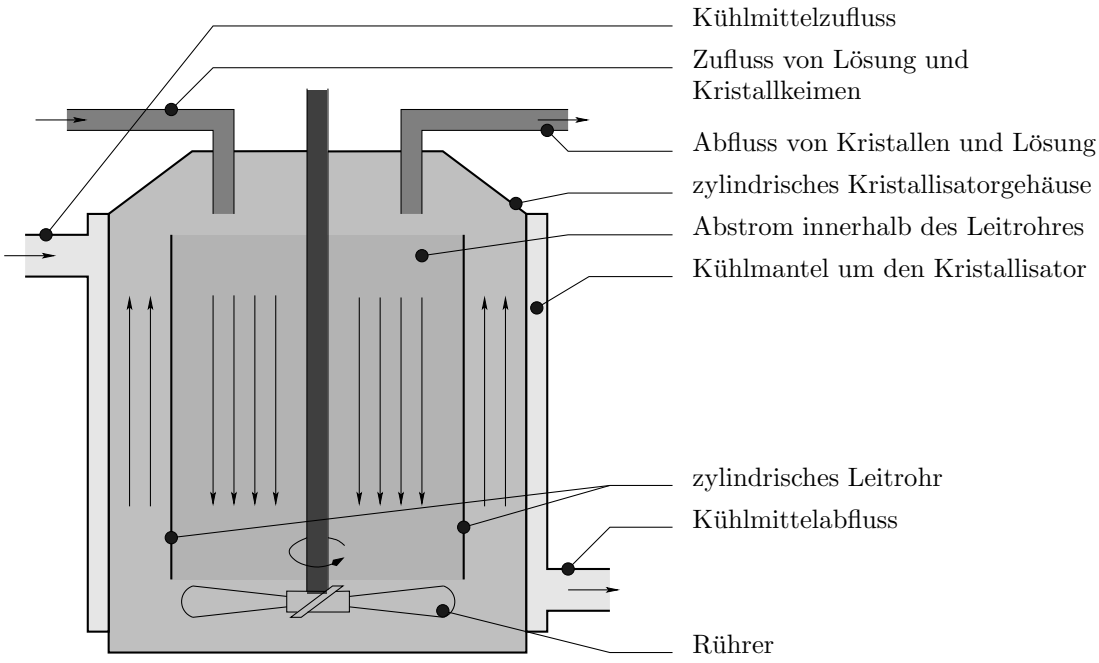


Abbildung 2.2.: Schnitt durch einen Kristallisator.

Die grobe Funktionsweise ist folgende: Man befüllt den Kristallisator mit einer Flüssigkeit, die gelöste Kristalle enthält oder auch bereits Kristallkeime. Durch verschiedene Maßnahmen (z.B. Kühlung) erhöht man die Konzentration (Übersättigung), so dass Kristalle entstehen bzw. wachsen. Dabei unterscheidet man im Wesentlichen drei Arten von Kristallisatoren:

a) *Batch-Kristallisator.* Dieser wird einmal befüllt, man lässt die Lösung auskristallisieren, danach wird der ganze Prozess gestoppt. Der Kristallisator wird entleert und dabei werden die Kristalle entnommen. Danach kann er erneut befüllt werden.

b) *Semi-Batch-Kristallisator.* Dieser funktioniert ähnlich wie der Batch-Kristallisator. Der Unterschied ist, dass mehrfach neue Lösung zugeführt wird. Zur Entnahme der Kristalle muss der Kristallisator wiederum komplett geleert werden.

c) *Kontinuierlicher Kristallisator.* Hier wird ständig neue Lösung zu und verbrauchte Lösung abgeführt. Ist die Produktion einmal angefahren, muss der Kristallisator im Prinzip nicht wieder gestoppt werden, die Kristalle werden während des Betriebes entnommen[1)].

Die Nährlösung wird mithilfe des Rührer während des Betriebes ständig durchmischt, damit sorgt man für eine gleichmäßige Konzentration der Lösung. Ein weiterer erwün-

[1)] sozusagen *kontinuierlich*

schter Vorgang beim Einsatz des Rührers ist, dass Kristalle, die gegen die Rührerblätter schlagen, zerkleinert werden. Dadurch werden neue Kristalle gebildet und das Entstehen zu großer Kristalle wird vermieden.

Beim Kristallisationsprozess treten verschiedene Populationsphänome auf, die eine Interpretation der Kristalle als Population rechtfertigen. Man unterscheidet dabei Ereignisse, die sich zwischen disperser und kontinuierlicher Phase abspielen und Ereignisse, die innerhalb der dispersen Phase stattfinden.

Ereignisse zwischen kontinuierlicher und disperser Phase: Diese sind durch einen Stofftransport zwischen den Phasen gekennzeichnet, hier findet also ein Phasenübergang statt. Die Phänomene sind im einzelnen:

a) *Keimbildung.* In der übersättigten Lösung bilden sich kleine Kristalle bzw. Kristallkeime aus, dieser Vorgang kann als Geburt angesehen werden. Der Stofftransport erfolgt von der kontinuierlichen Phase zur dispersen Phase, d.h. die Konzentration der Lösung nimmt ab, vgl. Abb. 2.3.

b) *Wachstum.* Schon vorhandene Kristalle bauen Teile der Nährlösung in ihr Gitter ein, bzw. Teile des Kristallgitters lösen sich wieder auf (negatives Wachstum), der Stofftransport erfolgt hier somit in beide Richtungen, die Konzentration nimmt entsprechend ab, bzw. zu, siehe Abb. 2.4.

c) *Auflösung.* Wenn die Lösung nicht mehr übersättigt ist, kann es vorkommen, dass sich kleinere Kristalle wieder auflösen. Dies kann als Tod eines Kristalls interpretiert werden. Der Stofftransport erfolgt hier von der dispersen Phase zur kontinuierlichen, somit nimmt die Konzentration der Lösung zu, siehe Abb. 2.5.

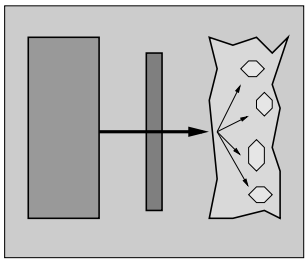


Abbildung 2.3.: Keime.

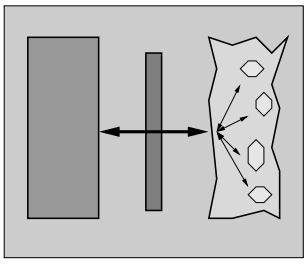


Abbildung 2.4.: Wachstum.

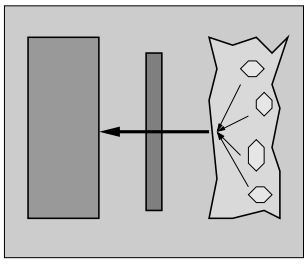


Abbildung 2.5.: Auflösung.

Ereignisse in der dispersen Phase: Da diese Ereignisse auf die disperse Phase beschränkt sind, findet hier auch kein Stofftransport statt, d.h. sie sind unabhängig von der Konzentration der Lösung. Wir unterscheiden hier zwei Phänomene:

a) *Agglomeration.* Zwei oder mehrere kleine Kristalle fügen sich zu einem großen Kristall zusammen. Dieses Ereignis kann als Tod der kleineren bzw. als Geburt des großen Kristalls gewertet werden, siehe Abb. 2.6.

b) *Dispergierung.* Ein Kristall zerfällt in zwei oder mehrere kleinere. Dieser Vorgang findet dadurch statt, dass Rührer und Kristall zusammenstoßen und dabei der Kristall zerschlagen wird. Entstehen ein großer Kristall und viele kleine, spricht man auch von *Abrieb*, siehe Abb. 2.7. Dieses Phänomen kann man als Vermehrung in der Kristallpopulation ansehen.

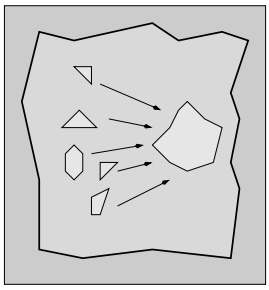


Abbildung 2.6.: Agglomeration.

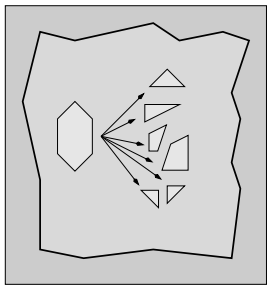


Abbildung 2.7.: Zerfall, Dispergierung.

2.3. Besonderheiten bei den Emulsionstropfen

Im Kristallmodell haben wir im Prinzip alle Populationsphänome kennengelernt, die so oder so ähnlich auch bei anderen Modellen auftreten können. Das Emulsionsmodell bildet hierbei keine Ausnahme. Dennoch ist es für das Modellverständnis wichtig, auf die Besonderheiten der Tropfenpopulation genauer einzugehen.

Wir legen hier ebenfalls ein Zwei-Phasen-System zu Grunde. Die Phasen sind hier zwei Flüssigkeiten, die nur schlecht miteinander mischbar sind, man spricht auch von einem *Flüssig-Flüssig-System.* Als einfaches Beispiel sei hier zum besseren Verständnis die Zusammenstellung Öl in Wasser angeführt. Die Öltropfen repräsentieren die Individuen und bilden somit die disperse Phase, und das Wasser stellt die kontinuierliche Phase dar. Auch hier haben wir eindeutig ausgebildete Phasengrenzen vorliegen.

Der Lebensraum der Tropfenpopulation ist ein *Rührkessel*, welcher im Aufbau dem Kristallisator sehr ähnlich ist. Es gibt einen Rührer, der die Phasen durchmischt. Man hat einen oder mehrere Zuflüsse, durch die neue Individuen in den Tank gelangen, und man hat einen Abfluss, durch welchen man Individuen aus dem Tank entfernt.

Aufgrund der schlechten Mischbarkeit spielen die Interaktionen zwischen den Phasen nur eine untergeordnete Rolle. Deshalb kann man die zugehörigen Phänomene vernachlässigen. Man interessiert sich hier vor allem für die Ereignisse innerhalb der dispersen Phase, also für die Zerteilung und für die Vereinigung von Tropfen. Der Zerteilungsprozess wird ebenfalls als Dispergierung bezeichnet, findet aber nicht notwendigerweise am Rührer statt. Der Vereinigungsprozess wird hier als *Koaleszenz* bezeichnet. Die kontinuierliche Phase stellt hauptsächlich den Aufenthaltsort für die Population dar.

2.4. Übersicht über die Modellgleichungen

2.4.1. Anzahldichtefunktion

Durch die disperse Phase wird die Population beschrieben. Dabei kann der Zustand eines Individuums aus dieser Phase durch bestimmte Größen quantitativ erfasst werden. Traditionell unterscheidet man zwischen *externen* und *internen* Größen bzw. Koordinaten. Den externen Koordinaten entsprechen die Ortskoordinaten $\boldsymbol{r} = (r_1, r_2, r_3)$, welche die geometrische Position des Individuums im Raum beschreiben. Die internen Koordinaten $\boldsymbol{e} = (e_1, e_2, ...)$ beschreiben die wesentlichen Partikeleigenschaften und werden auch als Eigenschaftskoordinaten bezeichnet. Es bezeichne $\mathbb{O}$ den Raum der externen und $\mathbb{E}$ den Raum der internen Koordinaten. Beide spannen wiederum einen Raum auf, welcher als *Partikelzustandsraum* bezeichnet wird, vgl. auch Gerstlauer [18].

Wie schon angedeutet, betrachtet man nicht jedes einzelne Individuum um die Population zu erfassen, vielmehr benutzt man eine Anzahldichtefunktion F. Diese beschreibt die Anzahl der Individuen, welche sich zu einem Zeitpunkt t in einem Teilvolumen des Partikelzustandsraumes befinden. Damit hängt F im Allgemeinen von $\boldsymbol{e}, \boldsymbol{r}$ und t ab. Die Gesamtzahl N aller Individuen zum Zeitpunkt t ergibt sich somit aus

$$N(t) = \int_{\mathbb{O}} \int_{\mathbb{E}} F(\boldsymbol{e}, \boldsymbol{r}, t)\, dV_{\boldsymbol{e}}\, dV_{\boldsymbol{r}},$$

wobei durch $dV_{\boldsymbol{e}}$ bzw. $dV_{\boldsymbol{r}}$ infinitesimale Volumenelemente aus $\mathbb{E}$ bzw. $\mathbb{O}$ bezeichnet seien. Geht man von dem wichtigen Spezialfall aus, dass F ortsunabhängig ist, man spricht dann auch von einem *örtlich konzentrierten System*, so erhalten wir N in der Form

$$\begin{aligned} N(t) &= \int_{\mathbb{O}} \int_{\mathbb{E}} F(\boldsymbol{e}, t)\, dV_{\boldsymbol{e}}\, dV_{\boldsymbol{r}} = \int_{\mathbb{O}} dV_{\boldsymbol{e}} \int_{\mathbb{E}} F(\boldsymbol{e}, t)\, dV_{\boldsymbol{r}} \\ &= V_{\text{total}} \int_{\mathbb{E}} F(\boldsymbol{e}, t)\, dV_{\boldsymbol{r}}. \end{aligned}$$

Da wir über den gesamten Ortsraum integrieren, ist in V_{total} sowohl das Volumen der dispersen als auch der kontinuierlichen Phase enthalten.

Bemerkung 2.4.1: Bei einem örtlich konzentrierten System wird mitunter die Anzahldichtefunktion durch das Volumen V_{total} geteilt (normalisiert), man spricht dann auch von einer *extensiven* oder Anzahldichtefunktion, die ursprüngliche Darstellung von F bezeichnet man auch als *volumenbezogene* Anzahldichtefunktion.

Für unsere Modellbeschreibung haben wir den Zugang so allgemein wie möglich gewählt, so dass unser Modell auf die meisten Populationen anwendbar ist. Auf verschiedene mögliche Vereinfachungen werden wir erst im Abschnitt 2.6 eingehen, wo wir jeweils ein spezielles Modell für Kristallisation und Emulsion vorstellen.

Zunächst sei bemerkt, dass die Anzahldichtefunktion F eine *Anzahl* von Individuen beschreibt und somit streng genommen nur ganzzahlige Werte annehmen sollte. Da sie bei dieser Betrachtungsweise insbesondere nicht differenzierbar wäre, fassen wir F daher als mittlere Anzahl von Individuen im Partikelzustandsraum auf, welche bzgl. t stetig

differenzierbar ist. Wir benutzen außerdem die Funktion c, welche die Konzentration in der kontinuierlichen Phase beschreibt und von $\boldsymbol{r}$ und t abhängt.

Wir interessieren uns für die zeitliche Änderung von F bzw. c und betrachten jeweils deren Bilanzen $\partial F/\partial t$ bzw. $\partial c/\partial t$. Allgemein können wir $\partial F/\partial t$ schreiben als

$$\begin{aligned}\frac{\partial F(\boldsymbol{r},\boldsymbol{e},t)}{\partial t} = &- \sum_{i=1}^{\dim \mathbb{O}} \frac{\partial}{\partial r_i}(v_\mathrm{r}(\boldsymbol{r}) \cdot F(\boldsymbol{r},\boldsymbol{e},t)) - \sum_{j=1}^{\dim \mathbb{E}} \frac{\partial}{\partial e_j}(v_\mathrm{e}(\boldsymbol{e},c) \cdot F(\boldsymbol{r},\boldsymbol{e},t)) \\ &+ \bar{F}_\mathrm{in}(\boldsymbol{r},\boldsymbol{e},t) - \bar{F}_\mathrm{out}(\boldsymbol{r},\boldsymbol{e},t) \\ &+ \bar{F}_\mathrm{nuc}(\boldsymbol{r},\boldsymbol{e},c) - \bar{F}_\mathrm{diss}(\boldsymbol{r},\boldsymbol{e},c) \\ &+ \bar{F}_\mathrm{intern}(\boldsymbol{r},\boldsymbol{e},t).\end{aligned} \tag{2.1}$$

Dabei bezeichnen

$$\sum_{i=1}^{\dim \mathbb{O}} \frac{\partial}{\partial r_i}(v_\mathrm{r}(\boldsymbol{r}) \cdot F(\boldsymbol{r},\boldsymbol{e},t)) \quad \text{bzw.} \quad \sum_{j=1}^{\dim \mathbb{E}} \frac{\partial}{\partial e_j}(v_\mathrm{e}(\boldsymbol{e},c) \cdot F(\boldsymbol{r},\boldsymbol{e},t)),$$

die Flüsse in Richtung der Ortskoordinaten $\boldsymbol{r}$ bzw. die Flüsse in Richtung der Eigenschaftskoordinaten $\boldsymbol{e}$ und v_r bzw. v_e die Geschwindigkeiten dieser Flüsse. Die unterschiedlichen Vorzeichen kennzeichnen Quellen- bzw. Senkenterme. So wird durch $\bar{F}_\mathrm{in}$ bzw. $\bar{F}_\mathrm{out}$ der Zufluss bzw. Abfluss von Individuen berücksichtigt, während durch $\bar{F}_\mathrm{nuc}$ bzw. $\bar{F}_\mathrm{diss}$ das Entstehen bzw. Auflösen von Mitgliedern der Population realisiert wird. Der Term $\bar{F}_\mathrm{intern}$ steht für die innerhalb der dispersen Phase ablaufenden Phänomene.

Die zeitliche Änderung der Konzentration c ist allgemein gegeben durch

$$\begin{aligned}\frac{\partial c(\boldsymbol{r},t)}{\partial t} = &- \sum_{i=1}^{\dim \mathbb{O}} \frac{\partial}{\partial r_i}(v(\boldsymbol{r}) \cdot c(\boldsymbol{r},\boldsymbol{e},t)) \\ &+ \dot{c}_\mathrm{in}(\boldsymbol{r},t) - \dot{c}_\mathrm{out}(\boldsymbol{r},t) \\ &- \dot{c}_\mathrm{transfer}(\boldsymbol{r},t).\end{aligned} \tag{2.2}$$

Dabei bedeutet

$$\sum_{i=1}^{\dim \mathbb{O}} \frac{\partial}{\partial r_i}(v(\boldsymbol{r}) \cdot c(\boldsymbol{r},\boldsymbol{e},t))$$

die Änderung der Konzentration infolge der im Reaktor herrschenden Strömung und $\dot{c}_\mathrm{in}$ bzw. $\dot{c}_\mathrm{out}$ den Zufluss bzw. Abfluss. Die Änderung der Konzentration durch den Stoffaustausch zwischen der kontinuierlichen und der dispersen Phase durch die entsprechenden Phänomene wird durch $\dot{c}_\mathrm{transfer}$ beschrieben.

2.4.2. Partikeleigenschaften

Die durch die Koordinaten $\boldsymbol{e}$ beschriebenen Eigenschaften hängen natürlich von der Art der Population ab. Bei der Herstellung von Kristallen spielt z.B. die Größe eine wesentliche Rolle, sie ist außerdem ein entscheidendes Qualitätsmerkmal. Die Größe

eines Kristalls beschreibt man durch dessen *charakteristische Länge*. In unseren eindimensionalen Modellen betrachten wir daher die Größenverteilung der Kristalle. Durch genaue Kenntnis der Modelle hofft man diese Verteilung bei der Kristallbildung gezielt steuern zu können, z.B. möchte man von einer bestimmten Kristallgröße so viele wie möglich erhalten. Eine weitere wichtige Eigenschaft für viele Anwendungen ist die Struktur des Kristallgitters. Sie sollte möglichst gleichmäßig sein. Ist dies nicht der Fall, kann es zu Verspannungen im Gitter kommen und dadurch die Wachstumsgeschwindigkeit erheblich reduziert werden. Die Eigenschaft, durch welche diese Gitterstruktur charakterisiert wird, nennt man *Spannungsenergie*.

Die Tropfen im Emulsionsmodell werden als kugelförmig angesehen. Die wesentliche Eigenschaft bei ihrer Beschreibung ist der Durchmesser der Tropfenkugel. Äquivalent dazu rechnet man häufig auch mit dem Volumen der Kugel.

Bemerkung 2.4.2: Da die Eigenschaft *Größe* eines Individuums eine primäre ist, wollen wir noch auf Folgendes hinweisen.

a) Obwohl ein Kristall im Allgemeinen nicht kugelförmig ist, kann man doch dessen Volumen v durch seine charakteristische Länge ℓ ausdrücken. In Analogie zum Tropfen entspricht die charakteristische Länge des Kristalls dem Durchmesser eines Tropfens. Es gilt die Beziehung

$$v = k_V \cdot \ell^3. \tag{2.3}$$

Dabei ist k_V ein *Volumen-Formfaktor* der von der Art des Kristalls abhängt. Im Falle einer Kugel, wie beim Tropfen, gilt offenbar $k_V = \pi/6$.

b) Sowohl bei den Kristallen als auch bei den Tropfen sind keine unendlich großen Längen bzw. Durchmesser zugelassen, was dadurch motiviert ist, dass die Wahrscheinlichkeit, vom Rührer getroffen zu werden, mit der Größe des Individuums steigt. Es gibt also ein $\ell_{\max}$ bzw. ein $v_{\max}$ mit

$$0 \leq \ell \leq \ell_{\max} < \infty \quad \forall \ell \quad \text{bzw.} \quad 0 \leq v \leq v_{\max} < \infty \quad \forall v. \tag{2.4}$$

Wir werden in unseren Modellen 0 als minimale Länge eines Individuums annehmen, wobei zu bemerken ist, dass die Länge eines Individuums zwar nahe Null liegen kann, aber doch immer von Null verschieden ist. Als kleinste mögliche Größe könnte man etwa den Atom- oder Molekülduchmesser der beteiligten Stoffe angeben[1)].

Notation 2.4.3: Ungeachtet dessen, dass $\boldsymbol{e}$ natürlich kein Individuum ist, sondern die Eigenschaften eines solchen beschreibt, werden wir häufig $\boldsymbol{e}$ mit einem Individuum identifizieren in dem Sinne, dass wir „das Individuum mit den Eigenschaften $\boldsymbol{e}$“ abkürzen durch „(das Individuum) $\boldsymbol{e}$“.

Um Verständnisproblemen vorzubeugen, die durch die Einführung neuer Begriffe im Abschnitt 2.3 auftreten könnten, sei zur Vereinheitlichung der Sprechweisen zunächst bemerkt:

[1)] Individuen dieser Größe gehören allerdings zur kontinuierlichen Phase.

Notation 2.4.4: Wenn nicht anders bemerkt, benutzen wir in Zukunft die folgenden allgemeineren Begriffe:

a) Von Agglomeration spricht man im Allgemeinen, wenn die ursprüngliche Struktur der Individuen, die sich vereinigt haben, nach der Vereinigung noch erkennbar ist, wie im Fall der Kristalle. Im Fall der Emulsion verschmelzen kleine Tropfen zu einem großen, dabei löst sich die Form der kleinen Tropfen auf, d.h. ihre ursprüngliche Form ist nicht mehr zu erkennen, und man spricht von Koaleszenz. Bei der allgemeinen Bezeichnung von Vereinigungsprozessen benutzen wir den Begriff der *Aggregation*.

b) Zur Verallgemeinerung der speziellen Termini *Kristallisator* und *Rührkessel* verwenden wir den Begriff des *Reaktors*.

2.5. Der Integralterm $\bar{F}_{\text{intern}}$

Im vorigen Abschnitt haben wir das vollständige Modell betrachtet. Unser Ziel ist die Implementierung des Populationsmodells bzw. der Populationsbilanz, durch welche die Ereignisse in der dispersen Phase beschrieben werden. Der dafür relevante Term ist $\bar{F}_{\text{intern}}$, der die Aggregation und und die Dispersion beinhaltet, man schreibt entsprechend

$$\bar{F}_{\text{intern}}(\boldsymbol{r}, \boldsymbol{e}, t) = \bar{F}_{\text{co}}(\boldsymbol{r}, \boldsymbol{e}, t) + \bar{F}_{\text{br}}(\boldsymbol{r}, \boldsymbol{e}, t). \tag{2.5}$$

Dabei teilt man $\bar{F}_{\text{co}}$ und $\bar{F}_{\text{br}}$ in entsprechende Quell- und Senkenterme. Der Ansatz lautet

$$\bar{F}_{\text{co}}(\boldsymbol{r}, \boldsymbol{e}, t) = \bar{F}^{+}_{\text{co}}(\boldsymbol{r}, \boldsymbol{e}, t) - \bar{F}^{-}_{\text{co}}(\boldsymbol{r}, \boldsymbol{e}, t) \tag{2.5a}$$

$$\bar{F}_{\text{br}}(\boldsymbol{r}, \boldsymbol{e}, t) = \bar{F}^{+}_{\text{br}}(\boldsymbol{r}, \boldsymbol{e}, t) - \bar{F}^{-}_{\text{br}}(\boldsymbol{r}, \boldsymbol{e}, t). \tag{2.5b}$$

Hier bezeichnen $\bar{F}^{+}$ die Quellterme und $\bar{F}^{-}$ die Senkenterme des entsprechenden Phänomens.

2.5.1. Dispergierung

Das allgemeine Modell für die Dispergierung setzt sich zusammen aus

$$\bar{F}^{+}_{\text{br}}(\boldsymbol{r}, \boldsymbol{e}, t) = \int_{\Omega(\boldsymbol{e})} \nu(\boldsymbol{r}, \tilde{\boldsymbol{e}})\, \varphi(\boldsymbol{r}, \boldsymbol{e}, \tilde{\boldsymbol{e}})\, \beta_{\text{br}}(\boldsymbol{r}, \tilde{\boldsymbol{e}})\, F(\boldsymbol{r}, \tilde{\boldsymbol{e}}, t)\, d\tilde{\boldsymbol{e}} \tag{2.6a}$$

$$\bar{F}^{-}_{\text{br}}(\boldsymbol{r}, \boldsymbol{e}, t) = \beta_{\text{br}}(\boldsymbol{r}, \boldsymbol{e})\, F(\boldsymbol{r}, \boldsymbol{e}, t). \tag{2.6b}$$

Die Quelle ist dadurch motiviert, dass ein Individuum mit dem Rührer zusammenstößt und in mindestens zwei Teile zerfällt, die als neue Individuen in die Population aufgenommen werden. Weiterhin verschwindet das Individuum, welches zerfällt, aus der Population und es kommt somit zu einer Senke.

Bemerkung 2.5.1: Ein wichtige Ausnahme für einen Zerfall ohne Einfluss des Rührers ist das Bakterienmodell. Dort teilt sich ein Individuum, wenn es z.B. genügend Nährstoffe aufgenommen oder eine gewisse Größe erreicht hat. Ein Vorteil dabei ist, dass eine Bakterie stets in genau zwei Tochterbakterien zerfällt.

Die Bedeutung der einzelnen Terme ist wie folgt:

ν: diese Funktion gibt die durchschnittliche Anzahl der Teilchen an, die bei einem Zerfall entstehen. Es gilt $\nu \geq 2$. In vielen Fällen kann man $\nu \equiv 2$ setzen, siehe auch Bemerkung 2.5.1.

β_{br}: wird als *Zerfalls-* oder *Dispergierrate* bezeichnet, sie gibt die Wahrscheinlichkeit an, dass ein Partikel $\tilde{\boldsymbol{e}}$ überhaupt zerfällt. Oft modelliert man die Zerfallsrate vermittels der Wahrscheinlichkeiten θ_{br} und ϑ_{br} gemäß

$$\beta_{\mathrm{br}}(\boldsymbol{r}, \tilde{\boldsymbol{e}}) = \theta_{\mathrm{br}}(\boldsymbol{r}, \tilde{\boldsymbol{e}}) \cdot \vartheta_{\mathrm{br}}(\boldsymbol{r}, \tilde{\boldsymbol{e}}). \tag{2.7}$$

Dabei ist θ_{br} die Wahrscheinlichkeit, dass ein Teilchen sich in einem für den Zerfall günstigen Bereich befindet, d.h. die Wahrscheinlichkeit, dass es mit dem Rührer kollidiert und die Funktion ϑ_{br} die Wahrscheinlichkeit, dass, wenn es denn zur Kollision kommt, auch wirklich ein Zerfall stattfindet. Die Funktionen θ_{br} bzw. ϑ_{br} bezeichnet man dementsprechend als *Kollisionsfrequenz* bzw. *Kollisionseffizienz* oder auch als *Dispersionsfrequenz* bzw. *Dispersionseffizienz.*

φ: ist eine Wahrscheinlichkeitsdichte, die das Ereignis wiedergibt, dass das Teilchen $\tilde{\boldsymbol{e}}$ in das Teilchen $\boldsymbol{e}$ zerfällt bzw. wie Bruchstücke $\boldsymbol{e}$ bzgl. des ursprünglichen Teilchens $\tilde{\boldsymbol{e}}$ verteilt sind. Insbesondere gilt

$$\int_{\Omega(\boldsymbol{e})} \varphi(\boldsymbol{r}, \boldsymbol{e}, \tilde{\boldsymbol{e}})\, d\tilde{\boldsymbol{e}} = 1. \tag{2.8}$$

Bemerkung 2.5.2: a) Bei der Dispergierung kann die Größe des entstehenden Teilchens nicht größer als die des ursprünglichen sein. Speziell für die Eigenschaften Länge, Größe oder Volumen ist das Integral (2.6a) identisch Null, falls $e > \tilde{e}$ gilt. Wir haben in diesem Falle eine Integralgleichung vom Volterratyp vorliegen, d.h. der Integrationsbereich ist von e abhängig. Im eindimensionalen Fall gilt somit für den Term $\bar{F}^{+}_{\mathrm{br}}$

$$\bar{F}^{+}_{\mathrm{br}}(\boldsymbol{r}, e, t) = \int_{e}^{e_{\max}} \nu(\boldsymbol{r}, \tilde{e})\, \varphi(\boldsymbol{r}, e, \tilde{e})\, \beta_{\mathrm{br}}(\boldsymbol{r}, \tilde{e})\, F(\boldsymbol{r}, \tilde{e}, t)\, d\tilde{e}, \tag{2.9a}$$

wobei $e_{\max}$ die maximale Partikelgröße bezeichnet, vgl. Bemerkung 2.4.2 b).

b) Auch zeitabhängige Modellierungen von β_{br} erscheinen durchaus sinnvoll. In der Literatur war allerdings kein solches Modell zu finden, daher werden wir hier nicht weiter darauf eingehen. Speziell bei der Diskretisierung würde eine Zeitabhängigkeit von β_{br} auch Probleme bereiten.

2.5.2. Aggregation

Bei der Dispergierung kollidierte ein Teilchen mit dem Rührer, bei der Aggregation hingegen zwei Teilchen miteinander. Bleiben diese genügend lange in Kontakt, so können sie sich zu einem neuen verbinden, wodurch der Quellenterm gerechtfertigt wird. Man geht davon aus, dass sich stets genau zwei Teilchen verbinden. Als (empirische) Begründung führen wir an, dass das Ereignis, dass mehr als zwei Teilchen zum gleichen Zeitpunkt aufeinandertreffen sehr unwahrscheinlich ist und somit vernachlässigt werden kann. Schon bei zwei Teilchen ist die Berechnung der Quelle aufwendig, da die Funktion F quadratisch im Integral auftritt[1)]. Der Senkenterm ist motiviert dadurch, dass die beiden Teilchen, die sich vereinigen, aus der Population verschwinden. Die Terme aus (2.5a) setzen sich somit zusammen aus

$$\bar{F}^{+}_{\text{co}}(\boldsymbol{r},\boldsymbol{e},t) = \frac{1}{2}\int_{\Omega(\boldsymbol{e})} \beta_{\text{co}}(\boldsymbol{r},\boldsymbol{e}-\tilde{\boldsymbol{e}},\tilde{\boldsymbol{e}})\, F(\boldsymbol{r},\boldsymbol{e}-\tilde{\boldsymbol{e}},t)\, F(\boldsymbol{r},\tilde{\boldsymbol{e}},t)\, d\tilde{\boldsymbol{e}} \tag{2.10a}$$

und

$$\bar{F}^{-}_{\text{co}}(\boldsymbol{r},\boldsymbol{e},t) = F(\boldsymbol{r},\boldsymbol{e},t)\int_{\Omega(\boldsymbol{e})} \beta_{\text{co}}(\boldsymbol{r},\boldsymbol{e},\tilde{\boldsymbol{e}})\, F(\boldsymbol{r},\tilde{\boldsymbol{e}},t)\, d\tilde{\boldsymbol{e}}. \tag{2.10b}$$

Hier gibt es nur einen neuen Term:

β_{co}: Wahrscheinlichkeit, dass sich Partikel $\boldsymbol{e}$ mit dem Partikel $\tilde{\boldsymbol{e}}$ verbindet. Diese bezeichnet man als *Aggregationsrate*[2)]. Ebenso wie beim Zerfall kann man die Aggregationsrate mit Hilfe der Wahrscheinlichkeiten θ_{co} und ϑ_{co} modellieren

$$\beta_{\text{co}}(\boldsymbol{r},\boldsymbol{e},\tilde{\boldsymbol{e}}) = \theta_{\text{co}}(\boldsymbol{r},\boldsymbol{e},\tilde{\boldsymbol{e}})\cdot\vartheta_{\text{co}}(\boldsymbol{r},\boldsymbol{e},\tilde{\boldsymbol{e}}). \tag{2.11}$$

Hier ist θ_{co} die Wahrscheinlichkeit, dass zwei Partikel miteinander kollidieren. Ähnlich wie bei der Dispergierung wird die Möglichkeit, dass nicht jede Kollision auch zur Aggregation führt, durch ϑ_{co} berücksichtigt. Die Funktionen θ_{co} bzw. ϑ_{co} bezeichnet man analog als *Aggregationsfrequenz*[3)] bzw. *Aggregationseffizienz*[4)] oder, ebenso wie im Falle der Dispersion, allgemein als *Kollisionsfrequenz* bzw. *Kollisionseffizienz*.

Die Struktur von (2.10a) scheint auf den ersten Blick ungewöhnlich. Zum besseren Verständnis wählen wir zunächst eine alternative Formulierung. Seien $\tilde{\tilde{\boldsymbol{e}}}$ und $\tilde{\boldsymbol{e}}$ zwei Partikel, die sich zu einem neuen Partikel $\boldsymbol{e}$ vereinigen, d.h.

$$\tilde{\tilde{\boldsymbol{e}}} + \tilde{\boldsymbol{e}} = \boldsymbol{e}, \tag{2.12}$$

wobei man sich als Eigenschaft das Volumen der Teilchen vorstellt, dann lässt sich (2.10a) auch auf diese Weise schreiben:

$$\bar{F}^{+}_{\text{co}}(\boldsymbol{r},\tilde{\tilde{\boldsymbol{e}}}+\tilde{\boldsymbol{e}},t) = \frac{1}{2}\int_{\Omega(\boldsymbol{e})} \beta_{\text{co}}(\boldsymbol{r},\tilde{\tilde{\boldsymbol{e}}},\tilde{\boldsymbol{e}})\, F(\boldsymbol{r},\tilde{\tilde{\boldsymbol{e}}},t)\, F(\boldsymbol{r},\tilde{\boldsymbol{e}},t)\, d\tilde{\boldsymbol{e}}, \tag{2.13}$$

[1)] wären drei Teilchen beteiligt, würde F kubisch auftreten, etc.

[2)] in den speziellen Fällen der Kristallisation bzw. der Emulsion spricht man auch von *Agglomerations-* bzw. *Koaleszenzrate*

[3)] man spricht auch von *Agglomerations-* bzw. *Koaleszenzfrequenz*

[4)] man spricht auch von *Agglomerations-* bzw. *Koaleszenzeffizienz*

was für eine spätere Lösung ungeeignet scheint. In (2.10a) betrachten wir daher das Ereignis, dass sich die Partikel $\boldsymbol{e} - \tilde{\boldsymbol{e}}$ und $\tilde{\boldsymbol{e}}$ zu dem Partikel $\boldsymbol{e}$ vereinigen. Setzen wir $\tilde{\tilde{\boldsymbol{e}}} = \boldsymbol{e} - \tilde{\boldsymbol{e}}$ in (2.13) ein, erhalten wir die ursprüngliche Gleichung (2.10a).

Bemerkung 2.5.3: a) Offenbar ist das Ereignis, dass sich $\boldsymbol{e}$ mit $\tilde{\boldsymbol{e}}$ verbindet gleichwahrscheinlich mit dem Ereignis, dass sich $\tilde{\boldsymbol{e}}$ mit $\boldsymbol{e}$ verbindet, d.h. β_{co} ist symmetrisch bzw.

$$\beta_{\text{co}}(\boldsymbol{r}, \boldsymbol{e}, \tilde{\boldsymbol{e}}) = \beta_{\text{co}}(\boldsymbol{r}, \tilde{\boldsymbol{e}}, \boldsymbol{e}). \tag{2.14}$$

b) Die beiden Teilchen, die sich verbinden, können bzgl. der Eigenschaften Größe, Länge oder Volumen niemals größer sein, als das entstehende. Es gilt also im eindimensionalen Fall $\tilde{\tilde{e}}, \tilde{e} \leq e$ bzw. $e - \tilde{e}, \tilde{e} \leq e$ und somit folgt für den Quellterm (2.10a)

$$\bar{F}^{+}_{\text{co}}(\boldsymbol{r}, e, t) = \frac{1}{2} \int_0^e \beta_{\text{co}}(\boldsymbol{r}, e - \tilde{e}, \tilde{e})\, F(\boldsymbol{r}, e - \tilde{e}, t)\, F(\boldsymbol{r}, \tilde{e}, t)\, d\tilde{e}. \tag{2.15a}$$

Andererseits gilt für den Senkenterm (2.10b), dass das neu entstandene Teilchen kleiner sein muss als $e_{\max}$, vgl. Bemerkung 2.4.2 b), d.h. $e + \tilde{e} \leq e_{\max}$ bzw. $\tilde{e} \leq e_{\max} - e$ und somit folgt

$$\bar{F}^{-}_{\text{co}}(\boldsymbol{r}, e, t) = F(\boldsymbol{r}, e, t) \int_0^{e_{\max} - e} \beta_{\text{co}}(\boldsymbol{r}, e, \tilde{e})\, F(\boldsymbol{r}, \tilde{e}, t)\, d\tilde{e}. \tag{2.15b}$$

c) Bei der Integralbildung über die Quelle zählt man die entstehenden Individuen doppelt, deshalb erscheint der Faktor 1/2 in (2.10a) und (2.15a). Alternativ dazu könnte man den Integrationsbereich entsprechend verkleinern.

2.6. Die Modellgleichungen im Einzelnen

2.6.1. Allgemeines

Die Gleichung (2.1) ist unser Ausgangspunkt, um Vorgänge wie Wachstum, Dispergierung und Aggregation von Teilchen zu untersuchen. Die Wahl der internen Koordinaten ist dabei bestimmt durch die beabsichtigte Nutzung des Modells.

Wir beschränken uns hier auf die Kristall-Größenverteilung, man spricht auch von *crystal size distribution* oder kurz *CSD*. In diesem Fall ist der Eigenschaftsvektor eindimensional, die zugehörige Eigenschaft ist die charakteristischen Kristalllänge ℓ.

Die Tropfen in der Emulsion modellieren wir ebenfalls eindimensional, die Eigenschaftskoordinate ist dabei das Tropfenvolumen v.

Die Modellbeschreibungen sind oft sehr komplex, so auch in unserem Fall, daher werden wir uns auf das Wesentliche beschränken und nicht auf alle Details eingehen.

2.6.2. Ein Kristallmodell

Das Modell für die Dispergierung wurde aus Gerstlauer et.al. [19] sowie Motz [36] und [37] entnommen und entspricht den Erfahrungen bei industrieller Fertigung von Kristallen. Die Eigenschaftskoordinate ist die charakteristische Länge ℓ eines Kristalls, siehe auch Gleichung (2.3). Die Ortsabhängigkeit beschränkt sich auf den Bereich des Rührers. Hier machen wir von der Rotationssymmetrie des Kristallisators Gebrauch. Als weitere Vereinfachung geht die Höhe des Rührers nicht explizit in die Abhängigkeit ein. Somit reduzieren wir die Ortskoordinaten auf den Radius r des Rührers.

Dispergierung

Man sollte annehmen, dass beim Zerfall eines Kristalls zwei annähernd gleichgroße Stücke entstehen und dass für unsere Dichtefunktion φ aus (2.6a) der Erwartungswert bei der Hälfte der Länge des Ausgangskristalls liegt. Bemerkenswerterweise zerfällt ein Kristall tatsächlich in ein großes Fragment, welches nur etwas kleiner ist als der Ursprungskristall und in viele kleine Kristalle. Man spricht hier anstelle von Zerfall auch treffender von *Abrieb*. Dementsprechend haben wir in diesem Fall auch zwei Quellterme. Der erste beschreibt das große Kristallfragment, den wir mit Hilfe der δ-Distribution ebenfalls als Integral

$$\bar{F}^{+}_{\mathrm{br},1}(\ell, r, t) = \int_{\ell}^{\ell_{\max}} \delta\left(\ell - \left[\tilde{\ell}^3 - \frac{V_{\mathrm{frag}}(\tilde{\ell}, r)}{k_V}\right]^{\frac{1}{3}}\right) \beta_{\mathrm{br}}(\tilde{\ell})\, F(\tilde{\ell}, t)\, d\tilde{\ell} \tag{2.16a}$$

schreiben können, wobei die Funktionen ν und φ aus (2.6a) identisch 1 sind. Für die kleinen Kristallfragmente, die vom großen Kristall abgerieben werden, haben wir

$$\begin{aligned}\bar{F}^{+}_{\mathrm{br},2}(\ell, r, t) = \int_{\ell}^{\ell_{\max}} \big[\, H(\ell - \ell_{\mathrm{frag,min}}) - H(\ell - \ell_{\mathrm{frag,max}}(\tilde{\ell}, r)) \,\big] \\ \cdot \nu(\tilde{\ell})\, \varphi(\ell, \tilde{\ell}, r)\, \beta_{\mathrm{br}}(\tilde{\ell})\, F(\tilde{\ell}, t)\, d\tilde{\ell}.\end{aligned} \tag{2.16b}$$

Dabei wird durch die *Heaviside-Funktionen* H sichergestellt, dass die Dichtefunktion φ nur Werte zwischen der minimal und maximal möglichen Fragmentgröße $\ell_{\mathrm{frag,min}}$ und $\ell_{\mathrm{frag,max}}$ annimmt. Für den Senkenterm ändert sich nichts:

$$\bar{F}^{-}_{\mathrm{br}}(\ell, t) = \beta_{\mathrm{br}}(\ell)\, F(\ell, t). \tag{2.16c}$$

Wir führen zuerst eine nur vom Material abhängige *Bruchkonstante* γ_{br} ein gemäß

$$\gamma_{\mathrm{br}} = \frac{\mu_{\mathrm{shear}}\, \Gamma}{K_r}. \tag{2.17}$$

Es bezeichnen μ_{shear} das auftretende *Schermodul*, Γ die *Bruchbeständigkeit* und K_r die *Bruchrate*. Das Verhältnis Γ/K_r nennt man auch *Risszähigkeit*, vergleiche für alle Konstanten auch Tabelle 2.1. Die Funktion V_{frag} aus (2.16a) repräsentiert gerade das Volumen, welches vom großen Kristallfragment abgerieben wird, gemäß

$$V_{\mathrm{frag}}(\tilde{\ell}, r) = \frac{2}{3} \frac{H_V^{\frac{2}{3}}}{\gamma_{\mathrm{br}}} E_{\mathrm{kin}}^{\frac{4}{3}}(\tilde{\ell}, r), \tag{2.18}$$

mit H_V als *Vickers-Härte*, Tabelle 2.1. Die kinetische Energie E_{kin} ergibt sich aus der bekannten Formel

$$E_{\text{kin}}(\tilde{\ell}, r) = \frac{m(\tilde{\ell})}{2} v_{\text{coll}}^2(\tilde{\ell}, r), \tag{2.19}$$

wobei v_{coll} die Kollisionsgeschwindigkeit von Kristall und Rührer ist und für die Masse m des Kristalls gemäß Gleichung (2.3) gilt

$$m(\tilde{\ell}) = v(\tilde{\ell})\varrho_{\text{d}} = k_V \tilde{\ell}^3 \varrho_{\text{d}}. \tag{2.20}$$

Auf v_{coll} werden wir später in Gleichung (2.33) eingehen. Zunächst betrachten wir den Definitionsbereich der Fragmentgrößen, die minimal mögliche ist eine reine Materialgröße und definiert durch

$$\ell_{\text{frag,min}} = \frac{32}{3} \frac{\gamma_{\text{br}}}{H_V^2}. \tag{2.21}$$

Die maximale Fragmentgröße ist außerdem von E_{kin} abhängig und somit von ℓ und r. Sie ist gegeben durch

$$\ell_{\text{frag,max}}(\tilde{\ell}, r) = \frac{1}{2} \frac{H_V^{\frac{2}{9}}}{\gamma_{\text{br}}^{\frac{1}{3}}} E_{\text{kin}}^{\frac{4}{9}}(\tilde{\ell}, r). \tag{2.22}$$

Damit haben wir alles Nötige, um die Funktion ν aus Gleichung (2.6a), die uns die Anzahl der entstehenden Kristallfragmente liefert, aufzuschreiben

$$\nu(\tilde{\ell}, r) = \frac{V_{\text{frag}}(\tilde{\ell}, r)}{3k_V} \frac{\ell_{\text{frag,min}}^{-2.25} - \ell_{\text{frag,max}}^{-2.25}(\tilde{\ell}, r)}{\ell_{\text{frag,max}}^{0.75}(\tilde{\ell}, r) - \ell_{\text{frag,min}}^{0.75}}. \tag{2.23}$$

Für die Wahrscheinlichkeitsdichte φ aus (2.6a) haben wir

$$\varphi(\ell, \tilde{\ell}, r) = \frac{2.25}{\ell_{\text{frag,min}}^{-2.25} - \ell_{\text{frag,max}}^{-2.25}(\tilde{\ell}, r)} \cdot \ell^{-3.25}. \tag{2.24}$$

Um die Zerfallsrate β_{br} aus (2.6a) darzustellen, betrachten wir die Möglichkeiten, wie ein Kristall mit dem Rührer, dessen Blätter um den Winkel β geneigt sind, vgl. Abbildungen 2.8 und 2.9, zusammenstoßen kann. Allgemein unterscheidet man zwei Arten von Zusammenstößen, mit der Rührerblattfläche und mit der Rührerblattkante, Abb. 2.8. Dabei nehmen wir zur Vereinfachung an, dass Blattfläche und Blattkante keine Krümmung besitzen und senkrecht aufeinander stehen. Der größte Anteil beim Kristallabrieb wird durch Stöße mit der Kante verursacht. In unserem Modell werden wir die Zusammenstöße mit der Rührerblattfläche vernachlässigen.

Bei der Modellierung beschränkt man sich weiterhin darauf, bei den Stößen nur diejenigen Partikelgeschwindigkeiten zu betrachten, deren Richtung senkrecht zur Auftrefffläche steht. Somit steht dem Kristall nicht die ganze Fläche $H \cdot dr$ zur Verfügung,

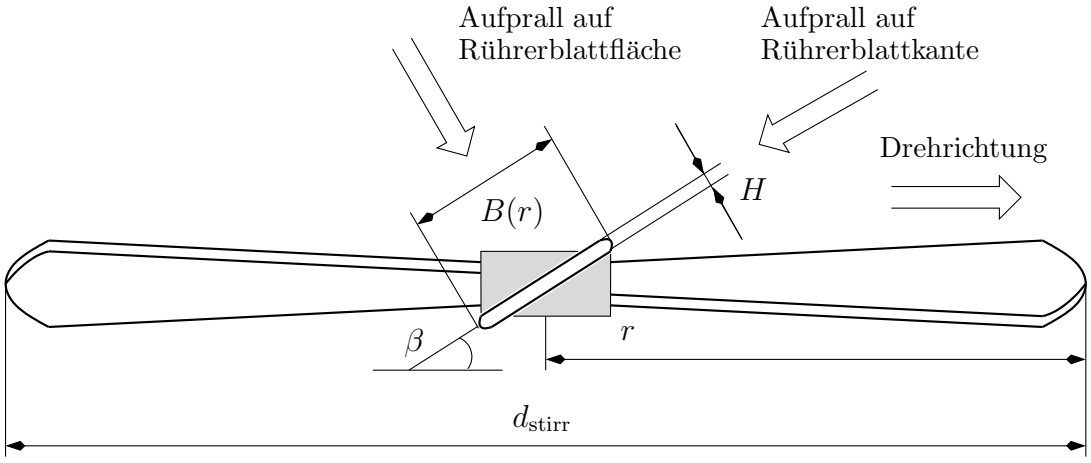


Abbildung 2.8.: Größen am Rührer.

sondern nur die *projizierte* Fläche $L_H(r) \cdot dr$, die damit genau senkrecht zum Geschwindigkeitsvektor $v(r)$ steht. In Abbildung 2.9 wird dies veranschaulicht. Für die projizierte Länge gilt somit

$$L_H(r) = H \cos \tilde{\alpha}(r). \tag{2.25}$$

Ebenfalls mit Hilfe von Abb. 2.9 findet man, dass gerade gilt

$$\begin{aligned} \tilde{\alpha}(r) &= \pi - \frac{\pi}{2} - \tilde{\beta} - \alpha(r) = \pi - \frac{\pi}{2} - \frac{\pi}{2} + \beta - \alpha(r) \\ &= \beta - \alpha(r). \end{aligned} \tag{2.26}$$

Aus dem Kräftediagramm in Abb. 2.9 erhält man α gemäß

$$\alpha(r) = \arctan \frac{v_{\text{axial}}}{v_{\text{stirr}}(r)} = \arctan \frac{v_{\text{axial}}}{2\pi\omega r}, \tag{2.27}$$

wobei mit v_{axial} die axiale Geschwindigkeit der Flüssigkeit im Leitrohr, v_{stirr} die Geschwindigkeit des Rührers und ω die Rührerdrehzahl bezeichnet wird. Zur genaueren Beschreibung von v_{axial} verweisen wir auf (2.31).

Der nächste Schritt bei der Darstellung von β_{br} ist die Einführung von zwei Wahrscheinlichkeiten, die im Prinzip θ_{br} und ϑ_{br} aus (2.7) entsprechen. In der uns zur Verfügung stehenden Literatur werden diese mit η_{geo} und η_{tar} bezeichnet. Die Funktion η_{geo} ist die (geometrische) Wahrscheinlichkeit, dass sich ein Kristall in einem günstigen Bereich über dem Rührer befindet, d.h., dass es sich auf einem Stromfaden befindet, auf dem es zum Zusammenstoß mit dem Rührer kommt. Somit entspricht η_{geo} der Dispersionsfrequenz. Die Funktion η_{tar} entspricht der Dispersionseffizienz und modelliert dementsprechend die Wahrscheinlichkeit, dass wenn sich der Kristall auf einem günstigen Stromfaden befindet, er auch träge genug ist, um mit dem Rührer (target) zu kollidieren. Die Wahrscheinlichkeiten entnehmen wir aus Motz [36]. Somit gilt

$$\eta_{\text{geo}}(r) = \frac{4\, a_{\text{stirr}}}{d_{\text{leit}}^2} \cdot \frac{L_H(r)}{\sin \alpha(r)}, \tag{2.28a}$$

dabei bezeichnet a_{stirr} die Anzahl der Rührerblätter und d_{leit} den Durchmesser des Leitrohres, für L_H bzw. α vgl. (2.25) bzw. (2.27). Ferner gilt

$$\eta_{\text{tar}}(\tilde{\ell}, r) = a_{\text{stirr}} \left(\frac{\Psi(\tilde{\ell}, r)}{0.32 + \Psi(\tilde{\ell}, r)} \right)^{2.1}, \tag{2.28b}$$

dabei ist Ψ die materialabhängige Stokes-Zahl, siehe Gahn und Mersmann [17], gegeben durch

$$\Psi(\tilde{\ell}, r) = \frac{\varrho_{\text{d}} - \varrho_{\text{c}}}{18\eta_{\text{c}}} \cdot \tilde{\ell}^2 \frac{v(r)}{L_H(r)}. \tag{2.29}$$

Die Konstanten ϱ_{d} bzw. ϱ_{c} bezeichnen die Dichten der dispersen bzw. der kontinuierlichen Phase sowie η_{c} die *dynamische Viskosität* der kontinuierlichen Phase, siehe Tabelle 2.1 und $v(r)$ die individuelle Geschwindigkeit eines Kristalls, siehe Abb. 2.9.

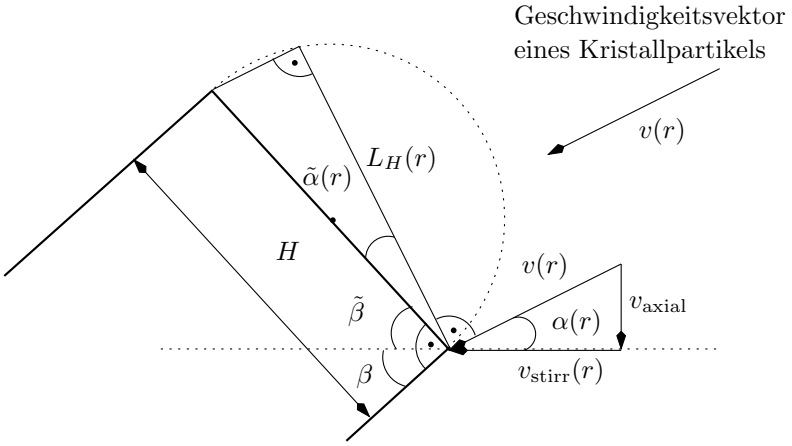


Abbildung 2.9.: Stoß an reduzierter Rührerkante $L_H(r)$.

Wir haben nun fast alles beisammen, um die Abriebsrate β_{br} gemäß Gerstlauer [19] anzugeben

$$\beta_{\text{br}}(\tilde{\ell}) = \frac{\dot{V}_{\text{pump}}}{V_{\text{total}}} \int_0^{\frac{1}{2}d_{\text{stirr}}} \eta_{\text{geo}}(r)\eta_{\text{tar}}(\tilde{\ell}, r)dr. \tag{2.30}$$

Die fehlenden Größen sind der Durchfluss durch den Rührer $\dot{V}_{\text{pump}}$, das totale Volumen[1)] V_{total} und d_{stirr} der Durchmesser des Rührers. Das Verhältnis von $\dot{V}_{\text{pump}}$ zu V_{total} nennt man auch *Umwälzzeit*. Nachzutragen sind noch die Geschwindigkeiten v_{coll}, v_{axial} und die individuelle Kristallgeschwindigkeit v. Zunächst geben wir die Geschwindigkeit

$$v_{\text{axial}} = \frac{4}{\pi} \frac{\dot{V}_{\text{pump}}}{d_{\text{leit}}^2} \tag{2.31}$$

[1)] das Volumen von kontinuierlicher und disperser Phase zusammengenommen

an. Mit Hilfe von v_{axial} können wir nach dem Vektordiagramm aus Abbildung 2.9 die Geschwindigkeit v beschreiben gemäß

$$v(r) = \sqrt{v_{\text{axial}}^2 + v_{\text{stirr}}^2(r)}. \tag{2.32}$$

Mit η_{tar} und $\tilde{\alpha}$ aus (2.26) haben wir dann ebenfalls die Kollisionsgeschwindigkeit

$$v_{\text{coll}}(\tilde{\ell}, r) = v(r)\eta_{\text{tar}}(\tilde{\ell}, r)\cos\tilde{\alpha}(r). \tag{2.33}$$

Konstante	Wert	Einheit	Name
Γ/K_{r}	$2,8$	$[J/m^2]$	Risszähigkeit
H_V	$2,65 \cdot 10^8$	$[N/m^2]$	Vickershärte
k_{V}	$\pi/6$	$[-]$	Formfaktor
μ_{shear}	$7,17 \cdot 10^9$	$[Pa]$	Schermodul
ϱ_{d}	2109	$[kg/m^3]$	Dichte disperse Phase (KNO_3)
ϱ_{c}	987	$[kg/m^3]$	Dichte kontinuierliche Phase (H_2O)
ω	10	$[1/s]$	Rührerdrehzahl
d_{stirr}	0.1	$[m]$	Rührerdurchmesser

Tabelle 2.1.: Materialkonstanten für Kaliumnitrat (KNO_3) in Wasser (H_2O).

Aggregation

Bei der Suche nach einem zur Dispergierung aus dem vorigen Abschnitt passenden Kern für die Aggregation sind wir auf Motz [37] bzw. Smoluchowski [44] gestoßen. Es handelt sich um den so genannten *Smoluchowski-Kern*, der bereits im Jahre 1917 veröffentlicht wurde (Smoluchowski [44]). Hier wird berücksichtigt, dass die Vereinigung besonders zwischen unterschiedlich großen Kristallen stattfindet. Man spricht deshalb auch von einem *größenabhängigen Kern*. Dieser Kern besteht im Prinzip nur aus der Aggregationsrate $\beta_{\text{co}} = \theta_{\text{co}}\vartheta_{\text{co}}$. In jedem Fall wird eine Aggregationseffizienz ϑ_{co} benutzt, aber nicht bei allen Kernen wird die Aggregationsfrequenz θ_{co} explizit ausformuliert, bei diesen setzt man $\theta_{\text{co}} \equiv 1$. Eine weitere Vereinfachung in der Beschreibung ist, dass die meisten Aggregationskerne als ortsunabhängig angenommen werden. Im eindimensionalen Modell haben Quell- und Senkenterm somit die Gestalt

$$\bar{F}_{\text{co}}^{+}(\ell, t) = \frac{1}{2}\int_0^{\ell} \beta_{\text{co}}(\ell - \tilde{\ell}, \tilde{\ell})\, F(\ell - \tilde{\ell}, t)\, F(\tilde{\ell}, t)\, d\tilde{\ell} \tag{2.34a}$$

$$\bar{F}_{\text{co}}^{-}(\ell, t) = F(\ell, t)\int_0^{\ell_{\max}-\ell} \beta_{\text{co}}(\ell, \tilde{\ell})\, F(\tilde{\ell}, t)\, d\tilde{\ell}, \tag{2.34b}$$

wobei der Kern

$$\beta_{\mathrm{co}}(\ell, \tilde{\ell}) = \frac{(\ell + \tilde{\ell})^2}{c_{\mathrm{smol}} + \ell\tilde{\ell}} \tag{2.35}$$

mit $c_{\mathrm{smol}} > 0$, gerade der erwähnte Smoluchowski-Kern ist. Genauer bezeichnet man (2.35) als *modifizierten Smoluchowski-Kern*, für $c_{\mathrm{smol}} \equiv 0$ hat man den ursprünglichen Smoluchowski-Kern vorliegen, vgl. neben Smoluchowski [44], besonders auch Levich [34] und Ginter [20].

Obwohl schon lange bekannt, ist dies ein sehr wichtiger Kern, da er Aggregationsvorgänge sehr gut beschreibt. Somit ist er auch heute noch, sowohl in der ursprünglichen als auch in modifizierter Form, Bestandteil vieler Modelle, so dass man ihn durchaus als Referenzkern bezeichnen kann.

2.6.3. Ein Emulsionsmodell

Dieses Modell stammt im Wesentlichen aus Coulaloglou und Tavlarides [9], vgl. dazu aber auch Gerstlauer [18]. Die Eigenschaftskoordinate ist wieder eindimensional und beschreibt hier das Volumen v eines Tropfens. Außerdem beziehen wir die Anzahldichtefunktion F auf das Gesamtvolumen $V_{\mathrm{total}}(t)$, in dem die Volumina der dispersen und der kontinuierlichen Phase zusammengenommen werden. Wir sprechen in dem Fall auch von der *volumenbezogenen* Anzahldichtefunktion.

Dispergierung

Wir beginnen bei der Beschreibung wieder mit dem Zerfall, die allgemeinen Gleichungen dazu lauten:

$$\bar{F}^{+}_{\mathrm{br}}(v, t) = \int_{v}^{v_{\max}} \nu(\tilde{v})\, \varphi(v, \tilde{v})\, \beta_{\mathrm{br}}(\tilde{v})\, F(\tilde{v}, t)\, d\tilde{v}, \tag{2.36a}$$

$$\bar{F}^{-}_{\mathrm{br}}(v, t) = \beta_{\mathrm{br}}(v)\, F(v, t). \tag{2.36b}$$

Als Besonderheit ist zunächst zu erwähnen, dass man in diesem Modell annimmt, dass der Zerfall von Tropfen nicht durch den Rührer direkt ausgelöst wird, sondern durch turbulente Strömungen (welche natürlich durch den Rührer erzeugt werden). Dabei nehmen wir an, dass stets genau zwei Tropfen als „Bruchstücke“ entstehen, d.h.

$$\nu(\tilde{v}) \equiv 2. \tag{2.37}$$

Ein Zerfall kann dabei nur stattfinden, wenn ein Tropfen mit einem Strömungsfaden einer der Strömungen kollidiert und die Oberflächenenergie dieses Tropfens kleiner ist als die kinetische Energie des Fadens. Man nimmt dabei an, dass bei einer solchen Kollision der Tropfen auf jeden Fall zerteilt wird. Deshalb spielt für die Dispergierrate β_{br} nur die Kollisionsfrequenz θ_{br} eine Rolle. Die Kollisionseffizienz ist somit das sichere Ereignis und es gilt somit $\vartheta_{\mathrm{br}} \equiv 1$.

Für θ_{br} und somit für β_{br} können wir schreiben

$$\beta_{\mathrm{br}}(v) = \frac{1}{t_{\mathrm{br}}(v)} \frac{\Delta N(v)}{N(v)}, \tag{2.38}$$

wobei t_{br} die sogenannte *Bruchzeit* bezeichnet, ΔN die Anzahl der zerfallenden Tropfen und N die Gesamtzahl der Tropfen. Gemäß Coulaloglou und Tavlarides [9] kann man Verhältnis $\Delta N/N$ in der Form

$$\frac{\Delta N(v)}{N(v)} = \exp\left[-\frac{E_{\mathrm{srf}}(v)}{\bar{E}(v)}\right] \tag{2.39}$$

ausdrücken. Dabei bezeichnet E_{srf} die *Oberflächenenergie* eines Tropfens und $\bar{E}$ die *mittlere turbulente kinetische Energie*. Die Oberflächenenergie ist zunächst über den Durchmesser d eines Tropfens gegeben durch

$$E_{\mathrm{srf}}(d) = C_1 \sigma d^2, \tag{2.40}$$

wobei σ die Oberflächenspannung bezeichnet, die eine reine Materialkonstante ist und sowohl von der dispersen als auch von der kontinuierlichen Phase abhängt. Da die Tropfen als kugelförmig angenommen werden, können wir d natürlich als Funktion von v schreiben

$$d(v) = v^{\frac{1}{3}} \frac{6}{\pi}, \tag{2.41}$$

vgl. auch Bemerkung 2.4.2 a). Für $\bar{E}$ gilt gemäß Coulaloglou und Tavlarides [9]

$$\bar{E}(d) = C_2 \, \varrho_{\mathrm{d}} \, \epsilon^{\frac{2}{3}} \, d^{\frac{2}{3}}, \tag{2.42}$$

wobei ϱ_{d} die Dichte der dispersen Phase bezeichnet und ϵ die *örtliche Dissipationsrate*.

Für die Bruchzeit t_{br} gilt entsprechend Coulaloglou und Tavlarides [9]

$$t_{\mathrm{br}}(d) = C_3 \, d^{\frac{2}{3}} \, \epsilon^{-\frac{1}{3}} \tag{2.43}$$

und für ϵ

$$\epsilon = C_4 \, d_{\mathrm{stirr}}^2 \, \omega^3, \tag{2.44}$$

wenn man eine gleichmäßige Verteilung der Energie im Reaktor annimmt. Dabei bezeichnen wieder d_{stirr} den Rührerdurchmesser und ω die Rührerdrehzahl.

Somit können wir die Dispergierrate β_{br} darstellen gemäß

$$\begin{aligned} \beta_{\mathrm{br}}(v) &= c_{\mathrm{br},1} \, d_{\mathrm{stirr}}^{\frac{2}{3}} \, \omega \, v^{-\frac{2}{9}} \, \exp\left[-c_{\mathrm{br},2} \frac{\sigma \, v^{-\frac{5}{9}}}{\rho_{\mathrm{d}} \, d_{\mathrm{stirr}}^{\frac{4}{3}} \, \omega^2}\right] \\ &= c_{\mathrm{br},3} \, v^{-\frac{2}{9}} \, \exp\left[-c_{\mathrm{br},4} \, v^{-\frac{5}{9}}\right], \end{aligned} \tag{2.45}$$

wobei $c_{\mathrm{br},1} > 0$ bzw. $c_{\mathrm{br},2} > 0$ so genannte Bruchkonstanten sind und wir die Konstanten $c_{\mathrm{br},3}$ bzw. $c_{\mathrm{br},4}$ zur besseren Übersicht gemäß

$$c_{\mathrm{br},3} = c_{\mathrm{br},1} \cdot d_{\mathrm{stirr}}^{\frac{2}{3}} \omega$$

bzw.

$$c_{\mathrm{br},4} = c_{\mathrm{br},2} \frac{\sigma}{\varrho_{\mathrm{d}}\, d_{\mathrm{stirr}}^{\frac{4}{3}}\, \omega^2}$$

definieren, vgl. z.B. Tabelle 2.2.

Wir nehmen weiterhin an, dass die Bruchstücke sich gemäß der Gaußschen Normalverteilung verhalten und schreiben die Wahrscheinlichkeitsdichte φ gemäß

$$\varphi(v, \tilde{v}) = \frac{c_{\varphi_1}}{\tilde{v}} \exp\left[-c_{\varphi_2} \frac{(2v - \tilde{v})^2}{\tilde{v}^2}\right], \tag{2.46}$$

wobei die Konstanten c_{φ_1} und c_{φ_2} gerade so gewählt wurden, dass mindestens 99.6% der Bruchstücke der Tropfen im Intervall $[0, \tilde{v}]$ liegen, vgl. Tabelle 2.2 bzw. Coulaloglou und Tavlarides [9].

Aggregation

Die allgemeine (volumenbezogene) Darstellung der Aggregationsterme lautet

$$\bar{F}_{\mathrm{co}}^{+}(v, t) = \frac{1}{2V_{\mathrm{total}}(t)} \int_0^v \beta_{\mathrm{co}}(v - \tilde{v}, \tilde{v})\, F(v - \tilde{v}, t)\, F(\tilde{v}, t)\, d\tilde{v} \tag{2.47}$$

$$\bar{F}_{\mathrm{co}}^{-}(v, t) = \frac{F(v, t)}{V_{\mathrm{total}}(t)} \int_0^{v_{\max} - v} \beta_{\mathrm{co}}(v, \tilde{v})\, F(\tilde{v}, t)\, d\tilde{v}. \tag{2.48}$$

Wir modellieren die Aggregationsrate β_{co} wieder als Produkt von Aggregationsfrequenz θ_{co} und der Aggregationseffizienz ϑ_{co}, wobei wir uns wieder auf Coulaloglou und Tavlarides [9] beziehen.

Damit eine ausreichend große Wahrscheinlichkeit gegeben ist, dass zwei Tropfen miteinander kollidieren, muss eine entsprechende Anzahl an Tropfen vorhanden sein. Außerdem muss die Kollisionsenergie groß genug sein. Analog zur Stoßwahrscheinlichkeit zweier Moleküle in der kinetischen Gastheorie beschreiben wir die Aggregationsfrequenz gemäß

$$\theta_{\mathrm{co}}(v, \tilde{v}) = C_5\, \epsilon^{\frac{1}{3}} \left(v^{\frac{2}{3}} + \tilde{v}^{\frac{2}{3}}\right) \left(v^{\frac{2}{9}} + \tilde{v}^{\frac{2}{9}}\right)^{\frac{1}{2}}. \tag{2.49}$$

Mit (2.44) erhalten wir θ_{co} in der Form

$$\begin{aligned} \theta_{\mathrm{co}}(v, \tilde{v}) &= c_{\mathrm{co},1}\, d_{\mathrm{stirr}}^{\frac{2}{3}}\, \omega \left(v^{\frac{2}{3}} + \tilde{v}^{\frac{2}{3}}\right) \left(v^{\frac{2}{9}} + \tilde{v}^{\frac{2}{9}}\right)^{\frac{1}{2}} \\ &= c_{\mathrm{co},3} \left(v^{\frac{2}{3}} + \tilde{v}^{\frac{2}{3}}\right) \left(v^{\frac{2}{9}} + \tilde{v}^{\frac{2}{9}}\right)^{\frac{1}{2}}, \end{aligned} \tag{2.50}$$

wobei wir durch die Wahl

$$c_{\mathrm{co},3} = c_{\mathrm{co},1}\, d_{\mathrm{stirr}}^{\frac{2}{3}}\, \omega \tag{2.51}$$

eine übersichtlichere Darstellung erhalten, dabei ist $c_{\mathrm{co},1} > 0$ eine Aggregationskonstante, siehe dazu auch Tabelle 2.2.

Damit nach erfolgter Kollision die Tropfen zusammenfließen können, muss die Kontaktzeit t_{cnt} größer sein als die Zeit, die benötigt wird, um den Film, der sich aufgrund der kontinuierliche Phase zwischen den Tropfen befindet, zu verdrängen bzw. abreißen zu lassen. Diese so genannte *Aggregations-* bzw. *Koaleszenzzeit* ist wieder eine Materialgröße. Wir bezeichnen sie mit t_{co}. Die Zeit t_{cnt} betrachtet man als Zufallsvariable, der eine Gaußsche Normalverteilung zu Grunde liegt. Damit gilt nach Coulaloglou und Tavlarides [9] für die Aggregationseffizienz ϑ_{co} bzgl. der Durchmesser d und $\tilde{d}$ der kollidierenden Tropfen

$$\vartheta_{\mathrm{co}}(d,\tilde{d}) = \exp\left[-\frac{\bar{t}_{\mathrm{cnt}}(d,\tilde{d})}{\bar{t}_{\mathrm{co}}(d,\tilde{d})}\right], \tag{2.52}$$

wobei mit $\bar{t}_{\mathrm{cnt}}$ und $\bar{t}_{\mathrm{co}}$ jeweils die Durchschnitte von t_{cnt} und t_{co} bezeichnet seien. Für die durchschnittlichen Zeiten gilt gemäß Coulaloglou und Tavlarides [9]

$$t_{\mathrm{cnt}}(d,\tilde{d}) = C_6\, \frac{\mu_{\mathrm{c}}\, \varrho_{\mathrm{c}}\, \epsilon^{\frac{2}{3}} (d+\tilde{d})^{\frac{2}{3}}}{\sigma^2} \left(\frac{d\tilde{d}}{d+\tilde{d}}\right)^4 \tag{2.53}$$

sowie

$$t_{\mathrm{co}}(d,\tilde{d}) = C_7\, (d+\tilde{d})^{\frac{2}{3}} \epsilon^{-\frac{1}{3}} \tag{2.54}$$

und für ϑ_{co} somit

$$\vartheta_{\mathrm{co}}(\upsilon,\tilde{\upsilon}) = \exp\left[-c_{\mathrm{co},2} \frac{\mu_{\mathrm{c}}\, \varrho_{\mathrm{c}}\, d_{\mathrm{stirr}}^2\, \omega^3}{\sigma^2} \left(\frac{\upsilon^{\frac{1}{3}} \tilde{\upsilon}^{\frac{1}{3}}}{\upsilon^{\frac{1}{3}} + \tilde{\upsilon}^{\frac{1}{3}}}\right)^4\right] \tag{2.55}$$

mit einer Aggregationskonstanten $c_{\mathrm{co},2} > 0$, wobei wir (2.44) und die Beziehung (2.41) benutzt haben.

Für die Aggregationsrate β_{co} gilt daher mit den Darstellungen (2.50) und (2.55)

$$\beta_{\mathrm{co}}(\upsilon,\tilde{\upsilon}) = c_{\mathrm{co},3} (\upsilon^{\frac{2}{3}} + \tilde{\upsilon}^{\frac{2}{3}})(\upsilon^{\frac{2}{9}} + \tilde{\upsilon}^{\frac{2}{9}})^{\frac{1}{2}} \exp\left[-c_{\mathrm{co},4} \left(\frac{\upsilon^{\frac{1}{3}} \tilde{\upsilon}^{\frac{1}{3}}}{\upsilon^{\frac{1}{3}} + \tilde{\upsilon}^{\frac{1}{3}}}\right)^4\right], \tag{2.56}$$

wobei wir zur Reduktion der Konstanten

$$c_{\mathrm{co},4} = c_{\mathrm{co},2} \frac{\mu_{\mathrm{c}}\, \varrho_{\mathrm{c}}\, d_{\mathrm{stirr}}^2\, \omega^3}{\sigma^2} \tag{2.57}$$

gesetzt haben, vgl. wieder Tabelle 2.2.

Konstante	Wert	Einheit	Name
$c_{\mathrm{br},1}$	0.4	$[-]$	Bruchkonstante
$c_{\mathrm{br},2}$	0.08	$[-]$	Bruchkonstante
$c_{\mathrm{br},3}$	0.35	$[m^{\frac{2}{3}}/s]$	Konstante
$c_{\mathrm{br},4}$	$4.6 \cdot 10^{-6}$	$[m^{\frac{5}{3}}]$	Konstante
$c_{\mathrm{co},1}$	$2.8 \cdot 10^{-6}$	$[-]$	Aggregationskonstante
$c_{\mathrm{co},2}$	$1.83 \cdot 10^{13}$	$[1/m^2]$	Aggregationskonstante
$c_{\mathrm{co},3}$	$1.46 \cdot 10^{-6}$	$[m^{\frac{2}{3}}/s]$	Konstante
$c_{\mathrm{co},4}$	$6.32 \cdot 10^{15}$	$[-]$	Konstante
c_{φ_1}	2.4	$[-]$	Konstante
c_{φ_2}	4.5	$[-]$	Konstante
μ_{c}	0.001	$[Pa\,s]$	Dynamische Viskosität
ϱ_{c}	998	$[kg/m^3]$	Dichte kontinuierliche Phase (H_2O)
ϱ_{d}	972	$[kg/m^3]$	Dichte disperse Phase ($\ddot{O}l$)
d_{stirr}	0.1	$[m]$	Rührerdurchmesser
ω	3.16 - 5.16	$[1/s]$	Rührerdrehzahl
σ	0.043	$[N/m]$	Oberflächenspannung

Tabelle 2.2.: Materialkonstanten für Öl (63% Kerosin ($C_{10}H_{22}..C_{16}H_{34}$), 37% Dichlorbenzol ($C_6H_4Cl_2$)) in Wasser ($H_2O$).

3. Numerische Behandlung

3.1. Allgemeines

Bei der Lösung der Bilanzgleichung wird ein Zeitschrittverfahren zugrunde gelegt. Dabei können die Differentialoperatoren sowohl implizit als auch explizit diskretisiert werden. Die Integraloperatoren gehen stets explizit in das Zeitschrittverfahren ein. Von diesen hängt dabei sehr stark der Lösungsaufwand für die Bilanzgleichung ab. Unsere Aufgabe besteht nun darin, die Integraloperatoren in einer effizienten Weise darzustellen, wobei uns die Modelle aus dem Abschnitt 2.6, als Beispiel dienen. In diesem Kapitel werden wir geeignete numerischen Methoden dafür vorstellen. Dabei stellt die variable Grenze der Integrale eine zusätzliche Herausforderung dar.

Für die Operatoren, denen die Terme (2.6a) und (2.10b) zugrunde liegen, werden wir die Verfahren zunächst für feste Integralgrenzen beschreiben, anschließend erläutern wir die Vorgehensweise bei variablen Grenzen. Den Operator, der durch (2.10a) charakterisiert ist (Faltungsoperator), werden wir gesondert behandeln. Wir werden uns hier auf den 1-dimensionalen Fall beschränken, da wir nur für diesen konkrete Beispiele vorliegen haben.

Notation 3.1.1: Um die Darstellung übersichtlicher zu gestalten, werden wir im Nachfolgenden die Abhängigkeiten von der Zeit und vom Ort nicht mehr notieren. Außerdem wählen wir etwas allgemeinere und in der Numerik übliche Bezeichnungen. So werden wir die Eigenschaftskoordinaten[1)] $\boldsymbol{e}$ bzw. $\tilde{\boldsymbol{e}}$ durch die (1-dimensionalen) Variablen x bzw. y ersetzten. Die Anzahldichtefunktion F ersetzen wir durch f. Allgemeine Kernfunktionen bezeichnen wir mit κ.

Bemerkung 3.1.2: Aufgrund von Bemerkung 2.4.2 b), gilt mit Notation 3.1.1 für den Definitionsbereich $\mathcal{D}$ der Anzahldichtefunktion f gerade $\mathcal{D}(f) = [0, x_{\max}]$. Für die Theorie benötigen wir eine feste obere Grenze für $x_{\max}$. Dabei machen wir die Annahme, dass stets $x_{\max} \leq 1$ bzw., dass gilt

$$\mathcal{D}(f) = [0, x_{\max}] \subset [0, 1].$$

Dies ist keine wirkliche Einschränkung, da wir die Einheit *Meter* zugrunde legen und somit Individuen bis zu einen Durchmesser von $1m$ bzw. einem Volumen von $1m^3$ zulassen. Dies ist für die betrachteten Modelle in jedem Falle ausreichend. Insbesondere ist auch der Integrationsbereich auf $[0, x_{\max}] \subset [0, 1]$ beschränkt.

[1)] für welche wir schon die speziellen Beispiele Länge (ℓ) und Volumen (v) kennengelernt haben

3.2. Darstellung der Integraloperatoren

Wir möchten zunächst die im Modell auftretenden Integraloperatoren klassifizieren.

Bei der Quelle der Dispergierung (2.6a) tritt die Funktion f nur unter dem Integral auf. Wir definieren daher den *linearen Integraloperator*

$$\begin{aligned}\mathcal{K}_{\text{lin}}[f](x) &:= \int_x^{x_{\max}} \nu(y)\,\varphi\,(x,y)\,\beta_{\text{br}}(y)\,f(y)dy \\ &= \int_x^{x_{\max}} \phi_{\text{br}}(x,y) f(y) dy,\end{aligned} \tag{3.1}$$

wobei die Kernfunktion ϕ_{br} definiert sei gemäß

$$\phi_{\text{br}}(x,y) := \nu(y)\,\varphi(x,y)\,\beta_{\text{br}}(y).$$

Der Vollständigkeit halber wollen wir außerdem die Senke (2.6b) der Dispergierung betrachten. Sie enthält kein Integral und ihre Behandlung erfordert somit nur linearen Aufwand. Sie kann, ähnlich wie die Differentialoperatoren, sowohl implizit als auch explizit diskretisiert werden. Für die explizite Darstellung definieren wir den *Diagonaloperator*

$$\mathcal{K}_{\text{diag}}[f](x) := \beta_{\text{br}}(x)\,f(x). \tag{3.2}$$

Bei der Quelle der Koaleszenz hingegen tritt f quadratisch unter dem Integral auf. Wir definieren somit den *quadratischen Operator* gemäß

$$\mathcal{K}_{\text{quad}}[f](x) := \int_0^x \beta_{\text{co}}(x-y,y) f(x-y) f(y) dy. \tag{3.3}$$

Schließlich tritt die Funktion f bei der Senke der Koaleszenz (2.10b) einmal unter dem Integral und einmal vor dem Integral auf. Den zugehörigen Operator definieren wir gemäß

$$\mathcal{K}_{\text{qlin}}[f](x) := f(x) \int_0^{x_{\max}-x} \beta_{\text{co}}(x,y) f(y) dy, \tag{3.4}$$

wir haben somit ebenfalls einen quadratischen Operator vorliegen. Diesen kann man jedoch als Kombination von linearem Integraloperator und Diagonaloperator auffassen. Um $\mathcal{K}_{\text{qlin}}$ von dem bereits definierten quadratischen Operator zu unterscheiden, bezeichnen wir ihn als *quasilinearen Operator*[1)].

Allen auftretenden Integraloperatoren (3.1), (3.3) und (3.4) liegen Volterrasche Integralgleichungen 1. Art zugrunde. Das zugehörige Integrationsgebiet ist jeweils eine Teilmenge von $[0, x_{\max}]$, d.h. gemäß Bemerkung 3.1.2 gilt insbesondere $x, y \in [0,1]$ für alle in den Gleichungen auftretenden Variablen x und y. Im Gegensatz zu den Randintegralmethoden wollen wir aber kein Gleichungssystem lösen, sondern eine Matrix-Vektor-Multiplikation ausführen.

[1)] Um Verwirrungen vorzubeugen, sei explizit darauf hingewiesen, dass der Begriff *quasilinear* aufgrund der besonderen Struktur des Integraloperators $\mathcal{K}_{\text{qlin}}$ gewählt wurde. Er ist nicht an die Bezeichnung *quasilinear* angelehnt, die mitunter bei der Klassifizierung vonDifferentialgleichungen benutzt wird. Da wir im ersten Fall einen Integraloperator und im zweiten Fall Differentialgleichungen vorliegen haben, sollte die Zuordnung eindeutig sein.

3.3. Diskretisierungsverfahren

In diesem Abschnitt gehen wir auf die numerischen Grundlagen ein, die wir bei der Diskretisierung der Integraloperatoren benötigen.

3.3.1. Basisfunktionen

Ganz allgemein haben wir bei jedem unserer Operatoren $\mathcal{K}$ eine Abbildung von einem Funktionenraum X in einen Funktionenraum Y über einem beschränkten, zusammenhängenden Lipschitz-Gebiet $\Omega \subset \mathbb{R}^d$ vorliegen,

$$\mathcal{K} : X \to Y, \quad f \mapsto \mathcal{K}[f] =: g. \tag{3.5}$$

Gewöhnlich wird es sich bei X und Y um unendlichdimensionale Funktionenräume handeln. In Hinblick auf die praktische Anwendbarkeit werden wir uns für unsere Berechnungen auf diskrete, n-dimensionale Unterräume $X_n \subset X$ und $Y_n \subset Y$, $n < \infty$, beschränken. Für jeden dieser Räume wählen wir üblicherweise Basen, deren Elemente (*Basisfunktionen*) durch kleine Träger charakterisiert sind. Wir sprechen in diesem Zusammenhang auch von *Finite-Elemente-Räumen* und *Finite-Elemente-Basen*. Solche Basen werden wir auch im Rahmen dieser Arbeit benutzen, gehen aber nur an Stellen, an denen es notwendig erscheint, genauer auf die verwendete Basis ein.

Bemerkung 3.3.1: Jeder Basis $\mathcal{B}$ ordnen wir eindeutig eine Indexmenge I zu, so dass $\mathcal{B} = (b_i)_{i \in I}$ gilt. Wir werden im weiteren Verlauf mitunter die Basis mit ihrer Indexmenge identifizieren.

Sehr häufig werden stückweise Polynome als Basisfunktionen eingesetzt. Die beiden wichtigsten sind *stückweise konstante* und *stückweise lineare* Basisfunktionen.

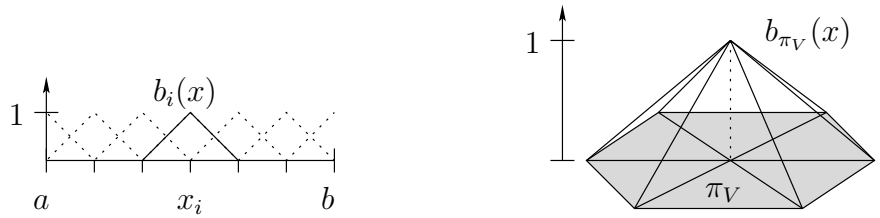


Abbildung 3.1.: Stückweise lineare Basisfunktionen (*Hutfunktionen*).

Beispiel 3.3.2: Sei $\Omega \subset \mathbb{R}$ ein Intervall $[a, b]$ und $a = x_1 < ... < x_n = b$ eine Zerlegung von Ω.

a) Stückweise konstante Basisfunktionen oder stückweise Polynome vom Grad 0 auf Ω sind definiert gemäß

$$b_i(x) = \begin{cases} 1 & \text{für } x \in (x_i, x_{i+1}) \\ 0 & \text{sonst} \end{cases} \quad \text{für} \quad i = 1, ..., n,$$

sie spannen einen $(n-1)$-dimensionalen Funktionenraum über Ω auf. Der Träger von $b_i(x)$ ist gerade $[x_i, x_{i+1}]$.

b) Stückweise lineare Basisfunktionen oder stückweise Polynome vom Grad 1 auf Ω sind definiert gemäß

$$b_i(x) = \begin{cases} \frac{x-x_{i-1}}{x_i-x_{i-1}} & \text{für } x \in (x_{i-1}, x_i) \\ \frac{x-x_{i+1}}{x_i-x_{i+1}} & \text{für } x \in [x_i, x_{i+1}) \\ 0 & \text{sonst} \end{cases}$$

für $i = 2, ..., n-1$ und für $i = 1$ bzw. $i = n$ gemäß

$$b_1(x) = \frac{x - x_2}{x_1 - x_2} \quad \text{bzw.} \quad b_n(x) = \frac{x - x_{n-1}}{x_n - x_{n-1}}.$$

Sie spannen einen n-dimensionalen Funktionenraum über Ω auf. Wenn wir $x_0 := x_1$ und $x_{n+1} := x_n$ setzen, so gilt $\text{supp}(b_i(x)) = [x_{i-1}, x_{i+1}]$, für $i = 1, \dots, n$.

Beispiel 3.3.3: Sei $\Omega \subset \mathbb{R}^3$ eine Oberfläche, I eine Indexmenge und $(\pi_i)_{i\in I} = \Upsilon$ eine Triangulierung von Ω mit $\Omega = \dot{\bigcup}_{i\in I}\pi_i$.

a) Stückweise konstante Basisfunktionen auf Ω sind dann definiert gemäß

$$b_\pi(x) = \begin{cases} 1 & \text{für } x \in \pi \\ 0 & \text{sonst} \end{cases} \quad \forall \pi \in \Upsilon.$$

Die Dimension des Raumes, den die b_π aufspannen, entspricht der Anzahl der Dreiecke. Träger der Basisfunktion b_π ist π.

b) Es bezeichne V_Υ die Menge der Eckpunkte aller Dreiecke $\pi \in \Upsilon$. Stückweise lineare Funktionen b_{π_V}, $\pi_V \in V_\Upsilon$ definieren dann Basisfunktionen auf Ω gemäß

$$b_{\pi_V}(x) = \begin{cases} 1 & \text{für } x = \pi_V \\ 0 & \text{sonst} \end{cases} \quad \forall x \in V_\Upsilon.$$

Die Dimension des Raumes, der durch die b_{π_V} aufgespannt wird, entspricht der Anzahl der Eckpunkte aller Dreiecke, und der Träger einer Basisfunktion b_{π_V} ist gerade die Vereinigung der Dreiecke, die den Eckpunkt π_V gemeinsam haben.

Bemerkung 3.3.4: a) Wir ordnen jeder Basisfunktion b_i einen speziellen Punkt $p(b_i) = p_i \in \Omega \subset \mathbb{R}^d$ zu. Diesen bezeichnen wir auch als *Referenzpunkt* von b_i. Dieser sollte in einem sinnvollen Bezug zu b_i stehen, insbesondere sollte die Zuordnung zu b_i eineindeutig sein. Bei unseren Beispielen wählt man bei konstanten Basisfunktionen gewöhnlich den Mittelpunkt des Trägers und bei den linearen Basisfunktionen für b_i gerade x_i bzw. für b_{π_V} gerade π_V, siehe Abbildung 3.1.

b) Im 1D-Fall führen wir eine Ordnung auf der Menge der Basisfunktionen ein. Eine Basis heißt *geordnet*, wenn für je zwei Basisfunktionen b_i, b_j, aus $i < j$ stets folgt:

$$p(b_i) < p(b_j).$$

Wir sprechen dann auch von *geordneten Basisfunktionen*.

c) Durch die Wahl von Basisfunktionen ergibt sich auch eine Zerlegung von Ω. Den Abstand der Referenzpunkte bzw. das Maximum über alle Referenzpunkte können wir dabei als Maß der Feinheit der Zerlegung ansehen. Man spricht in diesem Zusammenhang auch von der *Gitterweite* h. Ist der Abstand stets gleich, sprechen wir auch von einer *äquidistanten* Zerlegung von Ω.

3.3.2. Projektionsverfahren

Wir wollen unsere Operatoren aus Abschnitt 3.2 mit Hilfe von Projektionsverfahren diskretisieren, dabei gehen wir wieder von dem allgemeinen Operator (3.5) aus, fordern aber $X = Y$, d.h.

$$\mathcal{K} : X \to X, f \mapsto \mathcal{K}[f] := g. \tag{3.6}$$

Eine lineare Abbildung $\Pi_n : X \to X$ bezeichnet man als Projektion, falls $\Pi_n^2 = \Pi_n$. Durch eine solche Projektion Π_n ist eindeutig ein Unterraum $X_n \subseteq X$ definiert, gemäß $X_n := \text{Bild}\,(\Pi_n)$. Man spricht dann auch von einer Projektion *auf* X_n.

Sei nun mit X_n ein n-dimensionalen Unterraum von X bezeichnet, welchem die Projektion $\Pi_n : X \to X$ zugrunde liege. Projektionsverfahren sind durch die Forderung gekennzeichnet, dass der Defekt $d_n := (g - \mathcal{K}_n f)$, $\mathcal{K}_n f := \mathcal{K}\Pi_n(f)$, unter der Projektion Π_n verschwindet, d.h. dass

$$\Pi_n(g - \mathcal{K}_n f) = 0 \tag{3.7}$$

gilt, vergleiche auch Hackbusch [29]. Sei Π_n' nun der duale Operator zu Π_n. Dann ist Π_n' eine Projektion auf dem Dualraum X' von X. Weiter sei das Dualprodukt $\langle ., . \rangle$ auf $X' \times X$ definiert gemäß

$$\langle \varphi, f \rangle := \varphi(f), \quad \forall \varphi \in X'. \tag{3.8}$$

Dann ist die Forderung (3.7) äquivalent zu den Bedingungen (3.9)

$$\varphi(\Pi_n d_n) = 0, \quad \forall \varphi \in X' \tag{3.9a}$$

$$(\Pi_n' \varphi)(d_n) = \varphi_n(d_n) = 0, \quad \forall \varphi \in X' \quad \text{und} \quad \varphi_n := \Pi_n' \varphi \in X_n'. \tag{3.9b}$$

Sei $b_1', ..., b_n'$ eine Basis von X_n', die Forderung (3.7) ist bereits erfüllt, falls (3.9b) für alle Basiselemente von X_n' gilt, d.h. falls

$$b_i'(d_n) = \langle b_i', d_n \rangle = 0, \quad \forall i = 1, ..., n.$$

Damit und aus der Definition des Defekts erhalten wir die diskrete Darstellung

$$\langle b_i', g \rangle = \langle b_i', \mathcal{K}_n f \rangle, \quad \forall i = 1, ..., n.$$

In unserem Fall ist die Funktion f gegeben und die Funktion g gesucht. Da aber g aus dem gleichen Raum stammen soll wie f, können wir g und f bzgl. der gleichen Basis $b_1, ..., b_n$ darstellen in der Form

$$\Pi_n g = \sum_{j=1}^{n} g_j b_j \quad \text{und} \quad \Pi_n f = \sum_{j=1}^{n} f_j b_j,$$

wobei die $g_1, ..., g_n$ und $f_1, ..., f_n$ als *Basiskoeffizienten* von $\Pi_n g$ und $\Pi_n f$ bezeichnet werden. Damit haben wir letztendlich unser Problem diskretisiert gemäß

$$\sum_{j=1}^{n} \langle b_i', b_j \rangle g_j = \sum_{j=1}^{n} \langle b_i', \mathcal{K}_n b_j \rangle f_j, \quad \forall i = 1, ..., n,\ \mathcal{K}_n := \Pi_n \mathcal{K}.$$

Mit den Matrizen $\boldsymbol{K} \in \mathbb{K}^{n \times n}$ und $\boldsymbol{M} \in \mathbb{K}^{n \times n}$, deren Einträge gegeben sind durch

$$K_{ij} = \langle b_i', \mathcal{K}_n b_j \rangle \quad \text{und} \quad M_{ij} = \langle b_i', b_j \rangle,$$

definieren wir das Gleichungssystem

$$\boldsymbol{M}\boldsymbol{g} = \boldsymbol{K}\boldsymbol{f}, \tag{3.10}$$

wobei $\boldsymbol{f}$ und $\boldsymbol{g}$ gerade die Vektoren der Basiskoeffizienten von $\Pi_n f$ und $\Pi_n g$ seien.

Die Matrix $\boldsymbol{K}$ nennen wir *Systemmatrix* und die Matrix $\boldsymbol{M}$ ist die positiv definite Gramsche Matrix, welche üblicherweise *Massematrix* genannt wird. Die Vektoren $\boldsymbol{f}$ und $\boldsymbol{g}$ heißen *Koeffizientenvektoren*. Den Raum X_n bezeichnet man gemeinhin als *Ansatzraum* und den Raum X_n' als *Testraum*.

Beispiel 3.3.5 (Galerkin-Verfahren): Es sei X ein Hilbert-Raum, mit dem Skalarprodukt $\langle ., . \rangle$. Weiter sei $b_1, ..., b_n$ die Basis eines Unterraumes $X_n \subset X$. Wir definieren auf X die Funktionale

$$b_i'(f) := \langle b_i, f \rangle, \quad i = 1, ..., n,\ \forall f \in X. \tag{3.11}$$

Diese bilden damit eine Basis des Dualraumes X_n' von X_n. Der Testraum wird über (3.11) mit dem Ansatzraum identifiziert.

Beispiel 3.3.6 (Kollokations-Verfahren): Sei $X = C(\Omega)$ der Banachraum der stetigen Funktionen auf Ω und seien $\xi_1, ..., \xi_n$ disjunkte Stützstellen (*Kollokationspunkte*) auf Ω gegeben. Wir definieren auf X die Funktionale

$$b_i'(f) := f(\xi_i), \quad i = 1, ..., n,\ \forall f \in X. \tag{3.12}$$

Diese sind Elemente des Dualraumes X' von X und bilden somit die Basis eines Unterraumes X_n' von X', welchen wir als Testraum wählen. Das Dualprodukt ist nach (3.8) bereits durch (3.12) definiert. Die Funktionale b_i' bezeichnet man auch als *Diracsche Deltafunktionale* und schreibt $\delta_{\xi_i} = b_i'$.

3.4. Diskretisierung der Operatoren

Im Folgenden gehen wir auf die Besonderheiten der einzelnen Operatoren bei der Diskretisierung ein, dabei nehmen wir zunächst feste Integralgrenzen an. Den quadratischen Operator behandeln wir später gesondert in Abschnitt 3.8.

Massematrix

Die Massematrix $\boldsymbol{M}$ ist unabhängig vom zugrunde liegenden Operator. Im Fall des Galerkin-Verfahrens besitzt sie stets Bandgestalt

$$M_{ij} = \int_{\mathrm{supp}(b_i)\cap\mathrm{supp}(b_j)} b_i(x)\, b_j(x)\, dx \quad \text{für} \quad i,j = 1,\dots,n. \tag{3.13}$$

Für stückweise konstante Basisfunktionen haben wir gemäß Beispiel 3.3.2 a) gerade

$$M_{ij} = \begin{cases} \int_{x_i}^{x_{i+1}} dx = x_{i+1} - x_i & \text{für} \quad i = j \\ 0 & \text{sonst,} \end{cases}$$

für $i,j = 1,\dots,n$, d.h. $\boldsymbol{M}$ ist eine Diagonalmatrix.

Für das Kollokationsverfahren gilt noch einfacher

$$M_{ij} = b_j(\xi_i) \quad \text{für} \quad i,j = 1,\dots,n$$

und somit ist $\boldsymbol{M}$ für stückweise konstante und stückweise lineare Basisfunktionen die Einheitsmatrix.

Obwohl das Hauptaugenmerk auf der Multiplikation der Matrix $\boldsymbol{K}$ mit dem Koeffizientenvektor $\boldsymbol{f}$ liegt, müssen wir das System (3.10) noch nach $\boldsymbol{g}$ auflösen. Wie wir gesehen haben, besitzt $\boldsymbol{M}$ in unserem Fall eine sehr einfache Struktur, so dass dies kein Problem darstellt.

Bemerkung 3.4.1: Wir werden im Weiteren stets das Galerkin-Verfahren zugrunde legen mit $X = L^2(\Omega)$.

3.4.1. Fredholm-Operatoren

Zunächst nehmen wir bei unseren Operatoren feste Integralgrenzen an. Wir bezeichnen Operatoren dieser Art auch als *Fredholm-Operatoren* oder *Fredholm-Integraloperatoren.*

Linearer Operator

Wir betrachten anstelle des allgemeinen Operators in (3.6), den linearen Integraloperator aus (3.1) mit festen Integralgrenzen, d.h.

$$\mathcal{K}[f](x) = \int_0^{x_{\max}} \kappa(x,y)\, f(y)\, dy. \tag{3.14}$$

Damit haben im Falle des Galerkin-Verfahrens die Komponenten der Systemmatrix $\boldsymbol{K}$ die Gestalt

$$K_{ij} = \int_{\mathrm{supp}(b_i)} b_i(x) \int_{\mathrm{supp}(b_j)} \kappa(x,y)\, b_j(y)\, dy\, dx \quad \text{für} \quad i,j = 1,\dots,n.$$

Für den Fall konstanter Basisfunktionen gilt gemäß Beispiel 3.3.2 a)

$$K_{ij} = \int_{x_i}^{x_{i+1}} \int_{x_j}^{x_{j+1}} \kappa(x,y)\, dy\, dx.$$

Für das Kollokationsverfahren haben die Einträge der Systemmatrix $\boldsymbol{K}$ die Form

$$K_{ij} = \int_{\mathrm{supp}(b_j)} \kappa(\xi_i, y)\, b_j(y)\, dy \quad \text{bzw.} \quad K_{ij} = \int_{x_i}^{x_{i+1}} \kappa(\xi_i, y)\, dy$$

im Falle stückweise konstanter Basisfunktionen, für $i, j = 1, \dots, n$.

Quasilinearer Operator

Zunächst führen wir die folgende Definition ein.

Definition 3.4.2 (Hadamard-Multiplikation): Die *komponentenweise Multiplikation* oder auch *Hadamard-Multiplikation* definieren wir wie folgt.

a) Seien zwei Vektoren $\boldsymbol{u} = (u_1, ..., u_n)^\top, \boldsymbol{v} = (v_1, ..., v_n)^\top \in \mathbb{K}^n$ gegeben, dann definieren wir die komponentenweise Multiplikation durch

$$\boldsymbol{u} \circ \boldsymbol{v} := (u_1 \cdot v_1, ..., u_n \cdot v_n)^\top. \tag{3.15a}$$

b) Für zwei Matrizen $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{K}^{m\times n}$ mit Einträgen A_{ij}, B_{ij} definieren wir die Matrix $\boldsymbol{A} \circ \boldsymbol{B} \in \mathbb{K}^{m\times n}$ mit Einträgen AB_{ij} entsprechend

$$AB_{ij} := A_{ij} \cdot B_{ij} \quad \text{für} \quad i = 1, \dots, m \quad \text{und} \quad j = 1, \dots, n. \tag{3.15b}$$

Dabei legen wir fest, dass die Hadamard-Multiplikation $\circ$ Vorrang vor der normalen Matrix-Vektor- bzw. Matrix-Matrix-Multiplikation hat.

Der quasilineare Operator besitzt nach (3.4) für feste Integralgrenzen die Form

$$\mathcal{K}[f](x) = f(x) \int_0^{x_{\max}} \kappa(x, y)\, f(y)\, dy.$$

Nach Diskretisierung mittels Galerkin-Verfahren, stellen sich die Einträge der Systemmatrix $\boldsymbol{K}$ für $i, j = 1, \dots, n$ wie folgt dar

$$K_{ij} = \sum_{l=1}^{n} f_l \int_{\mathrm{supp}(b_i)\cap\mathrm{supp}(b_l)} b_i(x) b_l(x) \int_{\mathrm{supp}(b_j)} \kappa(x, y)\, b_j(y)\, dy\, dx. \tag{3.16}$$

Diese Darstellung ist ungünstig, da $\boldsymbol{K}$ noch von $\boldsymbol{f}$ abhängt. Wir betrachten (3.16) für stückweise konstante Basisfunktionen und erhalten die K_{ij} in der Form

$$K_{ij} = f_i \int_{x_i}^{x_{i+1}} \int_{x_j}^{x_{j+1}} \kappa(x, y)\, dy\, dx.$$

Wir können nun die von $\boldsymbol{f}$ unabhängige Matrix $\tilde{\boldsymbol{K}}$ definieren, mit Einträgen gemäß

$$\tilde{K}_{ij} = \int_{x_i}^{x_{i+1}} \int_{x_j}^{x_{j+1}} \kappa(x, y)\, dy\, dx. \tag{3.17}$$

Somit können wir das Gleichungssystem (3.10) mit Hilfe der Hadamard-Multiplikation aus Definition 3.4.2 für den quasilinearen Fall darstellen gemäß

$$\boldsymbol{M}\boldsymbol{g} = \boldsymbol{f} \circ (\tilde{\boldsymbol{K}}\boldsymbol{f}). \tag{3.18}$$

Diagonaloperator

Der Diagonaloperator ist nach (3.2) gegeben durch

$$\mathcal{K}[f](x) = \beta_{\mathrm{br}}(x)\, f(x).$$

Die Komponenten der Systemmatrix $\boldsymbol{K}$ haben für $i, j = 1, \ldots, n$ die Form

$$K_{ij} = \int_{\mathrm{supp}(b_i)\cap\mathrm{supp}(b_j)} \beta_{\mathrm{br}}(x)\, b_i(x) b_j(x)\, dx,$$

wenn wir das Galerkin-Verfahren zugrunde legen. Somit gilt für stückweise konstante Basisfunktionen

$$K_{ij} = \begin{cases} \int_{x_i}^{x_{i+1}} \beta_{\mathrm{br}}(x)\, dx & \text{für} \quad i = j \\ 0 & \text{sonst.} \end{cases}$$

Da in diesem Falle die Einträge K_{ij} unabhängig von j sind, können wir das System (3.10) einfacher mit Hilfe der Hadamard-Multiplikation schreiben

$$\boldsymbol{M}\boldsymbol{g} = \boldsymbol{k} \circ \boldsymbol{f}, \tag{3.19}$$

wobei die Einträge des Vektors $\boldsymbol{k}$ gerade gegeben sind durch $k_i := \int_{x_i}^{x_{i+1}} \beta_{\mathrm{br}}(x)\, dx$.

Quadratischer Operator

Siehe Abschnitt 3.8.

3.4.2. Volterra-Operatoren

In diesem Abschnitt gehen wir auf die Darstellung der variablen Integralgrenzen bei unseren Operatoren ein. Diese Grenzen bezeichnen wir im Folgenden auch als *Volterra-Grenzen*. Siehe dazu auch Abschnitt 3.2. Wir sprechen in diesem Zusammenhang ebenfalls von *Volterra-Integraloperatoren* oder *Volterra-Operatoren.*

Wir beginnen mit der Betrachtung eines allgemeinen linearen Volterra-Operators. Dabei seien die variablen Integralgrenzen durch die *Volterra-Funktionen* $v_{\mathrm{a}}, v_{\mathrm{b}} \in C^0([0,1])$ gegeben gemäß

$$\mathcal{K}[f](x) = \int_{v_{\mathrm{a}}(x)}^{v_{\mathrm{b}}(x)} \kappa(x,y)\, f(y) dy, \quad \text{mit} \quad v_{\mathrm{a}}(x) \le v_{\mathrm{b}}(x), \quad \forall x \in [0,1], \tag{3.20}$$

siehe Abbildung 3.2 a). Nach Galerkin-Diskretisierung können wir die zugehörige Systemmatrix $\boldsymbol{K}$ in der Form schreiben

$$K_{ij} = \int_{\mathrm{supp}(b_i)} b_i(x) \int_{\mathrm{supp}(b_j)\cap\{[v_{\mathrm{a}}(x), v_{\mathrm{b}}(x)]\,|\,x\in\mathrm{supp}(b_j)\}} \kappa(x,y)\, b_j(y)\, dy\, dx, \tag{3.21}$$

$i, j = 1, \ldots, n$. Hierbei sind die inneren Integralgrenzen von x abhängig. Zum besseren Verständnis führen wir noch die folgenden Bezeichnungen ein.

Definition 3.4.3: Für festes $i, j \in \{1, \dots, n\}$ sowie $x \in \operatorname{supp}(b_i)$ können wir für die Einträge K_{ij} der Galerkin-Systemmatrix $\boldsymbol{K}$ aus (3.21), in Abhängigkeit von den Volterra-Grenzen und den verwendeten Basisfunktionen, drei Fälle unterscheiden:

a) $\operatorname{supp}(b_j) \cap [v_{\mathrm{a}}(x), v_{\mathrm{b}}(x)] = \varnothing$, d.h., es gilt $K_{ij} = 0$.

b) $\operatorname{supp}(b_j) \cap [v_{\mathrm{a}}(x), v_{\mathrm{b}}(x)] = \operatorname{supp}(b_j)$, d.h., wir haben den gleichen Eintrag K_{ij} vorliegen, wie im Falle der festen Integralgrenzen.

c) $\operatorname{supp}(b_j) \cap [v_{\mathrm{a}}(x), v_{\mathrm{b}}(x)] \neq \varnothing$ und $\operatorname{supp}(b_j) \cap [v_{\mathrm{a}}(x), v_{\mathrm{b}}(x)] \neq \operatorname{supp}(b_j)$, d.h. bei der Berechnung des Eintrages K_{ij} taucht explizit die Volterra-Grenze im Integral auf.

Entsprechend bezeichnen wir die Matrixeinträge in den Fällen a), b) bzw. c), als *Null-*, *Fredholm-* bzw. *Volterra-Einträge*. Weiterhin bezeichnen wir mit $F(\boldsymbol{K})$ bzw. $V(\boldsymbol{K})$ die Menge aller Fredholm- bzw. Volterra-Einträge von $\boldsymbol{K}$.

Bemerkung 3.4.4:

a) Die Fredholm-Einträge der Systemmatrix eines Integraloperators mit variabler Grenze stimmen mit den entsprechenden Einträgen der Systemmatrix[1)] des gleichen Operators mit fester Integralgrenze überein.

b) Wenn wir im Folgenden von „von Null verschiedenen Einträgen“ bzw. „Nichtnull-Einträgen“ sprechen, so meinen wir damit entweder Fredholm- oder Volterra-Einträge.

Linearer Operator

Der lineare Integraloperator aus (3.1) besitzt mit variablen Integralgrenzen die Form

$$\mathcal{K}[f](x) = \int_{x}^{x_{\max}} \kappa(x, y)\, f(y)\, dy. \tag{3.22}$$

Entsprechend besitzen die Einträge der Galerkin-Systemmatrix $\boldsymbol{K}$ für $i, j = 1, \dots, n$ die Gestalt

$$K_{ij} = \int_{\operatorname{supp}(b_i)} b_i(x) \int_{\operatorname{supp}(b_j) \cap \{[x, x_{\max}] \mid x \in \operatorname{supp}(b_j)\}} \kappa(x, y)\, b_j(y)\, dy\, dx.$$

Somit gilt für stückweise konstante Basisfunktionen

$$K_{ij} = \begin{cases} 0 & \text{für} \quad x_{j+1} \leq x_i \\ \int_{x_i}^{x_{i+1}} \int_{x_j}^{x_{j+1}} \kappa(x, y)\, dy\, dx & \text{für} \quad x_j \geq x_{i+1} \\ \int_{x_i}^{x_{i+1}} \int_{x}^{x_{j+1}} \kappa(x, y)\, dy\, dx & \text{sonst.} \end{cases} \tag{3.23}$$

Siehe bzgl. des Integrationsgebietes auch Abbildung 3.2 b).

[1)] welche natürlich ausschließlich aus Fredholm-Einträgen besteht

Bemerkung 3.4.5: Legen wir eine geordnete Basis gemäß Bemerkung 3.3.4 b) zugrunde, so können wir die Einträge der Systemmatrix $\boldsymbol{K}$ aus (3.23) alternativ darstellen

$$K_{ij} = \begin{cases} 0 & \text{für} \quad j+1 \leq i \\ \int_{x_i}^{x_{i+1}} \int_{x_j}^{x_{j+1}} \kappa(x,y)\, dy\, dx & \text{für} \quad j \geq i+1 \\ \int_{x_i}^{x_{i+1}} \int_{x}^{x_{j+1}} \kappa(x,y)\, dy\, dx & \text{sonst,} \end{cases} \tag{3.24}$$

damit haben wir insbesondere eine rechte obere Dreiecksmatrix vorliegen.

Quasilinearer Operator

Der quasilineare Operator besitzt nach (3.4) mit Volterra-Grenzen die Darstellung

$$\mathcal{K}[f](x) = f(x) \int_0^{x_{\max}-x} \kappa(x,y)\, f(y)\, dy. \tag{3.25}$$

Nach Diskretisierung mittels Galerkin-Verfahren, stellen sich die Einträge der Systemmatrix $\boldsymbol{K}$ in Abhängigkeit vom Koeffizientenvektor $\boldsymbol{f}$ für $i,j = 1,\dots,n$ wie folgt dar

$$K_{ij} = \sum_{l=1}^{n} f_l \int_{\operatorname{supp}(b_i)\cap\operatorname{supp}(b_l)} b_i(x) b_l(x) \int_{\operatorname{supp}(b_j)\cap\{[0,x_{\max}-x]\,|\,x\in\operatorname{supp}(b_i)\cap\operatorname{supp}(b_l)\}} \kappa(x,y)\, b_j(y)\, dy\, dx.$$

Für die von $\boldsymbol{f}$ unabhängige Darstellung benutzen wir wieder die Matrix $\tilde{\boldsymbol{K}}$ aus Gleichungssystem (3.18) und legen wie gewöhnlich stückweise konstante Basisfunktionen zugrunde. Weiterhin setzen wir eine äquidistante Zerlegung des Integrationsgebietes voraus. Es gelte $x_i = ih, i = 0,\dots,n$ mit $h = x_{\max}/n$ bzw. $h = x_n/n$, siehe dazu auch Abschnitt 3.8.1. Damit erhalten wir für die Einträge von $\tilde{\boldsymbol{K}}$

$$\tilde{K}_{ij} \begin{cases} 0 & \text{für} \quad x_{j+1} \leq x_n - x_{i+1} \\ \int_{x_i}^{x_{i+1}} \int_{x_j}^{x_{j+1}} \kappa(x,y)\, dy\, dx & \text{für} \quad x_j \geq x_n - x_i \\ \int_{x_i}^{x_{i+1}} \int_{x_j}^{x_n-x} \kappa(x,y)\, dy\, dx & \text{sonst.} \end{cases} \tag{3.26}$$

Vergleiche ebenfalls Abbildung 3.2 c).

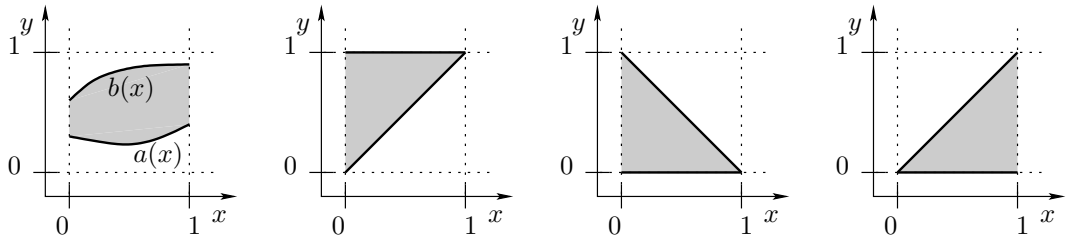


Abbildung 3.2.: Verschiedene Volterra-Integrationsgebiete.

Bemerkung 3.4.6: Wenn wir wieder geordnete Basen einsetzen, haben wir hier die Systemmatrix $\boldsymbol{K}$ in Gestalt einer linken oberen Dreiecksmatrix vorliegen.

Diagonaloperator

Da dies kein Integraloperator ist, existieren keine Volterra-Grenzen und somit ändert sich im Fall des Diagonaloperators nichts. Wir haben daher die gleiche Darstellung wie im Fredholm-Fall vorliegen, siehe Gleichung (3.19).

Quadratischer Operator

Siehe Abschnitt 3.8.

3.5. Spezielle Kernfunktionen

Wir beschreiben in diesem Abschnitt die Eigenschaften *entarteter* Kerne, mit deren Hilfe man die Systemmatrizen auf eine günstige Weise darstellen kann. Dies werden wir zunächst wieder für feste Integralgrenzen verwirklichen, in Abschnitt 3.5.3 wenden wir unser Wissen dann auf Operatoren mit Volterra-Grenzen an.

3.5.1. Separable Kerne

Einige der Kerne, die in der Populationsdynamik Verwendung finden, lassen sich als *separable* Kerne darstellen, z.B. der Smoluchowski-Kern, vgl. (2.35) mit $c_{\text{smol}} = 0$ im Abschnitt 2.6.2 bzw. Beispiel 3.7.1 a) im Abschnitt 3.7.1. Kerne dieser Art werden wir nachfolgend näher betrachten.

Definition 3.5.1 (separabler Kern): Ein Kern $\kappa(x, y)$, der sich als endliche Summe von in den Komponenten x und y getrennten Funktionen darstellen lässt, d.h. in der Form

$$\kappa(x, y) = \sum_{l=1}^{k_{\text{sep}}} \Phi_l(x)\Psi_l(y), \tag{3.27}$$

bezeichnen wir als *separablen*, *ausgearteten* oder auch *entarteten* Kern. Dabei bezeichne $k_{\text{sep}} \in \mathbb{N}$ den *Separationsrang*.

Aufgrund der Separabilität, können wir unsere Systemmatrix $\boldsymbol{K}$ aus dem Abschnitt 3.3.2 auf natürliche Weise als das Produkt von Matrizen $\boldsymbol{S} \in \mathbb{K}^{m \times k_{\text{sep}}}$ und $\boldsymbol{T} \in \mathbb{K}^{k_{\text{sep}} \times n}$ darstellen[1)]. Zur Veranschaulichung diene das Galerkin-Verfahren.

Beispiel 3.5.2: Wir betrachten wieder den linearen Operator (3.14)

$$\mathcal{K}[f](x) = \int_0^{x_{\max}} \kappa(x, y)\, f(y)\, dy.$$

[1)] m bzw. n entsprechen der Dimension des Test- bzw. Ansatzraumes

Die Einträge der Galerkin-Systemmatrix $\boldsymbol{K}$ lassen sich bei separabler Kernfunktion in der Form

$$\begin{aligned} K_{ij} &= \int_{\mathrm{supp}(b_i)} b_i(x) \int_{\mathrm{supp}(b_j)} \kappa(x,y)\, b_j(y)\, dy\, dx \\ &= \sum_{l=1}^{k_{\mathrm{sep}}} \int_{\mathrm{supp}(b_i)} \Phi_l(x)\, b_i(x) dx \int_{\mathrm{supp}(b_j)} \Psi_l(y)\, b_j(y)\, dy \end{aligned}$$

schreiben. Dann sind durch

$$S_{il} = \int_{\mathrm{supp}(b_i)} \Phi_l(x)\, b_i(x)\, dx \tag{3.28a}$$

und

$$T_{lj} = \int_{\mathrm{supp}(b_j)} \Psi_l(y)\, b_j(y)\, dy \tag{3.28b}$$

mit $l = 1, ..., k_{\mathrm{sep}}$, gerade die Matrizen $\boldsymbol{S} \in \mathbb{K}^{n \times k_{\mathrm{sep}}}$ und $\boldsymbol{T} \in \mathbb{K}^{k_{\mathrm{sep}} \times n}$ definiert.

Definition 3.5.3 (R(k)-Matrizen): Eine Matrix $\boldsymbol{N} \in \mathbb{K}^{m \times n}$ bezeichnen wir als *R(k)-Matrix*, *Rang-k-Matrix* oder *Niedrigrang-Matrix*, wenn sie in der Darstellung

$$\boldsymbol{N} = \boldsymbol{S}\boldsymbol{T} \quad \text{mit} \quad \boldsymbol{S} \in \mathbb{K}^{m \times k}, \boldsymbol{T} \in \mathbb{K}^{k \times n} \tag{3.29}$$

vorliegt bzw. in der Darstellung

$$\boldsymbol{N} = \sum_{l=1}^{k} \boldsymbol{s}_l\, \boldsymbol{t}_l^{\top} \quad \text{mit} \quad \boldsymbol{s}_l \in \mathbb{K}^m, \boldsymbol{t}_l \in \mathbb{K}^n. \tag{3.30}$$

Insbesondere gilt $\mathrm{rang}(\boldsymbol{N}) \leq \mathrm{k}$.

Aus der speziellen Struktur von R(k)-Matrizen ergeben sich einige günstige Eigenschaften.

Folgerung 3.5.4: *Sei $\boldsymbol{N} = \boldsymbol{S}\boldsymbol{T}$ eine R(k)-Matrix, $\boldsymbol{S} \in \mathbb{K}^{m \times k}, \boldsymbol{T} \in \mathbb{K}^{k \times n}$.*

a) Für die Darstellung von $\boldsymbol{N}$ in Form seiner Komponenten $\boldsymbol{S}$ und $\boldsymbol{T}$ brauchen wir nur $k(n+m)$ Zahlen zu speichern.

b) Die Matrix-Vektor-Multiplikation $\boldsymbol{x} \mapsto \boldsymbol{y} := \boldsymbol{N}\boldsymbol{x}$ von $\boldsymbol{N}$ mit einem Vektor $\boldsymbol{x} \in \mathbb{K}^n$ nehmen wir in zwei Schritten vor:

1. *Berechne $\boldsymbol{z} := \boldsymbol{T}\boldsymbol{x} \in \mathbb{K}^k$.*
2. *Berechne $\boldsymbol{y} := \boldsymbol{S}\boldsymbol{z} \in \mathbb{K}^m$.*

Dies bedeutet, wir müssen nur $k(2n-1) + m(2k-1) \leq 2k(m+n)$ Operationen bei einer Matrix-Vektor-Multiplikation ausführen.

c) Bei der Matrix-Vektor-Multiplikation der transponierten Matrix $\boldsymbol{N}^\top = \boldsymbol{T}^\top \boldsymbol{S}^\top$ gehen wir analog zu b) vor, und die Anzahl der Operationen besitzt die gleiche Komplexität $\mathcal{O}(k(n+m))$.

Die Vorteile separabler Kernfunktionen sind somit offensichtlich, durch sie ist auf natürliche Weise eine Darstellung der Systemmatrix $\boldsymbol{K}$ als R(k)-Matrix mit $k = k_{\text{sep}}$ möglich. Damit erreichen wir für $\boldsymbol{K}$ nach Folgerung 3.5.4 eine Komplexität von $\mathcal{O}(k_{\text{sep}} n)$, d.h. eine Reduktion des ursprünglich quadratischen Aufwands auf linearen.

3.5.2. Approximation durch separable Kerne

Im vorigen Abschnitt haben wir gesehen, dass separable Kernfunktionen zu linearem Aufwand führen. Einige der Kernfunktionen aus der Populationsdynamik besitzen bereits diese Eigenschaft, wir wollen sie aber auch für Kerne ausnutzen, die nicht separabel sind. Eine Möglichkeit ist die Approximation der Kernfunktion durch *Lagrange-Polynome*, siehe dazu auch Anhang B.

Definition 3.5.5 (Lagrange-Polynome): Das *i-te Lagrange-Polynom L_i^k vom Grade k* bzgl. der paarweise verschiedenen Stützstellen $\xi_0, \ldots, \xi_k$ ist definiert gemäß

$$L_i^k(x) := \prod_{j=0, i\neq j}^{k} \frac{x - \xi_j}{\xi_i - \xi_j} \quad \text{für} \quad i = 0, \ldots, k.$$

Definition 3.5.6 (Interpolationsoperator): Sei $[a, b]$ ein reelles Intervall und seien $\xi_i \in [a, b]$ paarweise verschiedenen Stützstellen, $i = 0, \ldots, k$. Weiter sei L_i^k das i-te Lagrange-Polynom bzgl. der Stützstellen ξ_i und $f \in C([a, b])$ eine stetige Funktion. Dann bezeichnet man den Operator, welcher definiert ist gemäß

$$\Pi_L^k[f](x) = \sum_{i=0}^{k} f(\xi_i)\, L_i^k(x),$$

als *Lagrangeschen Interpolationsoperator* oder als *Operator der Lagrange-Approximation.*

Besitzt die approximierte Funktion Ableitungen bis zu einer gewissen Ordnung, so lässt sich der Approximationsfehler durch folgendes Lemma abschätzen.

Lemma 3.5.7: *Sei die Funktion f $(k+1)$-mal stetig differenzierbar auf dem Intervall $[a, b]$. Sei weiter Π_L^k der Lagrangesche Interpolationsoperator und $\xi_0, \ldots, \xi_k \in [a, b]$ die zugehörigen Stützstellen. Dann gibt es für alle $x \in [a, b]$ ein $\xi(x) \in [a, b]$, so dass gilt*

$$f(x) - \Pi_L^k[f](x) = \frac{f^{k+1}(\xi(x))}{(k+1)!} \cdot (x - \xi_0) \cdot \ldots \cdot (x - \xi_k).$$

Beweis: Siehe etwa Stoer [45], Satz 2.1.4.1. □

Die Fehlerschranke aus Lemma 3.5.7 formulieren wir unabhängig von der Funktion ξ etwas allgemeiner, gemäß

$$\|f - \Pi_L^k[f]\|_{\infty,[a,b]} \leq \frac{\|f^{(k+1)}\|_{\infty,[a,b]}}{(k+1)!} \max_{x\in[a,b]} \sum_{i=0}^{k} |x - \xi_i|. \tag{3.31}$$

Die Schranke (3.31) hängt von zwei Termen ab. Der erste Term, $\|f^{(k+1)}\|_{\infty,[a,b]}/(k+1)!$ hängt, abgesehen vom Intervall $[a,b]$, ausschließlich von der Glattheit der Funktion f ab, der zweite $\max_{x\in[a,b]} \sum_{i=0}^{k} |x-\xi_i|$ ausschließlich von der Verteilung der Stützstellen.

Betrachten wir zunächst die Stützstellenverteilung. Wir suchen ξ_i, $i = 0, \ldots, k$, so dass $\max_{x\in[a,b]} \sum_{i=0}^{k} |x - \xi_i|$ minimal wird. Zur Vereinfachung nehmen wir zunächst $[a,b] = [-1,1]$ an. Es ist aber bekannt, dass unter allen Polynomen vom Grad $n+1$ gerade das *normierte Tschebyscheff-Polynom* $T_{n+1}/2^n$ die kleinste Maximumsnorm auf dem Intervall $[-1,1]$ besitzt, zur Definition siehe Anhang B. Die optimale Wahl ist daher $\sum_{i=0}^{k} |x - \xi_i| = T_{k+1}/2^k$, wobei die $k+1$ Nullstellen von T_{k+1} gerade den gesuchten Stützstellen ξ_i, $i = 0, \ldots, k$ entsprechen. Sie sind auf $[-1,1]$ gegeben durch

$$\xi_i := \cos\left(\frac{2i+1}{n+1}\frac{\pi}{2}\right), \quad i = 0, \ldots, k \tag{3.32}$$

und werden auch als *Tschebyscheff-Knoten* bezeichnet. Mithilfe der affinen Transformation $\alpha(x) = [(a+b) + (b-a)x]/2$, erhalten wir die entsprechenden Stützstellen auf dem Intervall $[a,b]$. Da für T_{k+1}

$$\max_{x\in[-1,1]} |T_{k+1}/2^k| = \max_{x\in[-1,1]} \sum_{i=0}^{k} |x - \xi_i| \leq 2^k$$

gilt, haben wir auf dem Intervall $[a,b]$ unter Verwendung von α die Abschätzung

$$\max_{x\in[a,b]} \sum_{i=0}^{k} |x - \xi_i| \leq 2\left(\frac{|b-a|}{4}\right)^{k+1}. \tag{3.33}$$

Betrachten wir nun den Term $\|f^{(k+1)}\|_{\infty,[a,b]}/(k+1)!$. Um den Fehler in (3.31) klein zu halten, sollten die Ableitungen von f in gewisser Weise beschränkt sein. Gilt z.B. auf $[a,b]$ eine Abschätzung der Form

$$\frac{f^{(k+1)}}{(k+1)!} \leq c_{\text{glatt}}^{-k}, \tag{3.34}$$

wobei $c_{\text{glatt}} > 1$ hinreichend groß sei, dann können wir den Fehler durch den Polynomgrad k kontrollieren.

Ist nun eine Kernfunktion κ glatt in dem Sinne, dass $\kappa(x,.)$ der Abschätzung (3.34) genügt, dann interpolieren wir κ auf $[a,b]$ in der zweiten Variable gemäß

$$\kappa(x,y) \sim \Pi_L^k[\kappa(x,.)](y) = \sum_{l=0}^{k} \kappa(x,\xi_l) L_l(y), \tag{3.35}$$

wobei wir Tschebyscheff-Knoten benutzen. Da die Funktion $\Pi_L^k[\kappa]$ separabel ist, können wir wieder die Vorteile der R(k)-Matrizen ausnutzen, wobei für den Separationsrang aus dem vorigen Abschnitt gerade $k_{\text{sep}} = k + 1$ gilt. Den entstehenden Interpolationsfehler können wir gemäß (3.33) und (3.34) abschätzen durch

$$|\kappa - \Pi_L^k[\kappa]| \leq 2\, c_{\text{glatt}} \left(\frac{|b-a|}{4\, c_{\text{glatt}}} \right)^{k+1}.$$

Falls nun $|b-a|/(4\, c_{\text{glatt}}) < 1$ gilt, was durch eine geeignete Wahl von a und b stets erreicht werden kann, lässt sich der Interpolationsfehler über k kontrollieren.

3.5.3. Separable Kernfunktionen und variable Integralgrenzen

Wir hatten bis jetzt keine Anforderung an die Reihenfolge der Basiselemente der benutzten Funktionenräume gestellt. Für die Konzepte, die wir in diesem Abschnitt vorstellen wollen, benötigen wir, im Zusammenhang mit den variablen Grenzen, die folgende Bedingung.

Voraussetzung 3.5.8 (Geordnete Basen): *Es seien die von uns betrachteten Basen der Funktionenräume stets geordnet, siehe Bemerkung 3.3.4 b). Dies kann gegebenenfalls durch eine Umordnung der Basiselemente erreicht werden.*

Wenn wir mit Voraussetzung 3.5.8 die zu unseren Volterra-Integraloperatoren gehörigen Systemmatrizen aufstellen, erhalten wir aufgrund der variablen Grenzen Matrizen, die Dreiecksstruktur besitzen, vgl. hierzu Bemerkungen 3.4.5 und 3.4.6. Damit haben die Systemmatrizen insbesondere vollen Rang. Somit lassen sie sich für separable Kernfunktionen nicht auf natürliche Weise als R(k)-Matrizen darstellen.

In diesem Abschnitt stellen wir Methoden vor, mit denen eine effiziente Behandlung von Volterra-Operatoren möglich ist. Sei dazu im Folgenden Voraussetzung 3.5.8 stets erfüllt und die Kernfunktion κ separabel.

Wir erläutern unser Vorgehen zunächst am Beispiel des linearen Operators (3.22), wobei wir das Galerkin-Verfahren mit stückweise konstanten Basisfunktionen benutzen. Mit festen Grenzen hat dieser Operator die Form

$$\mathcal{K}_{\text{F}}[f](x) = \int_0^{x_{\max}} \kappa(x,y)\, f(y)\, dy = \sum_{l=1}^{k_{\text{sep}}} \int_0^{x_{\max}} \Phi_l(x) \Psi_l(y)\, f(y)\, dy.$$

Sei $\boldsymbol{N} \in \mathbb{R}^{n \times n}$ die zugehörige Systemmatrix, welche sich gemäß Abschnitt 3.5.1 als Niedrigrangmatrix $\boldsymbol{N} = \boldsymbol{S}\boldsymbol{T}$ darstellen lässt. Nach (3.30) aus Definition 3.5.3 kann man die Einträge von $\boldsymbol{N}$ mit Hilfe von $\boldsymbol{S}$ und $\boldsymbol{T}$ auch in der Form

$$N_{ij} = \sum_{l=1}^{k_{\text{sep}}} S_{il}\, T_{lj} \quad \text{für} \quad i,j = 1, \ldots, n \tag{3.36}$$

schreiben.

Die Einträge von $\boldsymbol{S}$ und $\boldsymbol{T}$ besitzen im Fall stückweise konstanter Basisfunktionen die Gestalt

$$S_{il} = \int_{x_i}^{x_{i+1}} \Phi_l(x)\,dx \quad \text{und} \quad T_{lj} = \int_{x_j}^{x_{j+1}} \Psi_l(y)\,dy, \tag{3.37}$$

für $i, j = 1, \ldots, n$ und $l = 1, \ldots, k_{\text{sep}}$.

Es sei weiter $\boldsymbol{K}$ die Systemmatrix des linearen Operators mit variabler Grenze

$$\mathcal{K}_{\text{V}}[f](x) = \int_x^{x_{\max}} \kappa(x,y)\,f(y)\,dy = \sum_{l=1}^{k_{\text{sep}}} \int_x^{x_{\max}} \Phi_l(x)\Psi_l(y)\,f(y)\,dy, \tag{3.38}$$

definiert nach Vorschrift (3.23). Mit Voraussetzung 3.5.8 bzw. nach Bemerkung 3.4.5 stellt $\boldsymbol{K}$ eine obere rechte Dreiecksmatrix dar:

$$\boldsymbol{K} = \begin{pmatrix} K_{11} & K_{12} & \ldots & K_{1n} \\ 0 & K_{22} & \ddots & \vdots \\ \vdots & \ddots & \ddots & K_{(n-1)n} \\ 0 & \ldots & 0 & K_{nn} \end{pmatrix}. \tag{3.39}$$

Dabei entsprechen die Diagonalelemente K_{ii}, $i = 1, ..., n$ gerade den Volterra-Einträgen aus Definition 3.4.3 und die anderen von Null verschiedenen Elemente sind Fredholm-Einträge. Wir können diese Einträge trennen, indem wir das System (3.10) mit Hilfe der Matrix $\tilde{\boldsymbol{K}}$ und des Vektors $\boldsymbol{v}$ darstellen gemäß:

$$\tilde{\boldsymbol{g}} := \boldsymbol{M}\boldsymbol{g} = \boldsymbol{K}\boldsymbol{f} = \underbrace{\begin{pmatrix} 0 & K_{12} & \ldots & K_{1n} \\ 0 & 0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & K_{(n-1)n} \\ 0 & \ldots & 0 & 0 \end{pmatrix}}_{=:\,\tilde{\boldsymbol{K}}} \boldsymbol{f} + \underbrace{\begin{pmatrix} K_{11} \\ K_{22} \\ \vdots \\ K_{nn} \end{pmatrix}}_{=:\,\boldsymbol{v}} \circ \boldsymbol{f}. \tag{3.40}$$

Da gemäß Bemerkung 3.4.4 die Fredholm-Einträge von $\boldsymbol{K}$ mit den entsprechenden Einträgen von $\boldsymbol{N}$ übereinstimmen, gilt

$$K_{ij} = N_{ij} = \sum_{l=1}^{k_{\text{sep}}} S_{il}\,T_{lj} \quad \text{für} \quad i < j. \tag{3.41}$$

Um das Produkt $\tilde{\boldsymbol{K}}\boldsymbol{f} =: \tilde{\tilde{\boldsymbol{g}}} = [\tilde{\tilde{g}}_1, ..., \tilde{\tilde{g}}_n]^\top$ aus dem System (3.40) zu berechnen, benutzen wir die Darstellung (3.41) der K_{ij} für $i < j$. Der Vektor $\tilde{\tilde{\boldsymbol{g}}}$ kann dann in der folgenden Form geschrieben werden

$$\begin{pmatrix} \tilde{\tilde{g}}_1 \\ \vdots \\ \tilde{\tilde{g}}_{n-3} \\ \tilde{\tilde{g}}_{n-2} \\ \tilde{\tilde{g}}_{n-1} \\ \tilde{\tilde{g}}_n \end{pmatrix} = \sum_{l=1}^{k_{\text{sep}}} \begin{pmatrix} S_{1\,l} & [T_{l\,2}f_2 + \ldots + T_{l\,n-2}f_{n-2} + T_{l\,n-1}f_{n-1} + T_{l\,n}f_n] \\ \vdots & \vdots \quad\quad \vdots \quad\quad \vdots \\ S_{(n-3)l} & [\quad\quad T_{l\,n-2}f_{n-2} + T_{l\,n-1}f_{n-1} + T_{l\,n}f_n] \\ S_{(n-2)l} & [\quad\quad\quad\quad T_{l\,n-1}f_{n-1} + T_{l\,n}f_n] \\ S_{(n-1)l} & [\quad\quad\quad\quad\quad\quad T_{l\,n}f_n] \\ S_{n\,l} & [\quad\quad\quad\quad\quad\quad\quad\quad 0\ ] \end{pmatrix}.$$

Hier ist zu bemerken, dass wir Einträge z.T. mehrfach berechnen. So muss z.B. das Produkt $T_{l\,n} f_n$ für den 1. bis $(n-1)$-ten Eintrag berechnet werden, das Produkt $T_{l\,n-1} f_{n-1}$ für den 1. bis $(n-2)$-ten und so fort. Nachfolgend präsentieren wir einen Algorithmus zur Bestimmung von $\tilde{\boldsymbol{g}}$ aus (3.40), bei dem diese Mehrfachberechnung vermieden wird.

Algorithmus 3.5.9: *Seien die Matrizen $\boldsymbol{K}$ bzw. $\tilde{\boldsymbol{K}}$ sowie die Vektoren $\boldsymbol{f}$ und $\boldsymbol{v}$ gemäß (3.40) gegeben. Weiter seien $\boldsymbol{S}$ und $\boldsymbol{T}$ gemäß (3.37) definiert. Wir berechnen zunächst $\tilde{\boldsymbol{K}}\boldsymbol{f} = \tilde{\tilde{\boldsymbol{g}}} = [\tilde{\tilde{g}}_1, ..., \tilde{\tilde{g}}_n]^\top$ gestaffelt, wobei wir zu Beginn $\tilde{\tilde{\boldsymbol{g}}} = \boldsymbol{0}$ setzen.*

```
procedure StaggeredMultiplication(f, var g̃̃, S, T);
begin
  g̃_n := 0.0;
  for l := 1 to k_sep do
  begin
    j := n;
    sum := 0.0;
    for i := n − 1 downto 1 do
    begin
      while ( j > i ) do
      begin
        sum := sum + T_lj * f_j;
        j := j − 1;
      end;
      g̃̃_i := g̃̃_i + S_il * sum;
    end;
  end;
end;
```

*Zur vollständigen Berechnung von $\boldsymbol{K}\boldsymbol{f} =: \tilde{\boldsymbol{g}} = [\tilde{g}_1, \dots, \tilde{g}_n]^\top$ müssen zu $\tilde{\tilde{\boldsymbol{g}}}$ noch die Volterra-Anteile aus dem Vektor $\boldsymbol{v} = [K_{11}, \dots, K_{nn}]^\top$ addiert werden. Dies geschieht durch die Prozedur **AddVolterraEntries**, welche wir mit $\tilde{\boldsymbol{g}} = \tilde{\tilde{\boldsymbol{g}}}$ starten:*

```
procedure AddVolterraEntries(f, var g̃, v);
begin
  for i := 1 to n do
    g̃_i := g̃_i + K_ii * f_i;
end;
```

Folgerung 3.5.10: *Im Algorithmus 3.5.9 durchlaufen wir in der Prozedur **StaggeredMultiplication** die innere* `for`*-Schleife $(n-1)$-mal. Dabei werden innerhalb der* `while`*-Schleife höchstens n Additionen und n Multiplikationen[1)] ausgeführt und außerhalb noch einmal jeweils $n-1$ Additionen und Multiplikationen. Somit erfordert die Prozedur zusammen mit der äußeren Schleife, $k_{\text{sep}}(2(n-1)+2n) \leq 4\,k_{\text{sep}} n$ Operationen. Das Addieren der Volterra-Anteile erfordert zusätzlich $2\,n$ Operationen. Dabei*

[1)] Als arithmetische Operation zählt man gewöhnlich nur die reinen *Gleitkommaoperationen*, d.h. insbesondere keine Operationen bzgl. der Laufvariablen.

brauchen nur die Matrizen $\boldsymbol{S}$ und $\boldsymbol{T}$, sowie der Vektor $\boldsymbol{v}$ gespeichert zu werden. Somit haben wir einen Speicheraufwand von $2\,k_{\text{sep}}n + n$.

Wenn wir also eine separable Kernfunktion zugrunde legen, lässt sich die Systemmatrix $\boldsymbol{K}$ des linearen Volterra-Operators, dem das Galerkin-Verfahren und stückweise konstanten Basisfunktionen zugrunde liegen, mithilfe von Algorithmus 3.5.9 mit einen Aufwand von $\mathcal{O}(k_{\text{sep}}n)$ mit dem Koeffizientenvektor $\boldsymbol{f}$ multiplizieren. Der Speicheraufwand beträgt dabei ebenfalls $\mathcal{O}(k_{\text{sep}}n)$. Somit haben wir für festes k_{sep} optimale Komplexität erreicht.

Wir wollen nun das für den linearen Volterra-Operator vorgeschlagene Konzept verallgemeinern, wobei wir das Galerkin-Verfahren zugrunde legen und uns auf Volterra-Funktionen beschränken, die nachfolgenden Bedingungen genügen.

Voraussetzung 3.5.11: *Sind die Volterra-Funktionen $v_{\text{a}}, v_{\text{b}} \in C^0([0,1])$ gemäß (3.20) gegeben, so verlangen wir zusätzlich:*

a) Eine der Funktionen $v_{\text{a}}, v_{\text{b}}$ ist konstant.

b) Beide Funktionen $v_{\text{a}}, v_{\text{b}}$ sind monoton.

Bemerkung 3.5.12: Die Bedingungen von Voraussetzung 3.5.8 und 3.5.11 a) erlauben eine Darstellung der Galerkin-Matrix $\boldsymbol{K} \in \mathbb{R}^{n\times n}$, in der die Fredholm- und die Volterra-Einträge „fortlaufend“ sind. Die Forderung, dass v_{a} oder v_{b} konstant sind, stellt zusammen mit der Forderung nach geordneten Basen sicher, dass die von Null verschiedenen Einträge von $\boldsymbol{K}$ stets entweder am linken oder am rechten Rand der Matrix beginnen (linkes oder rechtes System). Insbesondere existiert für jede Zeile i ein Index $\mu(i) \geq 0$ und $\nu(i) \geq 0$, so dass für ein linkes bzw. rechtes System für $i, j \in \{1, \dots, n\}$ gilt:

a) $K_{ij} \in F(\boldsymbol{K})$ falls $j \leq \mu(i)$ bzw. $j \geq \mu(i)$ und $K_{ij} \notin F(\boldsymbol{K})$ sonst,

b) $K_{ij} \in V(\boldsymbol{K})$ falls $\mu(i) < j \leq \nu(i)$ bzw. $\mu(i) > j \geq \nu(i)$ und $K_{ij} \notin V(\boldsymbol{K})$ sonst.

Für den Fall, dass in einer Zeile keine Fredholm- bzw. Volterra-Einträge auftreten, setzen wir $\mu = 0$ bzw. $\nu = 0$.

Bemerkung 3.5.13: Fordern wir in Bemerkung 3.5.12 zusätzlich noch die Monotonie der Volterra-Funktionen gemäß Voraussetzung 3.5.11 b), so können wir vier Fälle unterscheiden

a) v_{a} konstant und v_{b} monoton wachsend $\Rightarrow \mu(i) \leq \mu(i+1)$ (linkes unteres System),

b) v_{a} konstant und v_{b} monoton fallend $\Rightarrow \mu(i) \geq \mu(i+1)$ (linkes oberes System),

c) v_{a} monoton wachsend und v_{b} konstant $\Rightarrow \mu(i) \geq \mu(i+1)$ (rechtes oberes System),

d) v_{a} monoton fallend und v_{b} konstant $\Rightarrow \mu(i) \leq \mu(i+1)$ (rechtes unteres System),

mit $i = 1, \dots, n-1$. In Abbildung 3.3 haben wir zu jeder Konstellation ein Beispiel skizziert.

Den von i abhängigen Index μ aus Bemerkung 3.5.13 kann man als diskrete Funktion auffassen. Dies motiviert die folgende Definition.

Definition 3.5.14: Es sei $\boldsymbol{K} \in \mathbb{R}^{n\times n}$ eine Galerkin-Systemmatrix.

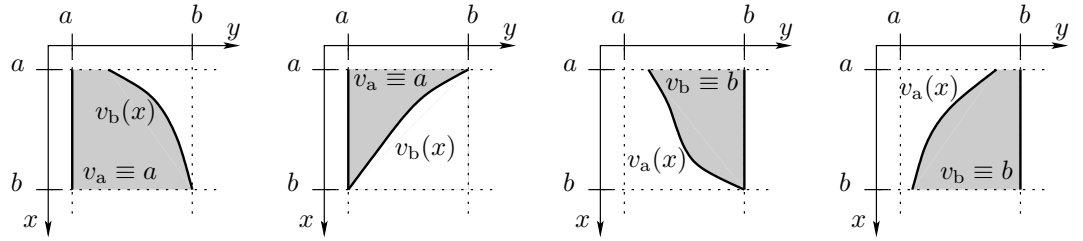
a) Wir definieren die diskrete Funktion

$$v_{\mathrm{vt}} : \{1, \dots, n\} \to \{0, \dots, n\},$$

welche für jede Zeile von $\boldsymbol{K}$ die Anzahl der Fredholm-Einträge angibt und bezeichnen v_{vt} als *Zeilenfunktion von* $\boldsymbol{K}$.

b) Es bezeichne $C_{\mathrm{vt}i} \in \mathbb{N}_0, C_{\mathrm{vt}i} \leq n$ die Anzahl der Volterra-Einträge der i-ten Zeile von $\boldsymbol{K}$, dann definieren wir die *Volterra-Konstante* C_{vt} von $\boldsymbol{K}$ gemäß

$$C_{\mathrm{vt}} := \max_{i=1,\dots,n} C_{\mathrm{vt}i}.$$



$\int_a^b \int_{v_{\mathrm{a}}\equiv a}^{v_{\mathrm{b}}(x)} dy\,dx$ $\quad$ $\int_a^b \int_{v_{\mathrm{a}}\equiv a}^{v_{\mathrm{b}}(x)} dy\,dx$ $\quad$ $\int_a^b \int_{v_{\mathrm{a}}(x)}^{v_{\mathrm{b}}\equiv b} dy\,dx$ $\quad$ $\int_a^b \int_{v_{\mathrm{a}}(x)}^{v_{\mathrm{b}}\equiv b} dy\,dx$

$$\begin{pmatrix} K_{11}\hat{K}_{12} & 0 & 0 & 0 \\ K_{21}K_{22}\hat{K}_{23} & 0 & 0 \\ K_{31}K_{32}K_{33}\hat{K}_{34} & 0 \\ K_{41}K_{42}K_{43}\hat{K}_{44} & 0 \\ K_{51}K_{52}K_{53}\hat{K}_{54} & 0 \end{pmatrix} \quad \begin{pmatrix} K_{11}K_{12}K_{13}\hat{K}_{14}\hat{K}_{15} \\ K_{21}K_{22}K_{23}\hat{K}_{24}\hat{K}_{25} \\ K_{31}\hat{K}_{32}\hat{K}_{33} & 0 & 0 \\ \hat{K}_{41}\hat{K}_{42} & 0 & 0 & 0 \\ \hat{K}_{51}\hat{K}_{52} & 0 & 0 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & \hat{K}_{12}K_{13}K_{14}K_{15} \\ 0 & \hat{K}_{22}K_{23}K_{24}K_{25} \\ 0 & 0 & \hat{K}_{33}K_{34}K_{35} \\ 0 & 0 & 0 & \hat{K}_{44}K_{45} \\ 0 & 0 & 0 & \hat{K}_{54}\hat{K}_{55} \end{pmatrix} \quad \begin{pmatrix} 0 & 0 & 0 & \hat{K}_{14}K_{15} \\ 0 & 0 & \hat{K}_{23}K_{24}K_{25} \\ 0 & \hat{K}_{32}K_{33}K_{34}K_{35} \\ 0 & \hat{K}_{42}K_{43}K_{44}K_{45} \\ \hat{K}_{51}\hat{K}_{52}K_{53}K_{54}K_{55} \end{pmatrix}$$

a) linkes unteres System $\quad$ b) linkes oberes System $\quad$ c) rechtes oberes System $\quad$ d) rechtes unteres System

Abbildung 3.3.: Grenzen und Matrizen.

Bemerkung 3.5.15: Sei die Zeilenfunktion v_{vt} aus Definition 3.5.14 gegeben.

a) Legen wir die Bedingungen aus den Voraussetzungen 3.5.8 und 3.5.11 zugrunde, so ist v_{vt} natürlich monoton.

b) Wir können $v_{\mathrm{vt}} : \{1, \dots, n\} \to \{0, \dots, n\}$ auch mithilfe des Vektors $\boldsymbol{v}_{\mathrm{vt}}$ realisieren, dessen Einträge $v_i \in \{0, \dots, n\}$ gerade gegeben sind durch:

$$v_i := v_{\mathrm{vt}}(i), \quad \text{für} \quad i = 1, \dots, n.$$

Beispiel 3.5.16: Für die Beispiele aus Abbildung 3.3 haben wir jeweils die folgenden Vektoren $\boldsymbol{v}_{\mathrm{vt}}$ sowie Konstanten C_{vt} vorliegen:

a) $\boldsymbol{v}_{\mathrm{vt}} = (1,2,3,3,3)^\top$ und $C_{\mathrm{vt}} = 1$

b) $\boldsymbol{v}_{\mathrm{vt}} = (3,3,1,0,0)^\top$ und $C_{\mathrm{vt}} = 2$

c) $\boldsymbol{v}_{\mathrm{vt}} = (3,3,2,1,0)^\top$ und $C_{\mathrm{vt}} = 2$

d) $\boldsymbol{v}_{\mathrm{vt}} = (1,2,3,3,3)^\top$ und $C_{\mathrm{vt}} = 2$.

Bemerkung 3.5.17: Sei $\boldsymbol{K}$ eine Volterra-Systemmatrix mit Einträgen $K_{ij} \in \mathbb{K}$ mit $i,j = 1,\dots,n$.

a) Wir können $\boldsymbol{K}$ stets in Matrizen $\tilde{\boldsymbol{K}}$ und $\boldsymbol{V}$ zerlegen, mit Einträgen $\tilde{K}_{ij}, V_{ij}$ gemäß

$$\tilde{K}_{ij} = \begin{cases} K_{ij} & \text{für} \quad K_{ij} \in F(\boldsymbol{K}) \\ 0 & \text{sonst} \end{cases} \text{ und } V_{ij} = \begin{cases} K_{ij} & \text{für} \quad K_{ij} \in V(\boldsymbol{K}) \\ 0 & \text{sonst,} \end{cases}$$

so dass gilt $\boldsymbol{K} = \tilde{\boldsymbol{K}} + \boldsymbol{V}$, wobei $\tilde{\boldsymbol{K}}$ alle Fredholm- und $\boldsymbol{V}$ alle Volterra-Einträge von $\boldsymbol{K}$ enthält, vgl. auch (3.40).

b) Mit Voraussetzung 3.5.8 sind die Einträge von $\boldsymbol{K}$ und somit auch von $\tilde{\boldsymbol{K}}$ und $\boldsymbol{V}$ zusammenhängend.

c) Ist das Fredholm-Pendant von $\boldsymbol{K}$ eine R(k)-Matrix mit entsprechenden Faktoren $\boldsymbol{S}$ und $\boldsymbol{T}$, so lassen sich die von Null verschiedenen Einträge von $\tilde{\boldsymbol{K}}$, gemäß Bemerkung 3.4.4 a), in der Form schreiben $K_{ij} = \sum_{l=1}^{k} S_{il} T_{lj}$.

Beispiel 3.5.18: Für das Beispiel c) aus Abbildung 3.3 haben die Matrizen $\tilde{\boldsymbol{K}}$ bzw. $\boldsymbol{V}$ nach Bemerkung 3.5.17 die Form

$$\begin{pmatrix} 0 & 0 & K_{13} & K_{14} & K_{15} \\ 0 & 0 & K_{23} & K_{24} & K_{25} \\ 0 & 0 & 0 & K_{34} & K_{35} \\ 0 & 0 & 0 & 0 & K_{45} \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad \text{bzw.} \quad \begin{pmatrix} 0 & \hat{K}_{12} & 0 & 0 & 0 \\ 0 & \hat{K}_{22} & 0 & 0 & 0 \\ 0 & 0 & \hat{K}_{33} & 0 & 0 \\ 0 & 0 & 0 & \hat{K}_{44} & 0 \\ 0 & 0 & 0 & \hat{K}_{54} & \hat{K}_{55} \end{pmatrix}.$$

Im Folgenden führen wir zunächst einen Matrizenkalkül ein und zeigen dann, dass wir die Volterra-Systemmatrizen mit der Hilfe ebendieses Kalküls sowie Bemerkung 3.5.17 darstellen können. Vorher führen wir noch nachstehende Definition ein, welche erheblich zur Vereinfachung in den Darstellungen beitragen wird.

Definition 3.5.19 (Indexfunktion): Wir definieren zwei diskrete Funktionen $I^{(L)}$ bzw. $I^{(R)}$ auf der Menge $\{1,\dots,n\}$ gemäß

$$I^{(L)}(i) = i \quad \text{bzw.} \quad I^{(R)}(i) = (n+1) - i,$$

für $i = 1,\dots,n$ und bezeichnen diese als *linke* bzw. *rechte Indexfunktion.*

Definition 3.5.20 (unvollständige R(k)-Matrix): Sei $\boldsymbol{N} = \boldsymbol{S}\boldsymbol{T} \in \mathbb{K}^{m\times n}$ eine R(k)-Matrix. Seien weiter $v : \{1,\dots,m\} \to \{0,\dots,n\}$ eine diskrete Funktion und $I^{(L)}$ bzw. $I^{(R)}$ die linke bzw. rechte Indexfunktion auf $\{1,\dots,n\}$. Dann definieren wir die Matrix $\boldsymbol{N}^{(L)}$ bzw. $\boldsymbol{N}^{(R)} \in \mathbb{K}^{m\times n}$ mit Einträgen

$$N_{ij}^{(L)} = \begin{cases} \sum_{l=1}^{k} S_{il}\, T_{lj} & \text{für} \quad I^{(L)}(j) \le v(i) \\ 0 & \text{sonst} \end{cases}$$

bzw.

$$N_{ij}^{(R)} = \begin{cases} \sum_{l=1}^{k} S_{il}\, T_{lj} & \text{für} \quad I^{(R)}(j) \le v(i) \\ 0 & \text{sonst,} \end{cases}$$

für $i = 1,\dots,m,\ j = 1,\dots,n$ und bezeichnen sie als *linke* bzw. *rechte unvollständige R(k)-Matrix*. Dabei sind jeweils S_{il} bzw. T_{lj} die Einträge der Matrizen $\boldsymbol{S}$ bzw. $\boldsymbol{T}$.

Definition 3.5.21 (*V*-Matrix): Sei $\boldsymbol{V} \in \mathbb{K}^{m\times n}$ eine schwachbesetzte Matrix, so dass in einer Zeile von $\boldsymbol{V}$ höchstens C_{v} zusammenhängende Einträge von Null verschieden sind, und sei weiter $v : \{1,\dots,m\} \to \{0,\dots,n\}$ eine diskrete Funktion. Wir bezeichnen $\boldsymbol{V}$ genau dann als *linke V-Matrix*, wenn $v(i)$ gerade der Anzahl der Nulleinträge *vor* dem ersten Nichtnull-Eintrag in der i-ten Zeile entspricht, $i = 1,\dots,m$. Analog dazu bezeichnen wir $\boldsymbol{V}$ genau dann als *rechte V-Matrix*, wenn $v(i)$ für $i = 1,\dots,m$ die Anzahl der Nulleinträge in der i-ten Zeile *nach* dem letzten Nichtnull-Eintrag angibt.

Folgerung 3.5.22: *Sei $\boldsymbol{V} \in \mathbb{K}^{m\times n}$ eine linke V-Matrix. Mit den Bezeichnungen aus Definition 3.5.21, können wir $\boldsymbol{V}$ in komprimierter Form durch eine Matrix $\boldsymbol{V}_{\mathrm{vt}}^{(L)} \in \mathbb{K}^{m\times C_{\mathrm{v}}}$ darstellen, mit Einträgen gemäß*

$$V_{il}^{(L)} = \begin{cases} V_{i\,(I^{(L)}(v(i)+l))} & \textit{für} \quad 1 \le I^{(L)}\,(v(i)+l) \le n \\ 0 & \textit{sonst,} \end{cases} \tag{3.42}$$

für $i = 1,\dots,m,\ l = 1,\dots,C_{\mathrm{v}}$, mit $I^{(L)}$ als linker Indexfunktion. Ist $\boldsymbol{V}$ ein rechte V-Matrix, können wir $\boldsymbol{V}$ auf analoge Weise durch eine Matrix $\boldsymbol{V}_{\mathrm{vt}}^{(R)} \in \mathbb{K}^{m\times C_{\mathrm{v}}}$ ausdrücken, deren Einträge $V_{il}^{(R)}$ wir dadurch erhalten, dass wir in (3.42) die Indexfunktion $I^{(L)}$ durch die rechte Indexfunktion $I^{(R)}$ ersetzen. Entsprechend können wir aus $\boldsymbol{V}_{\mathrm{vt}}^{(L)}$ bzw. $\boldsymbol{V}_{\mathrm{vt}}^{(R)}$ die ursprüngliche Matrix rekonstruieren, durch Anwendung der Vorschrift

$$V_{ij} = \begin{cases} V_{i\,(I^{(L)}(j)-v(i))}^{(L)} & \textit{für} \quad 0 < I^{(L)}(j) - v(i) \le C_{\mathrm{v}} \\ 0 & \textit{sonst,} \end{cases}$$

bzw.

$$V_{ij} = \begin{cases} V_{i\,(I^{(R)}(j)-v(i))}^{(R)} & \textit{für} \quad 0 < I^{(R)}(j) - v(i) \le C_{\mathrm{v}} \\ 0 & \textit{sonst,} \end{cases}$$

für $i = 1,\dots,m,\ j = 1,\dots,n$, wobei jeweils durch V_{ij} die Einträge von $\boldsymbol{V}$ bezeichnet seien.

Beispiel 3.5.23: Wenn wir wieder Abbildung 3.3 c) heranziehen, können wir die Matrix $\boldsymbol{V}$ aus Beispiel 3.5.18 als rechte V-Matrix auffassen, welche man in komprimierter Form durch die Matrix $\boldsymbol{V}_{\mathrm{vt}}^{(R)}$ ausdrücken kann, entsprechend

$$\boldsymbol{V} = \begin{pmatrix} 0 & \hat{K}_{12} & 0 & 0 & 0 \\ 0 & \hat{K}_{22} & 0 & 0 & 0 \\ 0 & 0 & \hat{K}_{33} & 0 & 0 \\ 0 & 0 & 0 & \hat{K}_{44} & 0 \\ 0 & 0 & 0 & \hat{K}_{54} & \hat{K}_{55} \end{pmatrix} \quad \Rightarrow \quad \boldsymbol{V}_{\mathrm{vt}}^{(R)} = \begin{pmatrix} \hat{K}_{12} & 0 \\ \hat{K}_{22} & 0 \\ \hat{K}_{33} & 0 \\ \hat{K}_{44} & 0 \\ \hat{K}_{55} & \hat{K}_{54} \end{pmatrix}.$$

Wenn wir die zugehörige Funktion v gemäß Bemerkung 3.5.15 b) als Vektor $\boldsymbol{v}$ auffassen, so gilt gerade $\boldsymbol{v} = (3, 3, 2, 1, 0)^{\top}$, vgl. Beispiel 3.5.16 c). Wenn wir $\boldsymbol{V}$ hingegen als linke V-Matrix auffassen, gilt $\boldsymbol{v} = (1, 1, 2, 3, 3)^{\top}$ und

$$\boldsymbol{V}_{\mathrm{vt}}^{(L)} = \begin{pmatrix} \hat{K}_{12} & 0 \\ \hat{K}_{22} & 0 \\ \hat{K}_{33} & 0 \\ \hat{K}_{44} & 0 \\ \hat{K}_{54} & \hat{K}_{55} \end{pmatrix}.$$

Definition 3.5.24 (VR(k)-Matrix): Es sei $\boldsymbol{N}^{(L)}$ eine linke unvollständige R(k)-Matrix, mit zugehöriger diskreter Funktion $v_L : \{1, \dots, m\} \to \{0, \dots, n\}$ sowie $\boldsymbol{V}$ eine linke V-Matrix mit zugehöriger diskreter Funktion $\tilde{v}_L : \{1, \dots, m\} \to \{0, \dots, n\}$. Dann nennen wir $\boldsymbol{N}^{(L)}$ und $\boldsymbol{V}$ *kompatibel*, falls $v_R \equiv \tilde{v}_R$. Sind $\boldsymbol{N}^{(L)}$ und $\boldsymbol{V}$ kompatibel, so definieren wir die Matrix $\boldsymbol{K}^{(L)} \in \mathbb{K}^{m \times n}$ gemäß

$$\boldsymbol{K}^{(L)} = \boldsymbol{N}^{(L)} + \boldsymbol{V}$$

und bezeichnen sie als *linke VR(k)-Matrix*. Analog dazu definieren wir die Matrix $\boldsymbol{K}^{(R)} \in \mathbb{K}^{m \times n}$ und bezeichnen sie entsprechend als *rechte VR(k)-Matrix*.

Folgerung 3.5.25 (Speicheraufwand): *Zur Speicherung einer linken bzw. rechten VR(k)-Matrix aus $\mathbb{K}^{m \times n}$ benötigen wir die $k(m+n)$ Einträge der zugrundeliegenden R(k)-Matrix[1)]. Für die V-Matrix brauchen wir nach Folgerung 3.5.22 nur $m\, C_{\mathrm{v}}$ Einträge zu speichern. Somit haben wir einen Gesamtaufwand von $k(m+n) + C_{\mathrm{v}}\, m$. Gilt überdies $k \geq C_{\mathrm{v}}$ bzw. $\mathcal{O}(k) = \mathcal{O}(C_{\mathrm{v}})$, dann haben wir, wie bei einer Niedrigrangmatrix, einen Speicheraufwand von $\mathcal{O}(k(m+n))$.*

Eine linke oder rechte VR(k)-Matrix kann man schnell mit einem Vektor multiplizieren, wenn die zugrundeliegende diskrete Funktion v monoton ist. Dann leistet der Algorithmus 3.5.27 das Gewünschte. Zunächst sei noch die folgende Notation gegeben.

[1)] Der Speicheraufwand der zugehörigen diskreten Funktion v, welche wir gemäß Bemerkung 3.5.15 durch einen Vektor $\boldsymbol{v} \in \mathbb{N}_0^n$ ausdrücken können, wird vernachlässigt, da wir hier keine Gleitkommazahlen vorliegen haben, vgl. auch Abschnitt 3.9.4.

Notation 3.5.26: Um Algorithmus 3.5.27 übersichtlicher darzustellen, führen wir eine allgemeinere `for` Schleife ein:

$$\text{„}\texttt{for } i := a \texttt{ (down)to } b\text{“} = \begin{cases} \text{„}\texttt{for } i := a \texttt{ to } b\text{“} & \text{für} \quad a \leq b \\ \text{„}\texttt{for } i := a \texttt{ downto } b\text{“} & \text{für} \quad a > b. \end{cases}$$

Algorithmus 3.5.27: *Sei $\boldsymbol{K} = \boldsymbol{N} + \boldsymbol{V} \in \mathbb{K}^{m \times n}$ eine linke bzw. rechte VR(k)-Matrix, welche wir mit einem Vektor $\boldsymbol{x} \in \mathbb{K}^n$ multiplizieren möchten. Dabei bezeichne $\boldsymbol{N} \in \mathbb{K}^{m \times n}$ eine linke bzw. rechte unvollständige R(k)-Matrix und $\boldsymbol{V} \in \mathbb{K}^{m \times C_v}$ eine zugehörige V-Matrix mit der Indexfunktion I, wobei wir voraussetzen, dass die zugrunde liegende diskrete Funktion $v : \{1, \dots, m\} \to \{0, \dots, n\}$ monoton ist. Der Geltungsbereich der Laufvariablen hängt dabei vom entsprechenden Monotonieverhalten ab, wir drücken ihn durch die Funktionen **start** und **stop** aus.*

```
integer function start();
begin
  if ( v monoton wachsend ) then return 1;
  else return m;                                  { v monoton fallend }
end;

integer function stop();
begin
  if ( v monoton wachsend ) then return m;
  else return 1;                                  { v monoton fallend }
end;
```

Zunächst bilden wir das Produkt $\boldsymbol{y} := \boldsymbol{N}\boldsymbol{x}$ mithilfe nachfolgender Prozedur, welche wir mit $\boldsymbol{y} = 0$ starten. Die Funktion v drücken wir dabei durch den Vektor $\boldsymbol{v}$ mit den Einträgen v_i aus, und wir benutzen Notation 3.5.26. Weiter seien mit S_{il} bzw. T_{lj} die Einträge der zu $\boldsymbol{N}$ gehörigen Matrizen $\boldsymbol{S}$ bzw. $\boldsymbol{T}$ bezeichnet.

```
procedure NMatrixMultiplication(N, x, var y);
begin
  for l := 1 to k do
  begin
    sum := 0.0;
    if ( linke VR(k)-Matrix ) then j := 1;
    else j := n;                                  { rechte VR(k)-Matrix }
    for i :=start() (down)to  stop() do
    begin
      while ( I(j) ≤ v_i ) do
      begin
        sum := sum + T_lj * x_j;
        if ( linke VR(k)-Matrix ) then j := j + 1;
        else j := j − 1;                          { rechte VR(k)-Matrix }
      end;
```

```
      y_i := y_i + S_il * sum;
    end;
  end;
```

Nun müssen wir zu $\boldsymbol{y}$ *noch das Produkt* $\boldsymbol{V}\boldsymbol{x}$ *addieren. Dazu benötigen wir nur die von Null verschiedenen Einträge, aus der zu* $\boldsymbol{V}$ *gehörigen komprimierten Matrix* $\boldsymbol{V}_{\mathrm{vt}} \in \mathbb{K}^{m \times C_{\mathrm{v}}}$*, deren Einträge mit* V_{ij} *bezeichnet seien. Der Aufruf nachstehender Prozedur* ***VMatrixMultiplication*** *multipliziert die entsprechenden Einträge aus* $\boldsymbol{V}$ *mit* $\boldsymbol{x}$ *und fügt sie zu* $\boldsymbol{y}$ *hinzu, wenn* $\boldsymbol{y}$ *zu Beginn das Resultat aus der Prozedur* ***NMatrixMultiplication*** *enthält.*

```
procedure VMatrixMultiplication(V_vt, x, var y);
begin
  for  i := 1 to m do
  begin
    sum := 0.0;
    for j := 1 to C_v do
      if ( 1 ≤ I(v_i + j) ≤ n ) then  sum := sum + V_ij * x_I(v_i+j)
    y_i := y_i + sum;
  end;
end;
```

Der Vektor $\boldsymbol{y}$ *enthält nun das vollständige Ergebnis der Matrix-Vektor-Multiplikation.*

Folgerung 3.5.28 (Rechenaufwand): *In Prozedur* ***NMatrixMultiplication*** *durchlaufen wir die innere* `for`*-Schleife* m*-mal und somit führen wir ohne die* `while`*-Schleife* m *Additionen und* m *Multiplikationen aus. In der* `while`*-Schleife werden jeweils höchstens* n *Additionen und Multiplikationen ausgeführt. Daher benötigen wir für die Prozedur unter Beachtung der äußeren* `for`*-Schleife, welche wir* k*-mal durchlaufen,* $2\,k(m+n)$ *Operationen, so wie bei einer R(k)-Matrix. Die Prozedur* ***VMatrixMultiplication*** *erfordert zusätzlich* $2\,C_{\mathrm{v}}m + m$ *Operationen. Somit haben wir einen Gesamtaufwand von* $2\,k(m+n) + m(2C_{\mathrm{v}} + 1)$ *arithmetischen Operationen. Gilt außerdem wieder* $k \geq C_{\mathrm{v}}$ *bzw.* $\mathcal{O}(k) = \mathcal{O}(C_{\mathrm{v}})$*, so beträgt der Aufwand* $\mathcal{O}(k(m+n))$*.*

Die Multiplikation mit einer transponierten VR(k)-Matrix realisieren wir analog zu Algorithmus 3.5.27, die Anzahl der Operationen besitzt die gleiche Komplexität.

Nun haben wir den angekündigten Matrizenkalkül vorliegen und können darangehen, mit seiner Hilfe eine Volterra-Systemmatrix darzustellen.

Folgerung 3.5.29: *Eine Volterra-Systemmatrix* $\boldsymbol{K}$ *lässt sich stets als linke bzw. rechte VR(k)-Matrix darstellen, wenn wir eine separable Kernfunktion zugrunde legen und die Voraussetzungen 3.5.8 und 3.5.11 erfüllt sind. Weiterhin lässt sich der Algorithmus 3.5.27 anwenden.*

Beweis: Aufgrund von Bemerkung 3.5.17 können wir $\boldsymbol{K}$ als Summe der beiden Matrizen $\tilde{\boldsymbol{K}}$ und $\boldsymbol{V}$ darstellen, deren Einträge wegen Voraussetzung 3.5.8 zusammenhängend sind. Die von Null verschiedenen Einträge von $\tilde{\boldsymbol{K}}$ entsprechen nach Bemerkung 3.5.17 c) gerade denen aus Definition 3.5.20. Ferner identifizieren wir die diskrete Funktion

v mit der Zeilenfunktion v_{vt} aus Definition 3.5.14. Somit ist $\tilde{\boldsymbol{K}}$ eine linke oder rechte unvollständige R(k)-Matrix ist, in Abhängigkeit davon, ob die Volterra-Funktion v_{a} oder v_{b} konstant ist (Voraussetzung 3.5.11 a)). Außerdem identifizieren wir die Konstante C_{v} mit der Volterra-Konstanten C_{vt} aus Definition 3.5.14 b). Damit ist nach dem Ebengesagten die Matrix $\boldsymbol{V}$ eine V-Matrix.

Da die Funktion v_{vt} nach Bemerkung 3.5.15 a) monoton ist, sind auch die Voraussetzungen für Algorithmus 3.5.27 gegeben. □

Somit können wir mithilfe von separablen Kernfunktionen ebenfalls Volterra-Systemmatrizen in effizienter Weise darstellen, mit einer Komplexität von $\mathcal{O}((k_{\mathrm{sep}} + C_{\mathrm{vt}})\, n)$.

Bemerkung 3.5.30: An dieser Stelle sei noch einmal explizit darauf hingewiesen, dass R(k)-Matrizen eine spezielle Art von VR(k)-Matrizen sind (wir setzen v und C_{v} identisch Null).

3.6. $\mathcal{H}$-Matrizen

Haben wir einen nichtseparablen Kern vorliegen, der zwar nicht glatt genug ist, um global durch einen separablen Kern approximiert zu werden, aber zumindest lokal diese Eigenschaft aufweist, so können wir den Kalkül der $\mathcal{H}$-Matrizen benutzen, siehe Hackbusch, Grasedyck [26, 22]. Dieser Kalkül wurde ursprünglich für singuläre Kerne entwickelt, welche z.B. bei *Randelementmethoden* bzw. *Boundary Element Methods* (BEM) auftreten und geht historisch aus dem *Panel-Clustering* hervor, siehe Hackbusch und Nowak [28].

Die bei Populationsphänomenen auftretenden Kerne weisen ebenfalls gewisse Singularitäten auf, und wir werden zeigen, dass der $\mathcal{H}$-Matrix-Kalkül mit einigen Modifikationen auch bei Populationskernen angewendet werden kann.

3.6.1. Konstruktion hierarchischer Bäume

Durch einen lokal glatten Kern wird i.A. keine Darstellung der kompletten Systemmatrix $\boldsymbol{K}$ als Niedrigrang-Matrix möglich sein. Sei im Folgenden I eine mit der Basis eines Finite-Elemente-Raumes korrespondierende Indexmenge, vgl. auch Bemerkung 3.3.1. Wir werden dann Teilmengen $\tau \times \sigma$ von $I \times I$ so bestimmen, dass zumindest auf dem Block $(K_{ij})_{i\in\tau, j\in\sigma}$ eine Niedrigrangdarstellung möglich ist. Damit ein zusammenhängender Matrixblock von $\boldsymbol{K}$ vorliegt, muss unter Umständen eine Umordnung der Basiselemente vorgenommen werden. Doch zunächst benötigen wir einige Begriffe:

Definition 3.6.1 (Clusterbaum): Sei $\mathcal{T}_I$ ein (binärer) Baum und bezeichne T_I die Menge seiner Knoten. Wir nennen $\mathcal{T}_I$ einen *Clusterbaum* bzgl. der Indexmenge I, falls folgendes gilt:

a) T_I ist Teilmenge der Potenzmenge von I.

b) I ist die Wurzel von $\mathcal{T}_I$.

c) Falls $\tau \in T_I$ ein Blatt ist, dann gilt $|\tau| \leq b_{\min}$, d.h. die Blätter bestehen aus einer relativ kleinen Anzahl von Indizes (zur Bedeutung von $b_{\min}$ siehe Bemerkung 3.6.16).

d) Falls $\tau \in T_I$ kein Blatt ist, so besitzt τ mindestens zwei Söhne und besteht aus deren disjunkter Vereinigung.

Beispiel 3.6.2: Für den in Abbildung 3.5 skizzierten Baum gilt gerade $I = \{1, 2, \dots, 8\}$ und $T_I = \{I, \{1,2,3,4\}, \{5,6,7,8\}, \{1,2\}, \{3,4\}, \{5,6\}, \{7,8\}, \{1\}, \{2\}, \dots, \{8\}\}$, dabei sind die Cluster $\{1\}, \dots, \{8\}$ die Blätter des Baumes.

Notation 3.6.3: Sei I eine Indexmenge und $\mathcal{T}_I$ der zugehörige Clusterbaum.

a) Die Knoten von $\mathcal{T}_I$ bezeichnen wir auch als *Cluster*.

b) Die Anzahl l der Vorfahren eines Clusters $\tau \in T_I$ sind eindeutig bestimmt, wir sagen auch τ befindet sich auf dem Level oder auch der Stufe l. Die Wurzel des Baumes besitzt keine Vorfahren und somit gilt für sie $l = 0$. Mit $T_{I,l}$ bezeichnen wir die Menge aller Cluster auf dem Level l und mit $l_{\max} := \max\{l \in \mathbb{N}_0 \,|\, T_{I,l} \neq \varnothing\}$ die Tiefe des Baumes.

c) Die Menge der Söhne von τ bezeichnen wir mit $\mathcal{S}(\tau)$.

d) Die Menge der Cluster ohne Söhne nennen wir Blätter von $\mathcal{T}_I$ und bezeichnen sie mit $\mathcal{L}(\mathcal{T}_I)$.

e) Die Menge aller Blätter von $\mathcal{T}_I$ im l-ten Level bezeichnen wir mit $\mathcal{L}_l(\mathcal{T}_I)$.

f) Mit $L_T := \{l \in \mathbb{N}_0 \,|\, \mathcal{L}_l(\mathcal{T}_I) \neq \varnothing\}$ bezeichnen wir die Menge der Stufen von $\mathcal{T}_I$, auf denen sich Blätter befinden.

Siehe hierzu auch Abbildung 3.4.

Definition 3.6.4 (Träger und Boundingboxen): Der *Träger* eines Clusters τ ist gegeben durch die Vereinigung der Träger, der mit den Elementen von τ korrespondierenden Basisfunktionen, d.h.

$$\operatorname{supp}(\tau) = \bigcup_{i \in \tau} \operatorname{supp}(b_i).$$

Weiter sei $B_\tau \subset \Omega \subset \mathbb{R}^d, d \in \mathbb{N}$, der kleinste achsenparallele Quader mit $\operatorname{supp}(\tau) \subset B_\tau$. Diesen bezeichnen wir als *die Boundingbox von* τ. Sie besitzt die Darstellung

$$B_\tau = J_\tau^1 \times \cdots \times J_\tau^d$$

mit abgeschlossenen Intervallen $J_\tau^j, j = 1, ..., d$.

Im $1d$-Fall sind die Boundingboxen bereits Intervalle, siehe auch Beispiel 3.3.2.

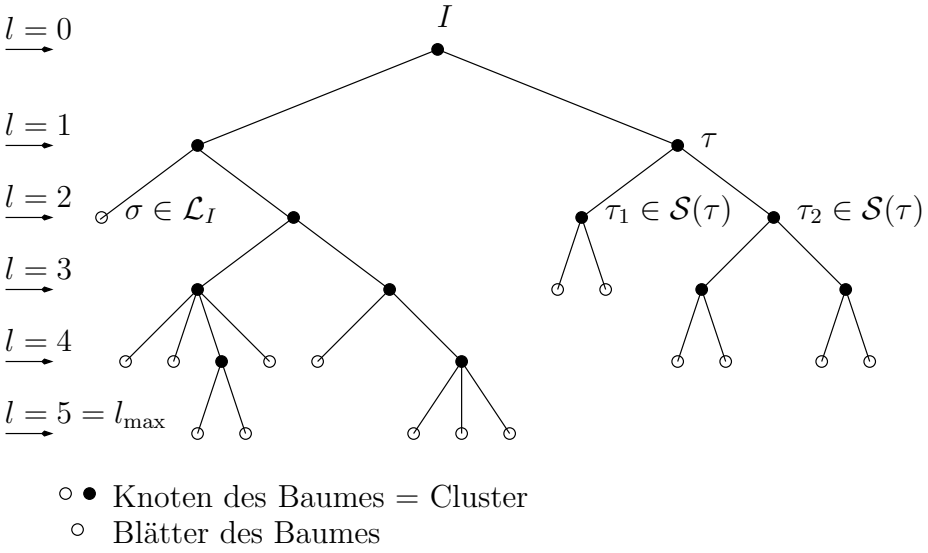


Abbildung 3.4.: Beispiel eines (nichtbinären) Clusterbaumes der Tiefe 5.

Definition 3.6.5: Der *Durchmesser* eines Clusters τ bzw. der Boundingbox von τ sei gegeben durch

$$\operatorname{diam}(\tau) := \max_{x,y \in \operatorname{supp}(\tau)} \|x - y\| \quad \text{bzw.} \quad \operatorname{diam}(B_\tau) := \max_{x,y \in B_\tau} \|x - y\|.$$

Der *Abstand* zweier Cluster τ und σ bzw. ihrer Boundingboxen sei definiert durch

$$\operatorname{dist}(\tau, \sigma) := \min_{x \in \operatorname{supp}(\tau), y \in \operatorname{supp}(\sigma)} \|x - y\| \quad \text{bzw.} \quad \operatorname{dist}(B_\tau, B_\sigma) := \min_{x \in B_\tau, y \in B_\sigma} \|x - y\|.$$

Dabei bezeichnet $\|.\|$ die Euklidische Norm auf $\mathbb{R}^d$, $d \in \mathbb{N}$.

Wir haben nun alle Werkzeuge, um den folgenden Algorithmus zur Konstruktion eines binären[1)] Clusterbaumes vorzustellen.

Algorithmus 3.6.6 (Konstruktion eines Clusterbaumes): *Wir assoziieren jede Basisfunktion bzw. jeden Index $i \in I$ mit einem passenden Punkt (Referenzpunkt) $p_i \in \Omega \subset \mathbb{R}^d$, siehe Bemerkung 3.3.4. Weiter sei $(e_j)_{j=1,\dots,d}$ die kanonische Orthonormalbasis des $\mathbb{R}^d$. Beginnend mit der Wurzel des Clusterbaumes, d.h. mit I, bestimmen wir in jedem Schritt für den vorgelegten Cluster die zugehörige Boundingbox und ihre längste Kante. Längs dieser Kante teilen wir die Box, so dass eines der beiden Kriterien erfüllt ist:*

a) Geometrisches Clustern: beide Teile sind etwa gleich groß,

b) Kardinalitätsbasiertes Clustern: in beiden Teilen liegen etwa gleichviele Referenzpunkte.

[1)] d.h. für alle $\tau \notin \mathcal{L}(I)$ gilt $|\mathcal{S}(\tau)| = 2$

Auf die so entstandenen Cluster wenden wir rekursiv solange die gleiche Prozedur an, bis die Anzahl der Indizes in den resultierenden Clustern den Parameter $b_{\min} \in \mathbb{N}$ *erreicht oder unterschreitet, siehe auch Bemerkung 3.6.16. Weiter bezeichnen* $J_\tau^j, j = 1, \dots, d$ *die zu einer Boundingbox gehörigen Intervalle aus Definition 3.6.4 und* $J_\tau^j(a)$ *bzw.* $J_\tau^j(b)$ *den Anfangs- bzw. Endpunkt von* J_τ^j*. Um eine Ordnung auf den Clustern einzuführen, bzw. zu gewährleisten, dass Cluster die aufeinanderfolgende Indizes besitzen, auch geometrisch benachbart sind, sei der Operator* $\boldsymbol{\pi}$ *definiert, welcher für einen Cluster* τ *die Permutationen* $\pi_1, \dots, \pi_d$ *so bestimmt, dass* $\langle p_{\pi_j(i)}, e_j \rangle \leq \langle p_{\pi_j(i+1)}, e_j \rangle$ *gilt, für* $j = 1, \dots, d$ *und* $i = 1, \dots, (|\tau| - 1)$*.*

```
procedure Split(τ, π(τ));
begin
  if (|τ| > b_min) then
  begin
    erzeuge Boundingbox B_τ = J_τ^1 × ··· × J_τ^d;
    j_max := argmax_{j∈{1,...,d}}{diam(J_τ^j)};     {Index der längsten Kante von B_τ}
    τ_1 := ∅; τ_2 := ∅;
    if (geometrisch clustern) then
    begin
      h := (J_τ^{j_max}(a) + J_τ^{j_max}(b))/2;     {teile Cluster längs der längsten Kante}
      for i ∈ τ do
        if (⟨p_i, e_{j_max}⟩ ≤ h) then τ_1 := τ_1 ∪ {i};
        else τ_2 := τ_2 ∪ {i};
    end;
    else                                            {kardinalitätsbasiert clustern}
    begin
      for 1 ≤ i ≤ |τ|/2 do
        τ_1 := τ_1 ∪ {π_{j_max}(i)};
      for |τ|/2 < i ≤ |τ| do
        τ_2 := τ_2 ∪ {π_{j_max}(i)};
    end;
    Split(τ_1, π(τ_1)); Split(τ_2, π(τ_2));
  end;
end;
```

Bemerkung 3.6.7: Die Frage, ob wir kardinalitätsbasiert oder geometrisch clustern sollten, werden wir später beantworten, siehe Bemerkung 3.6.17. Im ersten Fall werden wir stets einen ausgewogenen Baum bekommen, dessen Tiefe $\log(n)$ ist, wobei $|I| = n$. Um von der Wurzel des Baumes bis in alle Blätter zu gelangen, ist dann stets ein Aufwand von $\mathcal{O}(n \log(n))$ erforderlich. Ein geometrisch geclusterter Baum kann im schlimmsten Fall die Tiefe n haben, was für das Durchlaufen des Baumes für alle Blätter einen Aufwand von $\mathcal{O}(n^2)$ bedeuten würde.

Beispiel 3.6.8: Wir erstellen einen Beispielclusterbaum, auf den wir uns im Weiteren noch beziehen werden. Sei eine äquidistante Zerlegung des Intervalls $I = [0, 1]$ in $n = 2^{l_{\max}}$ Teilintervalle gegeben. Dann erhalten wir einen Clusterbaum, der für $l_{\max} = 3$ in

Abbildung 3.5 skizziert ist. Im l-ten Level befinden sich dann gerade 2^l Knoten bzw. Cluster, die wir mit $\tau_i^l, i = 1, ..., 2^l$ bezeichnen.

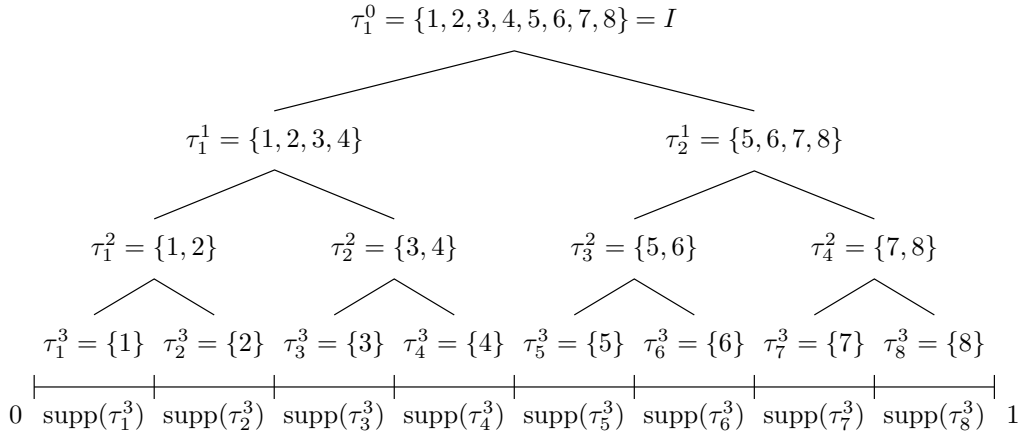


Abbildung 3.5.: Geometrisch balancierter Baum der Tiefe 3.

Wir wollen nun mit Hilfe der Clusterbäume zur Niedrigrangdarstellung geeignete Matrixblöcke auswählen. Dazu führen wir folgende Bezeichnung ein:

Notation 3.6.9 (Blockcluster): Seien τ und σ Cluster aus zwei Clusterbäumen, dann nennen wir die Menge $\tau \times \sigma$, einen *Blockcluster* oder auch einen *Block*.

Auf einem Blockcluster $\tau \times \sigma$ können wir nun eine Kernfunktion κ durch eine separable Funktion approximieren. Dies kann z.B. mithilfe des Interpolationsoperators aus Definition 3.5.6 geschehen. Im Abschnitt 3.5.2 haben wir mit (3.31) bereits eine Abschätzung für die Güte einer solchen Approximation kennengelernt. Insbesondere spielte die Beschränktheit der Ableitungen eine wichtige Rolle. Dem ursprünglichen Konzept der $\mathcal{H}$-Matrizen liegt daher auch eine spezielle Klasse von Kernfunktionen zugrunde, Kerne deren partielle Ableitungen *asymptotisch glatt* sind.

Definition 3.6.10 (Asymptotische Glattheit): Sei eine Kernfunktion κ auf dem Gebiet $\Omega \subset \mathbb{R}^d$ gegeben. Wir bezeichnen κ als *asymptotisch glatt zum Grad* $s \in \mathbb{N}_0$, falls es Konstanten $C_{\text{as1}}, C_{\text{as2}} > 0$ gibt, so dass gilt

$$|\partial_x^\alpha \partial_y^\beta \kappa(x,y)| \leq C_{\text{as1}}\, \alpha!\, \beta!\, (C_{\text{as2}} \|x-y\|)^{-|\alpha|-|\beta|-s}, \tag{3.43}$$

wobei $\alpha, \beta \in \mathbb{N}^d$ Multiindizes und $x, y \in \mathbb{R}^d$.

Kerne dieser Art besitzen gewöhnlich eine Singularität auf der Diagonalen. Man erkennt, dass die Abschätzung (3.43) beliebig schlecht wird, für $x \to y$. Als typischer Vertreter sei die Kernfunktion $\kappa(x,y) = \ln|x-y|$ genannt.

Lemma 3.6.11: *Sei κ eine asymptotisch glatte Kernfunktion zum Grad s auf dem Gebiet $\Omega \subset \mathbb{R}^d$. Weiter seinen τ und σ zwei Cluster, mit $B_\tau, B_\sigma \in \Omega$. Mit den Bezeichnungen aus Definition 3.6.10 setzen wir*

$$C_{\sigma,\tau} := \frac{C_{\text{as2}}\text{dist}(B_\tau, B_\sigma)}{\min(\text{diam}(B_\tau), \text{diam}(B_\sigma))} \quad \textit{und} \quad C_g := \frac{C_{\text{as1}}}{(C_{\text{as2}}\text{dist}(B_\tau, B_\sigma))^s}.$$

Interpoliert man κ auf $B_\tau \times B_\sigma$ mithilfe des Lagrangeschen Interpolationsoperators, so kann der Approximationsfehler abgeschätzt werden durch

$$\|\kappa - \Pi_L^k[\kappa]\|_{\infty, B_\tau \times B_\sigma} \leq 8\, e\, d\, C_g \Lambda_k^d\, (1 + C_{\sigma,\tau}^{-1})(k+1)\ (1 + 2C_{\sigma,\tau})^{-(k+1)}\,, \qquad (3.44)$$

wobei Λ_k die Lebesgue-Konstante bezeichnet, siehe Definition 3.7.32.

Beweis: Siehe etwa Börm und Grasedyck [5]. □

Der Interpolationsfehler in Lemma 3.6.11 hängt exponentiell von $C_{\sigma,\tau}$ ab. Der Fehler wird um so kleiner, je größer der Abstand zwischen den Boundingboxen B_τ und B_σ ist und je kleiner ihr Durchmesser. Diese Einflüsse von $\operatorname{dist}(B_\tau, B_\sigma)$, sowie von $\operatorname{diam}(B_\tau)$ und $\operatorname{diam}(B_\sigma)$ auf den Interpolationfehler, fasst man in folgendem Kriterium zusammen.

Definition 3.6.12 (Standard-Zulässigkeitsbedingung): Zwei Cluster τ und σ genügen der *Standardzulässigkeitsbedingung* $\mathcal{Z}_S(\eta)$, $\eta > 0$, falls

$$\min(\operatorname{diam}(B_\tau), \operatorname{diam}(B_\sigma)) \leq 2\eta \operatorname{dist}(B_\tau, B_\sigma).$$

Wenn der Bezug zu η eindeutig ist, schreiben wir mitunter auch nur $\mathcal{Z}_S$.

Notation 3.6.13 (Zulässigkeitsbedingung): Ein Kriterium mit dessen Hilfe man analog zur Standardzulässigkeitsbedingung $\mathcal{Z}_S$ entscheiden kann, ob auf den Blockcluster $\tau \times \sigma$ eine R(k)-Matrixdarstellung mit vertretbarem Aufwand möglich ist, bezeichnen wir allgemein als Zulässigkeitsbedingung $\mathcal{Z}$.

Mit Hilfe einer Zulässigkeitsbedingung $\mathcal{Z}$ können wir aus zwei Clusterbäumen $\mathcal{T}_I$ und $\mathcal{T}_{I'}$ Paare von Clustern auswählen, auf deren Träger eine Approximation durch eine separierte Kernfunktion sinnvoll ist.

Notation 3.6.14 (Blockclusterbaum): Mit $\mathcal{Z}$ konstruieren wir aus $\mathcal{T}_I$ und $\mathcal{T}_{I'}$ gemäß Algorithmus 3.6.15 einen Baum, dessen Knoten Blockcluster sind. Diesen Baum bezeichnen wir als *Blockclusterbaum* $\mathcal{T}_{I\times I'}$ bzgl. $\mathcal{Z}$.

Algorithmus 3.6.15 (Konstruktion eines Blockclusterbaums): *Seien I, I' zwei Indexmengen mit zugehörigen Clusterbäumen $\mathcal{T}_I$, $\mathcal{T}_{I'}$ und $\mathcal{Z}$ eine Zulässigkeitsbedingung. Mit nachfolgender Prozedur erzeugen wir einen zu $\mathcal{T}_I, \mathcal{T}_{I'}$ korrespondierenden Blockclusterbaum $\mathcal{T}_{I\times I'}$, wenn wir sie mit $\tau = I, \sigma = I'$ aufrufen, siehe dazu auch Abbildung 3.6 mit den Bezeichnungen aus Beispiel 3.6.8.*

```
procedure BuildBlockclusterTree(τ, σ);
begin
  Erzeuge Blockcluster τ × σ;
  if (τ × σ genügen nicht Z und S(τ) ≠ ∅ und S(σ) ≠ ∅) then
  begin
    S := {(τ', σ') | τ' ∈ S(τ), σ' ∈ S(σ)};
    for ((τ', σ') ∈ S) do
      BuildBlockclusterTree(τ', σ');
```

```
    end;
    else
      S(τ × σ) := ∅;
end;
```

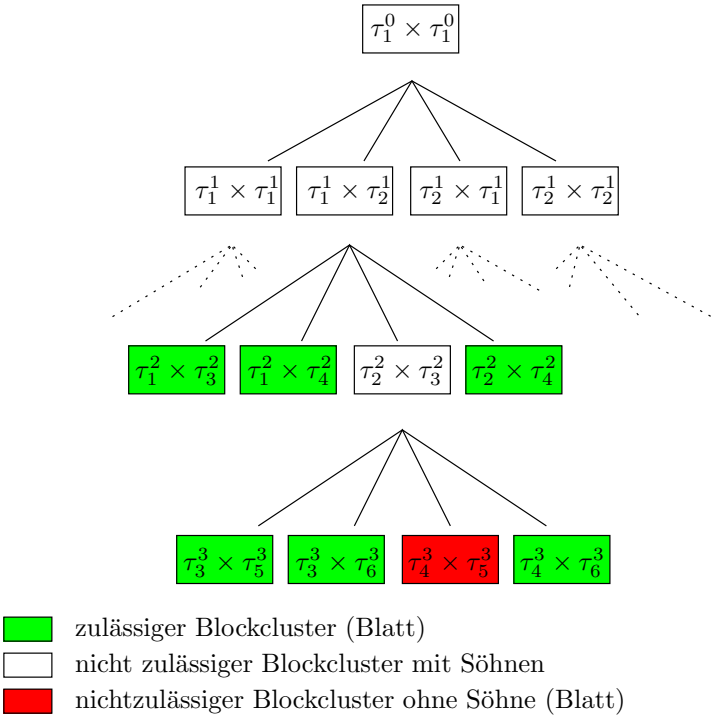


Abbildung 3.6.: Ausschnitt aus einem Blockclusterbaum bzgl. $\mathcal{Z}_S$.

Bemerkung 3.6.16 (Parameter $b_{\min}$): Theoretisch wäre in Algorithmus 3.6.6 eine Zerlegung denkbar, in der jedes Blatt genau einen Index enthält. Ab einer gewissen *Kleinheit* der Blöcke ist allerdings die Zerlegung in Niedrigrangmatrizen aufwendiger als der Einsatz einer vollbesetzten Matrix. Dies wird durch den Parameter $b_{\min}$ beschrieben, welcher insbesondere vom vorgegebenen Rang k abhängt. Empirisch hat sich $b_{\min} := 3k$ als eine gute Wahl herausgestellt. Eine ausführlichere Erläuterung zu $b_{\min}$ findet man in Grasedyck und Hackbusch [23].

Bemerkung 3.6.17: Die von uns betrachteten Auswahlkriterien sind rein geometrisch, vgl. auch Abschnitt 3.7. So hängt etwa $\mathcal{Z}_S$ für zwei Cluster τ und σ nur von der Größe und dem Abstand der zugehörigen Träger bzw. Boundingboxen von τ und σ ab, und ist so, von $b_{\min}$ abgesehen, unabhängig von der Anzahl der Indizes in τ und σ. Unsere Verfahren beruhen auf der Annahme, dass genügend viele zulässige Clusterpaare existieren. Wir werden daher geometrisch clustern, da wir auf diese Weise sicherstellen können, dass die Boundingboxen mit zunehmender Baumtiefe entsprechend verkleinert werden und somit ab einer gewissen Tiefe zulässig sind.

Gilt im Fall des geometrischen Clusterns für die Baumtiefe $l_{\max} = \mathcal{O}(\log(n))$, wie im kardinalitätsbasierten Fall, vgl. Bemerkung 3.6.7, sprechen wir von *geometrisch balancierten* Bäumen, vgl. auch Grasedyck [22]. In den von uns betrachteten Fällen haben wir geometrisch balancierte Bäume vorliegen.

3.6.2. Konstruktion von $\mathcal{H}$-Matrizen

Die Blätter des Blockclusterbaums bestehen aus zulässigen und nichtzulässigen Blöcken. Jeder dieser Blöcke entspricht einem Matrixblock aus der Systemmatrix. Auf jedem zulässigen Block kann die Systemmatrix lokal durch eine R(k)-Matrix approximiert werden, auf den nichtzulässigen Blöcken ist der Fehler einer solchen Approximation zu groß, und wir behalten den ursprünglichen vollbesetzten Matrixblock bei.

Definition 3.6.18 ($\mathcal{H}$-Matrix): Seien I, I' zwei Indexmengen und $\mathcal{T}_{I\times I'}$ ein Blockclusterbaum bzgl. der Zulässigkeitsbedingung $\mathcal{Z}$. Eine Matrix $\boldsymbol{M} \in \mathbb{R}^{I\times I'}$ heißt *$\mathcal{H}$-Matrix (hierarchische Matrix) mit blockweisem Rang k*, falls für jedes zulässige Blatt $b \in \mathcal{L}(\mathcal{T}_{I\times I'})$ der korrespondierende Matrixblock $M_b = (M_{ij})_{(i,j)\in b}$ eine R(k)-Matrix ist.

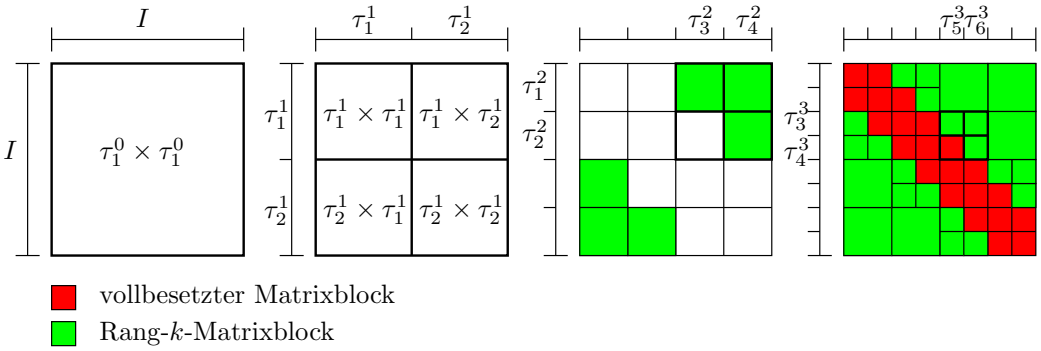


Abbildung 3.7.: Beispieldarstellung einer $\mathcal{H}$-Matrix.

In Abbildung 3.6 ist skizziert, wie eine $\mathcal{H}$-Matrix aus dem Blockclusterbaum hervorgeht (Bezeichnungen aus Beispiel 3.6.8).

Bemerkung 3.6.19: Der hier benutzte Zugang zu den $\mathcal{H}$-Matrizen über einen Blockclusterbaum ist der „klassische“. Hier bleiben die zugrundeliegenden Clusterbaumstrukturen erhalten und mit ihrer Hilfe können komplexere Matrixoperationen, wie z.B. Matrix-Matrix-Multiplikationen oder Matrix-Inversionen realisiert werden, siehe etwa Grasedyck [22]. Werden ausschließlich Matrix-Vektor-Multiplikationen durchgeführt, kann man als Alternative auch nur die Blätter des Blockclusterbaums speichern. Man hat somit grob gesagt nur eine unstrukturierte Menge von Blöcken vorliegen und man spricht in diesem Fall von einer *Blockpartition*. Die zugehörige $\mathcal{H}$-Matrix besitzt natürlich die selbe Form, aber die Informationen über die Baumstruktur werden nicht gespeichert. Der Verwaltungaufwand ist daher etwas geringer.

Algorithmus 3.6.20 (Matrix-Vektor-Multiplikation): *Sei* $\boldsymbol{M} \in \mathbb{R}^{I \times I'}$ *eine* $\mathcal{H}$*-Matrix. Um das Matrix-Vektor-Produkt* $\boldsymbol{y} := \boldsymbol{y} + \boldsymbol{M}\boldsymbol{x}$ *mit* $\boldsymbol{x} \in \mathbb{R}^I$ *und* $\boldsymbol{y} \in \mathbb{R}^{I'}$ *zu berechnen, benutzen wir folgende Prozedur, die eine Matrix-Vektor-Multiplikation auf jedem Blatt des zu* $\boldsymbol{M}$ *gehörigen Blockclusterbaums ausführt:*

```
procedure MVMultiplication(M, τ × σ, x, var y);
begin
  if (S(τ × σ) ≠ ∅) then
    for τ' × σ' ∈ S(τ × σ) do
      MVMultiplication(M, τ' × σ', x, var y);
  else
    y|_τ := y|_τ + M|_{τ×σ} x|_σ;
end;
```

Dabei starten wir die Prozedur mit den Indexmengen $I \times I'$*.*

3.6.3. Komplexitätsabschätzungen

Die Komplexität arithmetischer Operationen auf $\mathcal{H}$-Matrizen, in unserem Falle die Matrix-Vektor-Multiplikation, lassen sich relativ leicht auf den Speicheraufwand der Matrix zurückführen.

Bemerkung 3.6.21: Eine Zulässigkeitsbedingung kann man als Abbildung der Knoten eines Blockclusterbaums $\mathcal{T}_{I \times I'}$ auf die Menge $\{\textit{zulässig, nichtzulässig}\}$ auffassen:

$$\mathcal{Z} : \mathcal{T}_{I \times I'} \to \{\textit{zulässig, nichtzulässig}\}.$$

In diesem Sinne ist $\mathcal{Z}^{-1}$ in folgender Notation 3.6.22 zu verstehen.

Notation 3.6.22: Sei $\mathcal{T}_I$ ein Clusterbaum und $\mathcal{Z}$ eine zugehörige Zulässigkeitsbedingung, dann führen wir die folgenden Bezeichnungen ein:

$$\begin{aligned}
\mathcal{L}^+(\mathcal{T}) &:= \mathcal{Z}^{-1}(\textit{zulässig}) \cap \mathcal{L}(\mathcal{T}) \\
\mathcal{L}^-(\mathcal{T}) &:= \mathcal{Z}^{-1}(\textit{nichtzulässig}) \cap \mathcal{L}(\mathcal{T}) \\
\mathcal{L}_l^+(\mathcal{T}) &:= \mathcal{Z}^{-1}(\textit{zulässig}) \cap \mathcal{L}_l(\mathcal{T}), \quad l = 0, ..., l_{\max} \\
\mathcal{L}_l^-(\mathcal{T}) &:= \mathcal{Z}^{-1}(\textit{nichtzulässig}) \cap \mathcal{L}_l(\mathcal{T}), \quad l = 0, ..., l_{\max}
\end{aligned}$$

Notation 3.6.23: Wir bezeichnen den Aufwand, den eine Operation O in der Klasse X zu den Parametern $P_1, ...$ erfordert, mit $N_{X,O}(P_1, ...)$. Den Aufwand zur Speicherung einer $m \times n$-R(k)-Matrix bzw. einer vollbesetzten Matrix bezeichnen wir mit $N_{L,St}(m, n)$ bzw. $N_{F,St}(m, n)$[1)]. Nach Folgerung 3.5.4 a) gilt somit:

$$\begin{aligned}
&N_{L,St}(m, n) = k(m + n) \quad \text{bzw.} \\
&N_{F,St}(m, n) = mn.
\end{aligned}$$

[1)] dabei stehen F für *full*, L für *lowrank* und St für *storage*

Entsprechend gilt für die Komplexität $N_{L,Mv}$ bzw. $N_{F,Mv}$ der Matrix-Vektor-Multiplikation nach Folgerung 3.5.4 b), die Abschätzung:

$$\begin{aligned} k(m+n) \leq N_{L,Mv}(m,n) \leq 2k(m+n) \quad \text{bzw.} \\ mn \leq N_{F,Mv}(m,n) \leq 2mn. \end{aligned} \tag{3.45}$$

Definition 3.6.24: Wir definieren die *Schwachbesetztheitskonstante* C_{sp} eines Blockclusterbaums $\mathcal{T}_{I\times I'}$ durch

$$C_{\text{sp}} := \max\{C_{\text{sp1}}, C_{\text{sp2}}\},$$

wobei gilt

$$\begin{aligned} C_{\text{sp1}} &:= \max_{\tau\in T_I} \left|\{\sigma \in T_{I'} | \tau\times\sigma \in \mathcal{L}(I\times I')\}\right| \\ C_{\text{sp2}} &:= \max_{\sigma\in T_{I'}} \left|\{\tau \in T_I | \tau\times\sigma \in \mathcal{L}(I\times I')\}\right|. \end{aligned}$$

Einen Clusterbaum nennen wir *schwachbesetzt*, falls $C_{\text{sp}} = \mathcal{O}(1)$ gilt (d.h. nicht von I, I' abhängig ist).

Die Konstante C_{sp} ist ein Maß für die Schwachbesetztheit der Blockstruktur einer $\mathcal{H}$-Matrix, die aus der Menge $I \times I'$ gebildet wurde (siehe Abbildung 3.8).

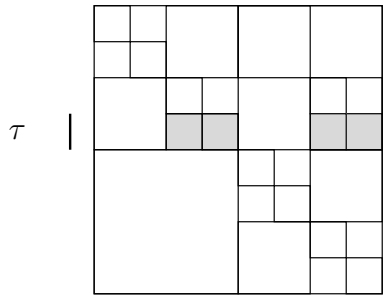


Abbildung 3.8.: Beispiel für $C_{\text{sp}} = 4$.

Lemma 3.6.25: *Für den Speicheraufwand $N_{\mathcal{H},St}(k,\mathcal{T},\mathcal{Z})$ einer $\mathcal{H}$-Matrix zum Blockclusterbaum $\mathcal{T} = \mathcal{T}_{I\times I'}$, der Zulässigkeitsbedingung $\mathcal{Z}$ und konstantem Rang k gilt:*

$$N_{\mathcal{H},St}(k,\mathcal{T},\mathcal{Z}) \leq |L_T|\, C_{\text{sp}} \max(k, b_{\min})(|I| + |I'|).$$

Beweis:

$$\begin{aligned} N_{\mathcal{H},St}(k,\mathcal{T},\mathcal{Z}) &= \sum_{\tau\times\sigma\in\mathcal{L}^+(\mathcal{T})} N_{L,St}(|\tau|,|\sigma|) \; + \sum_{\tau\times\sigma\in\mathcal{L}^-(\mathcal{T})} N_{F,St}(|\tau|,|\sigma|) \\ &\leq \sum_{l\in L_T} \sum_{\tau\times\sigma\in\mathcal{L}_l^+(\mathcal{T})} k|\tau| + \sum_{l\in L_T} \sum_{\tau\times\sigma\in\mathcal{L}_l^+(\mathcal{T})} k|\sigma| + \sum_{l\in L_T} \sum_{\tau\times\sigma\in\mathcal{L}_l^-(\mathcal{T})} |\tau||\sigma| \\ &\leq \sum_{l\in L_T} \sum_{\tau\times\sigma\in\mathcal{L}_l(\mathcal{T})} |\tau| \max(k, b_{\min}) + \sum_{l\in L_T} \sum_{\tau\times\sigma\in\mathcal{L}_l(\mathcal{T})} |\sigma| \max(k, b_{\min}) \\ &\leq |L_T|\, C_{\text{sp}} \max(k, b_{\min})(|I| + |I'|). \end{aligned}$$

□

Lemma 3.6.26: *Den Aufwand einer Matrix-Vektor-Multiplikation $N_{\mathcal{H},Mv}(k,\mathcal{T},\mathcal{Z})$ der $\mathcal{H}$-Matrix aus Lemma 3.6.25 können wir mithilfe des Speicheraufwandes abschätzen durch*

$$N_{\mathcal{H},St}(k,\mathcal{T},\mathcal{Z}) \leq N_{\mathcal{H},Mv}(k,\mathcal{T},\mathcal{Z}) \leq 2N_{\mathcal{H},St}(k,\mathcal{T},\mathcal{Z}).$$

Beweis: Nach (3.45) gilt:

$$\begin{aligned}
N_{\mathcal{H},Mv}(k,\mathcal{T},\mathcal{Z}) &= \sum_{\tau\times\sigma\in\mathcal{L}^+(\mathcal{T})} N_{L,Mv}(|\tau|,|\sigma|) \;+ \sum_{\tau\times\sigma\in\mathcal{L}^-(\mathcal{T})} N_{F,Mv}(|\tau|,|\sigma|) \\
&\geq \sum_{\tau\times\sigma\in\mathcal{L}^+(\mathcal{T})} N_{L,St}(|\tau|,|\sigma|) \;+ \sum_{\tau\times\sigma\in\mathcal{L}^-(\mathcal{T})} N_{F,St}(|\tau|,|\sigma|) \\
&= N_{\mathcal{H},St}(k,\mathcal{T},\mathcal{Z}) \\
N_{\mathcal{H},Mv}(k,\mathcal{T},\mathcal{Z}) &\leq \sum_{\tau\times\sigma\in\mathcal{L}^+(\mathcal{T})} 2N_{L,St}(|\tau|,|\sigma|) \;+ \sum_{\tau\times\sigma\in\mathcal{L}^-(\mathcal{T})} 2N_{F,St}(|\tau|,|\sigma|) \\
&= 2N_{\mathcal{H},St}(k,\mathcal{T},\mathcal{Z}).
\end{aligned}$$

□

Bemerkung 3.6.27: In die Komplexitätsbetrachtungen aus Lemma 3.6.26 geht nur der Aufwand für die reinen Multiplikationen ein, der Aufwand, welcher etwa durch das Suchen im Blockclusterbaum verursacht wird, bleibt unberücksichtigt.

3.6.4. $\mathcal{H}$-Matrizen und Volterra-Grenzen

Der ursprüngliche Kalkül sieht keine Volterra-Grenzen vor. Beim Aufbau eines Blockclusterbaums unterscheiden wir lediglich zulässige und nicht zulässige Blöcke, denen stets Fredholm-Operatoren zugrunde liegen. Aufgrund des Volterra-Charakters ist eine weitere Klassifizierung der Blöcke notwendig. In Anlehnung an Definition 3.4.3 aus Abschnitt 3.4.2 benutzen wir folgende Bezeichnungen.

Notation 3.6.28: Seien I, I' zwei Indexmengen und $\mathcal{T}_{I\times I'}$ ein Blockclusterbaum bzgl. der Zulässigkeitsbedingung $\mathcal{Z}$. Für jedes Blatt $b \in \mathcal{L}(\mathcal{T}_{I\times I'})$ können wir für den korrespondierenden Matrixblock $M_b = (M_{ij})_{(i,j)\in b}$ drei Fälle unterscheiden:

a) es gilt $M_{ij} = 0$ für alle $(i,j) \in b$,

b) M_{ij} besteht für $(i,j) \in b$ ausschließlich aus Fredholm-Einträgen

c) M_{ij} enthält für $(i,j) \in b$ mindestens einen Volterra-Eintrag.

Entsprechend werden die Matrixblöcke in den Fällen a), b) bzw. c) als *Null-*, *Fredholm-* bzw. *Volterra-Blöcke* bezeichnet.

Auf die Speicherung und Multiplikation der zu den Fredholm- bzw. Volterra-Blöcken gehörigen Matrizen, als R(k)- bzw. als VR(k)-Matrizen sind wir bereits in den Abschnitten 3.5.1 bzw. 3.5.3 eingegangen. Die zu einem Nullblock gehörige Matrix ist natürlich die Nullmatrix und somit benötigen wir für sie keinen Speicher und auch die Multiplikation mit einem Vektor ist trivial.

Beim Erzeugen eines Blockclusterbaums ist eine Modifizierung sinnvoll. Bei zwei Clustern, denen ein Nullblock zugrunde liegt, kann auf die Zulässigkeitsprüfung verzichtet werden. In diesem Fall besitzt die zugehörige Matrix bereits die denkbar einfachste Form und damit ist die Kernentwicklung überflüssig. Wir können diese Modifizierung mithilfe der Zulässigkeitsbedingung steuern.

Definition 3.6.29 (Erweiterte Zulässigkeitsbedingung): Es seien Volterra-Funktionen v_a und v_b und eine Zulässigkeitsbedingung $\mathcal{Z}$ gegeben. Zwei Cluster τ und σ heißen zulässig bzgl. der *erweiterten Zulässigkeitsbedingung* $\mathcal{E}(\mathcal{Z})$, wenn sie $\mathcal{Z}$ genügen oder der Blockcluster $\tau \times \sigma$ in Abhängigkeit von v_a und v_b ein Nullblock ist. Die Funktionen v_a und v_b bezeichnen wir auch als *zu* $\mathcal{E}(\mathcal{Z})$ *gehörige* Volterra-Funktionen.

Im Algorithmus 3.6.15 zur Erzeugung eines Blockclusterbaumes brauchen wir also nur die Zulässigkeitsbedingung durch die erweiterte zu ersetzen, um den Veränderungen durch die variablen Integralgrenzen Rechnung zu tragen. Entsprechend passen wir die Definition einer $\mathcal{H}$-Matrix an.

Definition 3.6.30 (Erweiterte $\mathcal{H}$-Matrix): Seien I, I' zwei Indexmengen und $\mathcal{T}_{I\times I'}$ ein Blockclusterbaum bzgl. der Zulässigkeitsbedingung $\mathcal{E}(\mathcal{Z})$. Eine Matrix $\boldsymbol{M} \in \mathbb{R}^{I\times I'}$ heißt *erweiterte $\mathcal{H}$-Matrix mit blockweisem Rang k*, falls für jedes zulässige Blatt $b \in \mathcal{L}(\mathcal{T}_{I\times I'})$ der korrespondierende Matrixblock $M_b = (M_{ij})_{(i,j)\in b}$ eine R(k)-Matrix, eine VR(k)-Matrix oder ein Nullblock ist. Den nichtzulässigen Blöcken entsprechen vollbesetzte Matrizen. Erweiterte $\mathcal{H}$-Matrizen bezeichnen wir auch als $\mathcal{H}_V$-Matrizen.

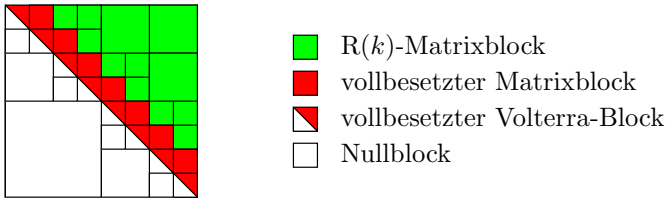


Abbildung 3.9.: $\mathcal{H}_V$-Matrix mit R(k)-Matrizen.

Beispiel 3.6.31: In Abbildung 3.9 haben wir eine $\mathcal{H}_V$-Matrix skizziert, wobei wir die Zulässigkeitsbedingung $\mathcal{E}(\mathcal{Z}_S)$ zugrunde gelegt haben und $v_\mathrm{a}(x) = x$ und $v_\mathrm{b}(x) \equiv 1$, für $x \in [0,1]$ die zu $\mathcal{E}(\mathcal{Z}_S)$ gehörigen Volterra-Grenzen sind.

Der Algorithmus 3.6.20 zur Matrix-Vektor-Multiplikation einer $\mathcal{H}$-Matrix kann für die erweiterten Matrizen beibehalten werden.

In der Nomenklatur von Notation 3.6.23 gilt für die Komplexität einer VR(k)-Matrix gemäß Folgerung 3.5.25 bzw. Folgerung 3.5.28

$$N_{V,St}(m,n) = k(m+n) + m\,C_{\mathrm{v}} \qquad \text{bzw.}$$
$$N_{V,Mv}(m,n) = 2\,k(m+n) + m(2\,C_{\mathrm{v}}+1)$$

und somit

$$N_{V,St}(m,n) \leq N_{V,Mv}(m,n) \leq 2N_{V,St}(m,n). \tag{3.46}$$

Damit können wir die Komplexität einer erweiterten $\mathcal{H}$-Matrix abschätzen.

Lemma 3.6.32: *Für den Speicheraufwand $N_{\mathcal{H}_V,St}(k,\mathcal{T},\mathcal{Z})$ einer $\mathcal{H}_V$-Matrix bzgl. des Clusterbaums $\mathcal{T} = \mathcal{T}_{I\times I'}$ und konstantem Rang k gilt:*

$$N_{\mathcal{H}_V,St}(m,n) \leq |L_T|\,C_{\mathrm{sp}} \max(C_{\mathrm{v}},k,b_{\min})\,(2|I|+|I'|).$$

Beweis: Wir führen den Beweis analog dem von Lemma 3.6.25, dabei vernachlässigen wir die Nullblöcke und nehmen an, dass jeder zulässige Block eine VR(k)-Matrix ist. Außerdem benutzen wir Teile der Abschätzung des Beweises.

$$\begin{aligned}
N_{\mathcal{H}_V,St}(k,\mathcal{T},\mathcal{Z}) = &\sum_{\tau\times\sigma\in\mathcal{L}^+(\mathcal{T})} N_{V,St}(|\tau|,|\sigma|) \;+ \sum_{\tau\times\sigma\in\mathcal{L}^-(\mathcal{T})} N_{F,St}(|\tau|,|\sigma|)\\
\leq &\sum_{l\in L_T}\sum_{\tau\times\sigma\in\mathcal{L}_l^+(\mathcal{T})} k|\tau| + \sum_{l\in L_T}\sum_{\tau\times\sigma\in\mathcal{L}_l^+(\mathcal{T})} k|\sigma| + \sum_{l\in L_T}\sum_{\tau\times\sigma\in\mathcal{L}_l^+(\mathcal{T})} C_{\mathrm{v}}|\tau|\\
&+\sum_{l\in L_T}\sum_{\tau\times\sigma\in\mathcal{L}_l^-(\mathcal{T})} |\tau||\sigma|\\
\leq &\;|L_T|\,C_{\mathrm{sp}}\max(C_{\mathrm{v}},k,b_{\min})\,(2|I|+|I'|).
\end{aligned}$$

□

Lemma 3.6.33: *Für den Aufwand der Matrix-Vektor-Multiplikation $N_{\mathcal{H}_V,Mv}(k,\mathcal{T},\mathcal{Z})$ einer $\mathcal{H}_V$-Matrix gilt:*

$$N_{\mathcal{H}_V,St}(k,\mathcal{T},\mathcal{Z}) \leq N_{\mathcal{H}_V,Mv}(k,\mathcal{T},\mathcal{Z}) \leq 2N_{\mathcal{H}_V,St}(k,\mathcal{T},\mathcal{Z}).$$

Beweis: Wir benutzen (3.45) sowie (3.46) und führen den Beweis analog zu dem von Lemma 3.6.26. □

Speziell die Abschätzung für den Speicherbedarf einer $\mathcal{H}_V$-Matrix ist natürlich sehr grob, da wir davon ausgehen, dass alle zulässigen Blöcke der Matrix eine VR(k)-Matrix darstellen. Hierbei spielen die Volterra-Grenzen eine wichtige Rolle, in Abbildung 3.9 haben wir ein Beispiel, welches nur mit R(k)-Matrizen[1)] auskommt, hier wäre der Aufwand offensichtlich nur etwa halb so groß wie bei einer $\mathcal{H}$-Matrix. Im Abschnitt 3.7 geben wir neben einer neuen Zulässigkeitsbedingung auch eine bessere Abschätzung für die von uns betrachteten Volterra-Grenzen an.

[1)] welche natürlich ebenfalls VR(k)-Matrizen sind, vgl. Bemerkung 3.5.30

3.6.5. Ausblick: $\mathcal{H}^2$-Matrizen

In (3.35) haben wir die Kernfunktion κ in einer Variablen durch ein Interpolationspolynom ersetzt. Man kann auch noch einen Schritt weiter gehen und eine Interpolation in beiden Variablen vornehmen

$$\kappa(x,y) \sim \tilde{\kappa}(x,y) := \sum_{l,j=0}^{k} \kappa(\zeta_l, \xi_j) L_l(x) P_j(y),$$

wobei für $l, j = 0, \dots, k$ durch L_l, P_j etwa Lagrange-Polynome vom Grad k bezeichnet seien und durch ζ_l, ξ_j die entsprechenden Interpolationspunkte. Eine Matrix $\boldsymbol{R} \in \mathbb{K}^{m\times n}$ kann damit für $k_{\text{sep}} = k+1$ dargestellt werden gemäß

$$\boldsymbol{R} = \boldsymbol{SKT} \quad \text{mit} \quad \boldsymbol{S} \in \mathbb{K}^{m\times k_{\text{sep}}}, \boldsymbol{T} \in \mathbb{K}^{k_{\text{sep}}\times n}, \boldsymbol{K} \in \mathbb{K}^{k_{\text{sep}}\times k_{\text{sep}}}. \tag{3.47}$$

Die Blockclusterbäume konstruiert man auf ähnliche Weise. Die zulässigen Blockcluster werden dabei gemäß (3.47) dargestellt. Effektiv werden allerdings nur die Matrizen $\boldsymbol{K}$ gespeichert, die Matrizen $\boldsymbol{S}$ und $\boldsymbol{T}$ drückt man durch *geschachtelte Basen* aus. Man hat also eine zusätzliche Hierarchie und bezeichnet daher diese Matrix-Darstellung als $\mathcal{H}^2$*-Matrizen*, siehe Hackbusch et.al. [27] oder Börm [4]. Dadurch kann der Aufwand auf $\mathcal{O}(n)$ reduziert werden.

Aus vornehmlich drei Gründen haben wir dieses Konzept nicht weiter verfolgt. Zum Ersten haben wir Volterra-Grenzen vorliegen, die schon bei der Benutzung von $\mathcal{H}$-Matrizen einen gewissen Aufwand erfordern. Zum Zweiten führen wir im nächsten Abschnitt neue Zulässigkeitsbedingungen ein, die eine für $\mathcal{H}^2$-Matrizen notwendige Interpolation in beiden Variablen nicht erlauben. Und zum Dritten werden wir sehen, dass wir im Falle der linearen und quasi-linearen Operatoren ebenfalls einen Aufwand von $\mathcal{O}(n)$ erreichen.

3.7. Neue Zulässigkeitsbedingungen

Im vorigen Abschnitt haben wir die Grundlagen des (erweiterten) $\mathcal{H}$-Matrix-Kalküls kennengelernt. In diesem Abschnitt werden wir die notwendigen Anpassungen an unser Problem vornehmen.

3.7.1. Aggregationskerne

In diesem Abschnitt formulieren wir für die Klasse von Kernen, welche die Phänomene der Aggregation beschreiben, eine passende Zulässigkeitsbedingung. Aus naheliegenden Gründen bezeichnen wir die Kerne als *Aggregationskerne*. Kerne dieser Art sind insbesondere symmetrisch und haben gewöhnlich eine Singularität im Nullpunkt.

Beispiel 3.7.1 (Aggregationskerne):

a) Smoluchowski-Kern, siehe (2.35) im Abschnitt 2.6.2:

$$\kappa(x,y) = \frac{(x+y)^2}{xy} = 2 + \frac{x}{y} + \frac{y}{x}. \tag{3.48}$$

b) Modifizierter Smoluchowski-Kern, siehe ebenfalls (2.35):

$$\kappa(x,y) = \frac{(x+y)^2}{(c_\mathrm{s}+xy)}, \quad c_\mathrm{s} > 0. \tag{3.49}$$

c) Aggregationskern des Emulsionsmodells, siehe (2.56) im Abschnitt 2.6.3:

$$\kappa(x,y) = c_1\,(x^{\frac{2}{3}}+y^{\frac{2}{3}})\,(x^{\frac{2}{9}}+y^{\frac{2}{9}})^{\frac{1}{2}}\,\exp\left[-c_2\left(\frac{x^{\frac{1}{3}}y^{\frac{1}{3}}}{x^{\frac{1}{3}}+y^{\frac{1}{3}}}\right)^4\right], \quad c_1, c_2 > 0. \tag{3.50}$$

Die Kernfunktion (3.48) ist bereits separabel, weist aber eine starke Singularität im Nullpunkt auf. Deswegen wurde der modifizierte Smoluchowski-Kern (3.49) eingeführt, der das zugrundeliegende Phänomen besser modelliert. Dabei ist c_s eine kleine Konstante. Wir werden zunächst mit Hilfe des modifizierten Smoluchowski-Kerns ein Zulässigkeitskriterium formulieren. Dazu nutzen wir aus, dass die n-ten partiellen Ableitungen explizit dargestellt werden können. Anschließend werden wir das Kriterium verallgemeinern.

Lemma 3.7.2: *Für die partiellen Ableitungen der Kernfunktion (3.49) gilt für $x, y \geq 0$*

$$\frac{\partial^n}{\partial x^n}\kappa(x,y) = (-1)^n n!\frac{y^{n-2}(c_\mathrm{s}-y^2)^2}{(c_\mathrm{s}+xy)^{n+1}} \quad \forall n > 1 \quad \textit{und} \tag{3.51a}$$

$$\frac{\partial^n}{\partial y^n}\kappa(x,y) = (-1)^n n!\frac{x^{n-2}(c_\mathrm{s}-x^2)^2}{(c_\mathrm{s}+xy)^{n+1}} \quad \forall n > 1. \tag{3.51b}$$

Beweis: Für die Ableitungen nach x benutzen wir die vollständige Induktion nach n. Wir zeigen zunächst den Induktionsanfang

$$\begin{aligned}
\frac{\partial^2}{\partial x^2}\frac{(x+y)^2}{(c_\mathrm{s}+xy)} &= \frac{\partial}{\partial x}\frac{(x+y)(2c_\mathrm{s}+xy-y^2)}{(c_\mathrm{s}+xy)^2} \\
&= \frac{(2c_\mathrm{s}+2xy)\cdot(c_\mathrm{s}+xy)^2 - (x+y)(2c_\mathrm{s}+xy-y^2)\cdot 2(c_\mathrm{s}+xy)y}{(c_\mathrm{s}+xy)^4} \\
&= \frac{2c_\mathrm{s}^2+4c_\mathrm{s}xy+2x^2y^2-4c_\mathrm{s}xy-2x^2y^2+2xy^3-4c_\mathrm{s}y^2-2xy^3+2y^4}{(c_\mathrm{s}+xy)^3} \\
&= \frac{2c_\mathrm{s}^2-4c_\mathrm{s}y^2+2y^4}{(c_\mathrm{s}+xy)^3} = 2\frac{(c_\mathrm{s}-y^2)^2}{(c_\mathrm{s}+xy)^3}.
\end{aligned}$$

Somit gilt (3.51a) für n=2. Sei nun die Behauptung (3.51a) für ein gewisses $n \in \mathbb{N}, n > 2$ erfüllt, so gilt für $n+1$

$$\begin{aligned}
\frac{\partial^{n+1}}{\partial x^{n+1}}\kappa(x,y) &= \frac{\partial}{\partial x}(-1)^n n!\frac{y^{n-2}(c_\mathrm{s}-y^2)^2}{(c_\mathrm{s}+xy)^{n+1}} \\
&= (-1)^n n! y^{n-2}(c_\mathrm{s}-y^2)^2\left(-\frac{(n+1)(c_\mathrm{s}+xy)^n y}{(c_\mathrm{s}+xy)^{2n+2}}\right) \\
&= (-1)^{n+1}(n+1)!\frac{y^{n-1}(c_\mathrm{s}-y^2)^2}{(c_\mathrm{s}+xy)^{n+2}}.
\end{aligned}$$

Somit gilt (3.51a) für beliebige $n > 2$. Die Darstellung (3.51b) der Ableitungen nach y folgt dann aufgrund der Symmetrie von κ. □

Lemma 3.7.3: *Die Kernfunktion (3.49) lässt sich auf jedem Intervall $[a, b] \subset (0, 1]$ für ein festes $\hat{y} \in (0, 1]$ bzgl. x als unendliche Taylor-Reihe darstellen, falls die Bedingung*

$$b - a < 2a \tag{3.52}$$

erfüllt ist und als Entwicklungspunkt gerade $x_0 = (a + b)/2$ gewählt wird.

Beweis: Gemäß Lemma 3.7.2 ist κ auf $[0, 1]$ bzgl. x beliebig oft differenzierbar. Es bleibt noch die Konvergenz des Restgliedes R_n zu zeigen. Sei $x \in [a, b], \hat{y} \in (0, 1]$ und sei außerdem ein $\theta \in (0, 1)$ gegeben. Zunächst bemerken wir, dass der Ausdruck $[x_0(1 - \theta) + x\theta]$ jeden Punkt zwischen x_0 und x darstellt, wenn θ zwischen 0 und 1 variiert. Daher kann man ihn durch a nach unten abschätzen gemäß

$$[x_0(1 - \theta) + x\theta] \geq a. \tag{3.53}$$

Mit der Darstellung (3.51a) und der Abschätzung (3.53), gilt für das Restglied

$$\begin{aligned}
|R_n(x, \hat{y})| &= \left| \frac{(x - x_0)^{n+1}}{(n+1)!} \frac{\partial^{n+1}}{\partial x^{n+1}} \kappa(x_0 + \theta(x - x_0), \hat{y}) \right| \\
&= \left| \frac{(x - x_0)^{n+1}}{(n+1)!} \frac{(-1)^{n+1}(n+1)! \hat{y}^{n-1} (c_\mathrm{s} - \hat{y}^2)^2}{(c_\mathrm{s} + (x_0 + \theta(x - x_0))\hat{y})^{n+2}} \right| \\
&\leq \frac{(c_\mathrm{s} - \hat{y}^2)^2 \, \hat{y}^{-2}}{c_\mathrm{s} + [x_0(1 - \theta) + x\theta]} \left| \frac{\hat{y}(x - x_0)}{c_\mathrm{s} + [x_0(1 - \theta) + x\theta]\hat{y}} \right|^{n+1} \\
&\leq \frac{(c_\mathrm{s} - \hat{y}^2)^2}{(c_\mathrm{s} + a)\hat{y}^2} \left| \frac{\hat{y}(x - x_0)}{c_\mathrm{s} + a\hat{y}} \right|^{n+1} \leq \frac{(c_\mathrm{s} - \hat{y}^2)^2}{(c_\mathrm{s} + a)\hat{y}^2} \left| \frac{|x - x_0|}{a} \right|^{n+1}.
\end{aligned}$$

Für festes $\hat{y}$ lässt sich der Faktor $\frac{(c_\mathrm{s} - \hat{y}^2)^2}{(c_\mathrm{s} + a)\hat{y}^2}$ durch eine Konstante c_y abschätzen. Aufgrund der Wahl von x_0 gilt gerade $|x - x_0| \leq (b - a)/2$. Somit gilt

$$|R_n(x, \hat{y})| \leq c_y \left(\frac{b - a}{2a} \right)^{n+1}$$

und wegen Bedingung (3.52) konvergiert diese Reihe gegen Null für alle $x \in [a, b]$. Daraus folgt die Behauptung. □

Folgerung 3.7.4: *Wegen der Symmetrie des modifizierten Smoluchowski-Kerns (3.49) gilt die Aussage von Lemma 3.7.3 analog für die Taylor-Entwicklung nach der zweiten Variable.*

Wir möchten keine unendliche Taylor-Entwicklungen benutzen, sondern nur Entwicklungen bis zu einer gewissen Ordnung der Ableitungen. Dazu definieren wir den folgenden Operator.

Definition 3.7.5: Besitze die Funktion f auf dem Intervall $[a,b]$ stetige Ableitungen bis zum Grad k, und existiere $f^{(k+1)}$ wenigstens im Inneren von $[a,b]$. Dann definiert für $x_0 \in [a,b]$

$$\Pi_T^k[f](x) = \sum_{j=0}^{k} \frac{f^{(j)}(x_0)}{k!}(x-x_0)^k$$

den *Taylorschen Approximationsoperator* bzgl. des Entwicklungspunktes x_0 (abgeschnittene Taylor-Reihe).

Lemma 3.7.6: *Sei der modifizierte Smoluchowski-Kern $\kappa(x,y) = \frac{(x+y)^2}{(c_s+xy)}$, $c_s > 0$ vorgelegt. Dann kann $\kappa(.,\hat{y})$ auf dem Intervall $[a,b] \in (0,1]$ bzgl. der ersten Variable durch die abgeschnittene Taylor-Reihe*

$$\Pi_T^k[\kappa(.,\hat{y})](x) = \sum_{\nu=0}^{k-1} \frac{1}{\nu!}\partial_x^\nu \kappa(x_0,\hat{y})(x-x_0)^\nu, \quad k \in \mathbb{N},\ k>1,\ x,x_0 \in [a,b]$$

approximiert werden, für ein festes $\hat{y} \in (0,1]$, mit dem Fehler

$$|\kappa(x,\hat{y}) - \Pi_T^k[\kappa(.,\hat{y})](x)| \le \frac{(c_s-\hat{y}^2)^2}{(c_s+x_0\hat{y})\hat{y}^2}\frac{1}{2^{k-1}}, \tag{3.54}$$

falls die Bedingung $b-a<2a$ erfüllt ist, vgl. (3.52).

Beweis: Wir wählen als Entwicklungspunkt für die Taylor-Reihe $x_0 := (a+b)/2$. Damit sind die Voraussetzungen von Lemma 3.7.3 erfüllt und für die Taylor-Entwicklung von κ auf dem Intervall $[a,b]$ gilt

$$\kappa(x,\hat{y}) = \sum_{\nu=0}^{\infty} \frac{1}{\nu!}\partial_x^\nu \kappa(x_0,\hat{y})(x-x_0)^\nu.$$

Für das Restglied $\sum_{\nu=k}^{\infty} \frac{1}{\nu!}\partial_x^\nu \kappa(x_0,\hat{y})(x-x_0)^\nu$, $k>1$ dieser Entwicklung ist für ein festes $\hat{y} \in (0,1]$ mit Lemma 3.7.2 zunächst die folgende Abschätzung möglich

$$\begin{aligned}
|\kappa(x,\hat{y}) - \Pi_T^k[\kappa(.,\hat{y})](x)| &= \Big|\sum_{\nu=k}^{\infty} \frac{1}{\nu!}\partial_x^\nu \kappa(x_0,\hat{y})(x-x_0)^\nu\Big| \\
&= \Big|\sum_{\nu=k}^{\infty} \frac{1}{\nu!}(-1)^\nu \nu! \frac{\hat{y}^{\nu-2}(c_s-\hat{y}^2)^2}{(c_s+x_0\hat{y})^{\nu+1}}(x-x_0)^\nu\Big| \\
&\le \frac{(c_s-\hat{y}^2)^2}{(c_s+x_0\hat{y})\hat{y}^2}\sum_{\nu=k}^{\infty}\Big|\frac{\hat{y}(x-x_0)}{c_s+x_0\hat{y}}\Big|^\nu \\
&\le \frac{(c_s-\hat{y}^2)^2}{(c_s+x_0\hat{y})\hat{y}^2}\Big(\frac{|x-x_0|}{x_0}\Big)^k\sum_{\nu=0}^{\infty}\Big|\frac{x-x_0}{x_0}\Big|^\nu,
\end{aligned} \tag{3.55}$$

für alle $x \in [a,b]$. Aus der Bedingung $(b-a)<2a$ folgt nun

$$2b-2a < a+b \quad \text{bzw.} \quad (b-a) < \frac{a+b}{2}.$$

Demnach erhalten wir aus der Definition von x_0 und der Abschätzung $|x-x_0| \leq (b-a)/2$ für alle $x \in [a,b]$

$$2|x - x_0| \leq (b-a) < \frac{a+b}{2} = x_0$$

und somit

$$\frac{|x - x_0|}{x_0} < \frac{1}{2}, \quad \forall x \in [a,b]. \tag{3.56}$$

Damit folgt aus (3.55)

$$|\kappa(x,\hat{y}) - \Pi_T^k[\kappa(.,\hat{y})](x)| \leq \frac{(c_\mathrm{s} - \hat{y}^2)^2}{(c_\mathrm{s} + x_0\hat{y})\hat{y}^2}\frac{1}{2^{k-1}}, \quad \forall x \in [a,b]$$

□

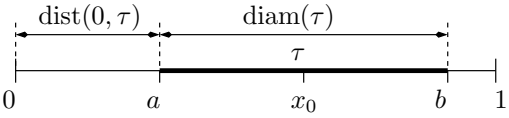
Folgerung 3.7.7: *Aufgrund der Symmetrie der Aggregationskerne gilt die Abschätzung aus Lemma 3.7.6 in analoger Weise für y.*

Der Fehler (3.54) welcher bei der Approximation durch $\Pi_T^k[\kappa]$ entsteht, kann für ein festes $\hat{y} \in (0,1]$ mithilfe des Parameters k gesteuert werden. Damit erlaubt das Kriterium (3.52) aus dem Lemma 3.7.3 die Formulierung einer Zulässigkeitsbedingung für den modifizierten Smoluchowski-Kern (3.49). Dabei sehen wir das Intervall $[a,b]$ gerade als Träger eines Clusters τ an. Entsprechend gilt

$$a = \mathrm{dist}(0,\tau) \quad \text{und} \quad b - a = \mathrm{diam}(\tau).$$

Bezüglich τ erhalten wir somit eine alternative Formulierung der Bedingung (3.52):

$$\mathrm{diam}(\tau) \leq 2\,\mathrm{dist}(0,\tau). \tag{3.57}$$



Mit dieser und mit der Symmetrie des Kerns formulieren wir die erste Version einer Zulässigkeitsbedingung.

Definition 3.7.8 (Zulässigkeit (modifizierter Smoluchowski-Kern)): Zwei Cluster τ und σ genügen der Zulässigkeitsbedingung $\mathcal{Z}_S m$, falls gilt

$$\mathrm{diam}(\tau) < 2\,\mathrm{dist}(0,\tau) \quad \text{oder} \quad \mathrm{diam}(\sigma) < 2\,\mathrm{dist}(0,\sigma),$$

die Entwicklung der Kernfunktion erfolgt dabei auf dem Cluster, der den größeren Abstand zum Punkt 0 hat.

Die Abschätzung (3.54) kann allerding auch beliebig schlecht werden, wenn $\hat{y}$ gegen Null strebt. Außerdem ist eine Fehlerabschätzung wie in Lemma 3.7.6 nicht möglich, wenn die partiellen Ableitungen beliebiger Ordnung nicht explizit verfügbar sind. Daher wollen wir ein allgemeineres Kriterium für die Approximation einer Kernfunktion mit Hilfe der *Cauchyschen Integralformel* formulieren und eine dazu passende Zulässigkeitsbedingung, die ebenfalls auf den Kern des Emulsionsmodells (3.50) anwendbar ist. Zunächst benötigen wir die folgenden Definitionen und Sätze.

Definition 3.7.9: Sei X ein metrischer Raum und $a, b \in \mathbb{R}$.

a) Eine stetige Abbildung $\gamma : [a, b] \to X$ nennt man einen *Weg* in X. Die Bildmenge $C_\gamma := \{\gamma(t) | t \in [a, b]\}$ heißt die durch γ erzeugte *Kurve*.

b) Der Weg $\gamma : [a, b] \to X$ heißt *rektifizierbar*, falls es eine Konstante L gibt, so dass für alle Zerlegungen $Z := [t_0, ..., t_n]$ des Intervalls $[a, b]$ stets

$$L(\gamma, Z) := \sum_{k=1}^{n} |\gamma(t_k) - \gamma(t_{k-1})| \leq L$$

gilt. Dabei nennen wir die Zahl $L(\gamma) := \sup_Z L(\gamma, Z)$ die Länge des Weges γ.

c) Eine Kurve C nennt man *Jordankurve*, wenn sie von einem Weg erzeugt werden kann, der auf $[a, b)$ injektiv ist, während $\gamma(a) = \gamma(b)$ gilt.

d) Eine Jordankurve heißt rektifizierbar, wenn sie von einem rektifizierbaren Weg erzeugt werden kann.

Bemerkung 3.7.10: Sei C eine rektifizierbare Jordankurve, dann besitzen alle erzeugenden Wege von C die gleiche Weglänge, und wir identifizieren die Weglänge mit der Länge der Kurve C (siehe z.B. Heuser [31, Kap. XXI.178]).

Definition 3.7.11: Sei M eine nichtleere Teilmenge eines metrischen Raumes.

a) M heißt *bogenweise zusammenhängend*, wenn sich zwei beliebige Punkte aus M stets durch eine stetige Kurve, die ebenfalls in M liegt, verbinden lassen.

b) M heißt *einfach zusammenhängend*, wenn sie bogenweise zusammenhängend ist und sich jede geschlossene Kurve in M stetig auf einen Punkt zusammenziehen lässt[1)], d.h., wenn es zu jeder stetigen Funktion $\varphi : [0, 1] \to M$ mit $\varphi(0) = \varphi(1)$ eine stetige Funktion $H : [0, 1] \times M \to M$ gibt, mit

$$H(0, x) = \varphi(x) \quad \text{und} \quad H(1, x) = x_0,$$

für alle $x \in M$, wobei x_0 ein fester Punkt aus M ist.

c) M nennt man ein *Gebiet*, wenn M offen und bogenweise zusammenhängend ist.

Satz 3.7.12: *Sei die Funktion f analytisch auf dem Intervall $[a, b] \subset \mathbb{R}$. Dann finden wir ein Gebiet $G \subset \mathbb{C}$, welches $[a, b]$ enthält und auf das wir f analytisch fortsetzen können, so dass f auf G holomorph ist.*

Beweis: Siehe z.B. Davis [11, Theorem 1.9.1] □

Satz 3.7.13 (Potenzreihenentwicklungssatz): *Die Funktion f sei holomorph auf dem Gebiet $G \subset \mathbb{C}$. Sei weiterhin eine Kreisscheibe mit dem Mittelpunkt z_0 und dem Radius δ gegeben, welche vollständig in G liegt. Dann gilt*

$$f(z) = \sum_{\nu=0}^{\infty} a_\nu (z - z_0)^\nu \quad \textit{mit} \quad a_\nu = \frac{f^{(\nu)}(z_0)}{\nu!}.$$

[1)] was anschaulich bedeutet, dass M keine *Löcher* enthält

Beweis: Siehe z.B. BUSAM und FREITAG [7, Theorem III.2.2] □

Definition 3.7.14: Sei $C \subset G \subset \mathbb{C}$ eine rektifizierbare Jordankurve, die von einem Weg $\gamma : [a, b] \to \mathbb{C}$ erzeugt wird, $a, b \in \mathbb{R}$. Weiter sei $f : G \to \mathbb{C}$ ein stetige Funktion. Dann ist das *Kurvenintegral von f über C* definiert gemäß

$$\oint_C f(z)dz = \int_a^b f(\gamma(t))\,\gamma'(t)dt.$$

Beispiel 3.7.15: Ein Kreis mit dem Mittelpunkt $z_0 \in \mathbb{C}$ und dem Radius δ, welcher entgegen dem Uhrzeigersinn durchlaufen wird, kann durch den Weg $\gamma : [0, 2\pi] \to \mathbb{C}$ erzeugt werden, der definiert ist durch

$$\gamma(t) = z_0 + \delta \exp[i\,t].$$

Soll der Kreis im Uhrzeigersinn durchlaufen werden, setzt man

$$\gamma(t) = z_0 + \delta \exp[-i\,t].$$

Satz 3.7.16 (Cauchysche Integralformel): *Die Funktion f sei auf dem einfach zusammenhängenden Gebiet $G \subset \mathbb{C}$ holomorph, und es sei $z_0 \in G$. Sei weiter C eine rektifizierbare Jordankurve, die in G liegt und den Punkt z_0 in mathematisch positivem Sinn, d.h., entgegen dem Uhrzeigersinn, umläuft. Dann gilt*

$$f^{(n)}(z_0) = \frac{n!}{2\pi i} \oint_C \frac{f(z)}{(z - z_0)^{n+1}} dz.$$

Beweis: Siehe z.B. DAVIS [11, Theorem 1.9.2] □

Folgerung 3.7.17: *Sei f analytisch auf dem Intervall $[a, b] \subset \mathbb{R}$. Dann existiert ein einfach zusammenhängendes Gebiet $G \subset \mathbb{C}$, das $[a, b]$ enthält und eine rektifizierbare Jordankurve C, die das Intervall $[a, b]$ umschließt und in G liegt. Außerdem gilt die Abschätzung*

$$|f^{(n)}(x)| \leq \frac{L(C)\,n!}{2\pi\delta^{n+1}} \max_{z \in C} |f(z)|, \quad x \in [a, b],$$

wobei $L(C)$ die Länge der Kurve C bezeichne und δ den kleinsten Abstand zwischen der Kurve C und dem Intervall $[a, b]$.

Beweis: Behauptung folgt direkt aus den Sätzen 3.7.12 und 3.7.16. □

Lemma 3.7.18: *Sei f eine Funktion auf dem Intervall $[a, b] \subset [0, 1]$. Sei weiter $x_0 := (a+b)/2$ und $C \subset \mathbb{C}$ eine Kreiskurve mit Mittelpunkt x_0 und dem Radius $\delta > (b-a)/2$. Wenn ein Gebiet $G \subset \mathbb{C}$ existiert, welches C enthält und auf welchem sich die Funktion f komplex fortsetzen lässt, so dass sie in G holomorph ist, siehe Abbildung 3.10, so lässt sich f auf dem Intervall $[a, b]$ in eine Taylor-Reihe entwickeln und kann mit dem Fehler*

$$|f(x) - \Pi_T^k[f](x)| \leq \|f\|_{\infty,C} \frac{|x - x_0|^k}{\delta^k} \frac{\delta}{\delta - |x - x_0|}, \quad \forall x \in [a, b] \tag{3.58}$$

approximiert werden.

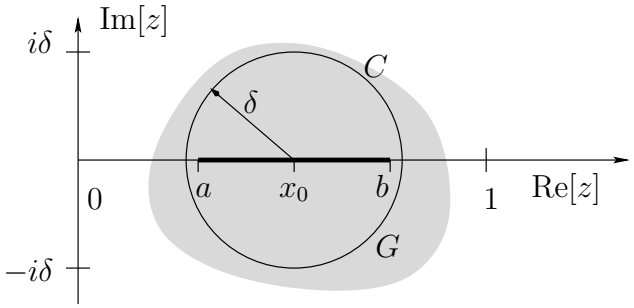


Abbildung 3.10.: Fortsetzung in die komplexe Ebene.

Beweis: Mithilfe der Cauchyschen Integralformel können wir die n-te Ableitung vom f abschätzen gemäß

$$|f^{(n)}(x_0)| \leq \frac{\|f\|_{\infty,C}\, L(C)\, n!}{2\pi\delta^{n+1}} = \frac{\|f\|_{\infty,C}\, n!}{\delta^n}.$$

Damit und mit $x_0 := (a+b)/2$ gilt

$$\begin{aligned} |f(x) - \Pi_T^k[f](x)| &= \left| \sum_{\nu=k}^{\infty} \frac{1}{\nu!} f^{(\nu)}(x_0)(x-x_0)^\nu \right| \\ &\leq \|f\|_{\infty,C} \sum_{\nu=k}^{\infty} \left| \frac{x-x_0}{\delta} \right|^\nu \\ &= \|f\|_{\infty,C}\, \frac{|x-x_0|^k}{\delta^k} \sum_{\nu=0}^{\infty} \left(\frac{|x-x_0|}{\delta} \right)^\nu \\ &= \|f\|_{\infty,C}\, \frac{|x-x_0|^k}{\delta^k} \frac{\delta}{\delta - |x-x_0|}, \end{aligned} \tag{3.59}$$

da $\frac{|x-x_0|}{\delta} < 1$ wegen $|x-x_0| \leq (b-a)/2$. □

Die Abschätzung (3.58) ist noch ziemlich allgemein. Wenn wir eine Singularität zulassen wollen und schärfere Bedingungen an das Entwicklungsintervall $[a,b]$ stellen, so liefert uns das folgende Lemma eine aussagekräftigere Abschätzung.

Lemma 3.7.19: *Sei f eine Funktion auf dem Intervall $[0,1]$ mit einer Singularität höchstens im Nullpunkt. Es lasse sich f komplex fortsetzen, so dass f auf dem Gebiet $G = (0,2] \times [-1,1] \subset \mathbb{C}$ holomorph ist. Dann lässt sich f auf jedem Intervall $[a,b] \in (0,1]$ in eine Taylorreihe entwickeln. Falls die Intervallgrenzen a, b zusätzlich die η-Bedingung*

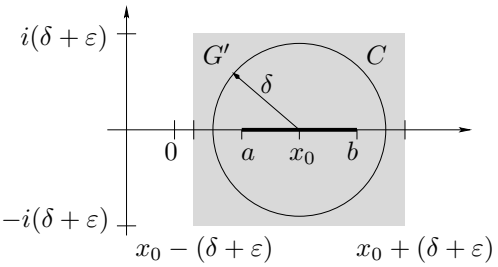
$$(b-a) < 2\,\eta\, a, \quad \eta \in \left(\frac{b-a}{2a}, 1 \right), \tag{3.60}$$

erfüllen, so kann man f mit dem Fehler

$$|f(x) - \Pi_T^k[f](x)| \leq \|f\|_{\infty,C}\, \frac{1}{2^{k-1}} \tag{3.61}$$

durch die abgeschnittene Taylor-Reihe approximieren. Dabei ist $C \in \mathbb{C}$ ein Kreis mit dem Mittelpunkt $x_0 := (a+b)/2$ und einem Radius $\delta > (b-a)/2$.

Beweis: Sei zunächst $[a,b] \subset (0,1]$ beliebig. Wir wählen einen Kreis C mit dem Mittelpunkt x_0 und Radius δ gemäß Voraussetzung. Außerdem gelte $\delta < x_0$, (vgl. auch Abbildung 3.10), da sonst bei einer Singularität im Nullpunkt diese von C eingeschlossen würde[1)] und somit insbesondere $\delta < 1$. Wir können δ stets in der Form $\delta + 2\varepsilon = x_0$ darstellen, $\varepsilon > 0$. Somit ist C in dem Gebiet



$$G' = [x_0 - (\delta+\varepsilon), x_0 + (\delta+\varepsilon)] \times [-(\delta+\varepsilon), \delta+\varepsilon]$$

vollständig enthalten, vgl. auch obenstehende Skizze. Wegen $G' \subset G$ können wir Lemma 3.7.18 anwenden und somit existiert die behauptete Taylor-Reihe für beliebige $[a,b] \subset (0,1]$. Sei nun zusätzlich die Bedingung (3.60) erfüllt. Wir stellen δ nun in der Form $\delta = x_0\,\eta'$ dar, dabei ergibt sich aus den Bedingungen

$$(b-a)/2 < \delta = \eta' x_0 = \eta'(a+b)/2 < (a+b)/2,$$

welche wir an δ stellen, dass $\eta' \in \left(\frac{b-a}{b+a}, 1\right)$. Damit folgt aus (3.60)

$$\frac{b-a}{2} + \eta\frac{b-a}{2} < \eta a + \eta\frac{b-a}{2} \quad \text{bzw.} \quad \frac{b-a}{2}(1+\eta) < \eta\frac{a+b}{2}$$

und daraus ergibt sich wegen $|x - x_0| \le (b-a)/2$

$$|x - x_0| < \frac{\eta}{1+\eta}\, x_0, \quad \forall x \in [a,b]. \tag{3.62}$$

Wir setzen nun

$$\eta' = 2\frac{\eta}{1+\eta}, \quad \text{wobei} \quad \eta \in \left(\frac{b-a}{b+3a}, 1\right)$$

und erhalten aus (3.62)

$$|x - x_0| \le \frac{1}{2}\delta \quad \text{bzw.} \quad \frac{|x-x_0|}{\delta} \le \frac{1}{2}.$$

Damit können wir die Abschätzung (3.59) aus dem Lemma 3.7.18 letztendlich in der Form (3.61) wie behauptet darstellen. Aufgrund von Bedingung (3.60) gilt insbesondere $(b-a)/2a < \eta$ und somit folgt wegen $(b-a)/(b+3a) < (b-a)/2a$, dass

$$\eta \in \left(\frac{b-a}{2a}, 1\right).$$

□

Lemma 3.7.19 erlaubt eine bessere Abschätzung für die Taylor-Approximation des modifizierten Smoluchowski-Kerns (3.49). Dabei wird der Einfluss von $x_{\max}$ auf den Fehler ebenfalls berücksichtigt.

Lemma 3.7.20: *Der modifizierte Smoluchowski-Kern $\kappa(.,\hat{y})$ lässt sich auf dem Intervall $[a,b] \subset (0,x_{\max}] \subset [0,1]$ bzgl. der ersten Variablen in eine Taylor-Reihe entwickeln, mit dem Entwicklungspunkt $x_0 := (a+b)/2$ und einem festen $\hat{y} \in [0,x_{\max}]$. Genügen a und b außerdem der η-Bedingung (3.60) aus dem Lemma 3.7.19, so können wir den Fehler, welcher durch Approximation von κ auf $[a,b]$ durch die abgeschnittene Taylor-Reihe $\tilde{\kappa}_k, k \in \mathbb{N}$ entsteht, abschätzen gemäß:*

$$|\kappa(x,\hat{y}) - \Pi_T^k[\kappa(.,\hat{y})](x)| \leq 11 \frac{x_{\max}^2}{c_{\mathrm{s}}} \frac{1}{2^{k-1}}.$$

Beweis: Die Funktion $\kappa(.,\hat{y})$ ist auf dem Gebiet $G = (0,2] \times [-1,1] \subset \mathbb{C}$ mit festem $\hat{y} \in [0,x_{\max}]$ holomorph. Somit können $\kappa(.,\hat{y})$ gemäß Lemma 3.7.19 auf dem Intervall $[a,b] \subset (0,x_{\max}]$ als Taylor-Reihe darstellen. Sei weiterhin die η-Bedingung erfüllt. Für alle $z \in C$ gilt

$$\mathrm{Re}[z] \in (0, 2\,x_{\max}), \quad \text{und} \quad \mathrm{Im}[z] \in [-\delta, \delta]$$

und somit folgt $|z| \leq \sqrt{5}\,x_{\max}$ wegen $\delta < x_{\max}$. Für alle $z \in C$, $\hat{y} \in [0,x_{\max}]$ haben wir dann

$$|\kappa(z,\hat{y})| = \left| \frac{(z+\hat{y})^2}{c_{\mathrm{s}} + z\hat{y}} \right| \leq \frac{(|z|+\hat{y})^2}{c_{\mathrm{s}}} \leq \frac{(\sqrt{5}\,x_{\max} + x_{\max})^2}{c_{\mathrm{s}}} \leq \frac{11\,x_{\max}^2}{c_{\mathrm{s}}}$$

und somit

$$\|\kappa(.,\hat{y})\|_{C,\infty} \leq 11 x_{\max}^2 / c_{\mathrm{s}}. \tag{3.63}$$

Zusammen mit Abschätzung (3.61) aus Lemma 3.7.19 folgt aus (3.63) die behauptete Fehlerabschätzung. □

Folgerung 3.7.21: *Aufgrund der Symmetrie des modifizierten Smoluchowski-Kerns, gelten die Aussagen von Lemma 3.7.20 entsprechend für die Entwicklung der Kernfunktion κ nach der zweiten Variable.*

Das Lemma 3.7.19 erlaubt ebenfalls Aussagen über die Approximation der Kernfunktion (3.50) durch ihre abgeschnittene Taylor-Reihe.

Lemma 3.7.22: *Es sei κ die Kernfunktion*

$$\kappa(x,y) = c_1 (x^{\frac{2}{3}} + y^{\frac{2}{3}})(x^{\frac{2}{9}} + x^{\frac{2}{9}})^{\frac{1}{2}} \exp\left[-c_2 \left(\frac{x^{\frac{1}{3}} y^{\frac{1}{3}}}{x^{\frac{1}{3}} + y^{\frac{1}{3}}} \right)^4 \right]$$

mit $x,y \in [0,x_{\max}] \subset [0,1], c_1, c_2 > 0$ gegeben, vgl. (3.50), Beispiel 3.7.1. Dann lässt sich $\kappa(.,\hat{y})$ auf dem Intervall $[a,b] \subset (0,x_{\max}] \subset [0,1]$ in eine Taylor-Reihe entwickeln, mit dem Entwicklungspunkt $x_0 := (a+b)/2$. Ist außerdem die η-Bedingung erfüllt,

vgl. (3.60) Lemma 3.7.19, so approximiert die abgeschnittene Taylor-Reihe $\tilde{\kappa}_k(.,\hat{y})$ die Kernfunktion κ auf dem Intervall $[a,b]$ mit dem Fehler

$$|\kappa(x,\hat{y}) - \Pi_T^k\left[\kappa(.,\hat{y})\right](x)| \leq c_1 5 \exp[c_2\, x_{\max}] \frac{1}{2^{k-1}}. \tag{3.64}$$

Intervall $[a,b] \subset (0,x_{\max}) \subset [0,1]$, mit einem festen $\hat{y} \in [0,x_{\max}]$.

Beweis: Die Funktion $\kappa(.,\hat{y})$ ist auf dem Gebiet $G = (0,2] \times [-1,1] \subset \mathbb{C}$ mit festem $\hat{y} \in [0,x_{\max}]$ holomorph. Nach Lemma 3.7.19 existiert somit die Taylor-Entwicklung von $\kappa(.,\hat{y})$ auf jedem Intervall $[a,b] \subset (0,x_{\max}]$. Sei nun außerdem die η-Bedingung erfüllt. Für die Abschätzung des Terms $\|\kappa(.,\hat{y})\|_{\infty,C}$ benutzen wir wieder, dass $|z| \leq \sqrt{5}\, x_{\max}$ für alle $z \in C$. Damit gilt für alle $\hat{y} \in [0,x_{\max}]$

$$\begin{aligned}
|\kappa(z,\hat{y})| &= \left| c_1 (z^{\frac{2}{3}} + \hat{y}^{\frac{2}{3}})(z^{\frac{2}{9}} + \hat{y}^{\frac{2}{9}})^{\frac{1}{2}} \exp\left[-c_2 \left(\frac{z^{\frac{1}{3}}\hat{y}^{\frac{1}{3}}}{z^{\frac{1}{3}} + \hat{y}^{\frac{1}{3}}} \right)^4 \right] \right| \\
&\leq c_1 |z^{\frac{2}{3}} + \hat{y}^{\frac{2}{3}}||z^{\frac{2}{9}} + \hat{y}^{\frac{2}{9}}|^{\frac{1}{2}} \exp\left[c_2 \left| \frac{z^{\frac{1}{3}}\hat{y}^{\frac{1}{3}}}{z^{\frac{1}{3}} + \hat{y}^{\frac{1}{3}}} \right|^4 \right] \\
&\leq c_1 (|z|^{\frac{2}{3}} + \hat{y}^{\frac{2}{3}})(|z|^{\frac{2}{9}} + \hat{y}^{\frac{2}{9}})^{\frac{1}{2}} \exp\left[c_2 \left(\frac{|z|^{\frac{1}{3}}\hat{y}^{\frac{1}{3}}}{|z|^{\frac{1}{3}}} \right)^4 \right] \\
&\leq c_1 \left((\sqrt{5} x_{\max})^{\frac{2}{3}} + x_{\max}^{\frac{2}{3}} \right) \left((\sqrt{5} x_{\max})^{\frac{2}{9}} + x_{\max}^{\frac{2}{9}} \right)^{\frac{1}{2}} \exp\left[c_2\, x_{\max}^{\frac{4}{3}} \right] \\
&\leq c_1\, 5\, x_{\max}^{\frac{7}{9}} \exp\left[c_2\, x_{\max}^{\frac{4}{3}} \right]
\end{aligned}$$

und somit

$$\|\kappa(.,\hat{y})\|_{\infty,C} \leq c_1\, 5\, x_{\max}^{\frac{7}{9}} \exp\left[c_2\, x_{\max}^{\frac{4}{3}} \right]. \tag{3.65}$$

Mit (3.61) aus Lemma 3.7.19 folgt aus (3.65) die Behauptung. □

Folgerung 3.7.23: *Aufgrund der Symmetrie der Kernfunktion κ (3.50) gelten die Aussagen aus dem Lemma 3.7.22 entsprechend für eine Taylor-Darstellung bzgl. der zweiten Variable, d.h. von $\kappa(\hat{x},.)$, wobei $\kappa(\hat{x},.)$ wieder auf $[a,b] \subset (0,x_{\max}] \subset [0,1]$ definiert sei und $\hat{x} \in [0,x_{\max}]$.*

Das Lemma 3.7.18 stellt die Existenz einer Darstellung der von uns betrachteten nicht separablen Aggregationskerne (3.49) und (3.50) als Taylor-Reihe auf jedem Intervall $[a,b] \subset (0,x_{\max}] \subset [0,1]$ sicher. Stellen wir strengere Forderungen an das Entwicklungsintervall $[a,b]$, so erlaubt η-Bedingung (3.60) aus Lemma 3.7.19 eine Kontrolle des Fehlers bei der Verwendung der abgeschnittenen Taylor-Reihe. Sehen wir das Intervall $[a,b]$ als Träger eines Clusters an, so können wir mithilfe dieser Bedingung eine allgemeine Zulässigkeitsbedingung formulieren.

Definition 3.7.24 (Zulässigkeit für Aggregationskerne): Zwei Cluster τ und σ genügen der Zulässigkeitsbedingung $\mathcal{Z}_A(\eta)$, falls gilt

$$\operatorname{diam}(\tau) < 2\,\eta\,\operatorname{dist}(0,\tau) \quad \text{oder} \quad \operatorname{diam}(\sigma) < 2\,\eta\,\operatorname{dist}(0,\sigma), \tag{3.66}$$

mit einem η

$$\max\left(\frac{\operatorname{diam}(\tau)}{2\operatorname{dist}(0,\tau)}, \frac{\operatorname{diam}(\sigma)}{2\operatorname{dist}(0,\sigma)}\right) < \eta < 1,$$

(wobei auf dem Cluster entwickelt wird, der die größere Entfernung zum Nullpunkt besitzt).

Beispiel 3.7.25: Mit der neuen Zulässigkeitsbedingung $\mathcal{Z}_A(\eta)$ ergibt sich auch eine andere Struktur in der $\mathcal{H}$-Matrix, siehe Abbildung 3.11 links. Legen wir die erweiterte Zulässigkeitsbedingung $\mathcal{E}(\mathcal{Z}_\mathcal{A})$ zugrunde, mit zugehörigen Volterra-Funktionen $v_\mathrm{a} \equiv 0$ und $v_\mathrm{b}(x) = x_{\max} - x$, mit $x \in [0, x_{\max}]$, so skizziert die rechte Matrix in Abbildung 3.11 die $\mathcal{H}_V$-Matrix des quasilinearen Operators.

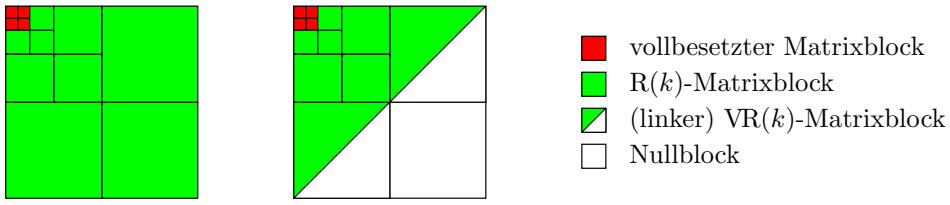


Abbildung 3.11.: $\mathcal{H}$-Matrizen mit der Zulässigkeitsbedingung $\mathcal{Z}_A$, $\eta = 0.9$.

3.7.2. Dispersionskerne

Wir zeigen in diesem Abschnitt, dass die Zulässigkeitsbedingung $\mathcal{Z}_A$ auch für Dispersionskerne Anwendung finden kann.

Beispiel 3.7.26 (Dispersionskern): Der Kern des Emulsionsmodells, setzt sich aus den drei Funktionen ν, β_br und φ zusammen, siehe (2.37), (2.45) und (2.46) im Abschnitt 2.6.3. Die vollständige Kernfunktion besitzt die Darstellung

$$\kappa(x,y) = c_1 y^{-\frac{11}{9}} \exp\left[-c_2\, y^{-\frac{5}{9}}\right] \exp\left[-c_3\left(\frac{2x-y}{y}\right)^2\right], \quad c_1, c_2, c_3 > 0. \tag{3.67}$$

Dabei gilt $\kappa \equiv 0$ für $y < x$.

Bemerkung 3.7.27: Die Kernfunktion 3.67 ist bereits teilweise separiert, daher beschränken wir uns auf die Approximation der Funktion

$$\tilde{\kappa}(x,y) = \exp\left[-c_3\left(\frac{2x-y}{y}\right)^2\right]. \tag{3.68}$$

Lemma 3.7.28: *Sei die Kernfunktion $\tilde{\kappa}$ gemäß (3.68) gegeben. Weiter sei das Intervall $[a,b] \subset (0, x_{\max}] \subset [0,1]$ gegeben, welches der η-Bedingung (3.60) aus Lemma 3.7.19 genügt. Sei außerdem $x_0 := (a+b)/2$, dann gilt:*

a) Die Funktion $\tilde{\kappa}(.,\hat{y})$ lässt sich auf dem Intervall $[a,b]$ in eine Taylor-Reihe in x_0 entwickeln, für ein festes $\hat{y} \in [a, x_{\max}]$. Dabei kann der Approximationfehler der abgeschnittenen Taylor-Reihe abgeschätzt werden durch

$$|\tilde{\kappa}(x,\hat{y}) - \Pi_T^k [\tilde{\kappa}(.,\hat{y})](x)| \leq \exp\left[16 c_3 \eta^2\right] \frac{1}{2^{k-1}}. \tag{3.69}$$

b) Die Funktion $\tilde{\kappa}(\hat{x},.)$ lässt sich auf dem Intervall $[a,b]$ in x_0 in eine Taylor-Reihe entwickeln, für ein festes $\hat{x} \in [0,b]$. Der Approximationsfehler der abgeschnittenen Taylor-Reihe kann dabei dargestellt werden durch

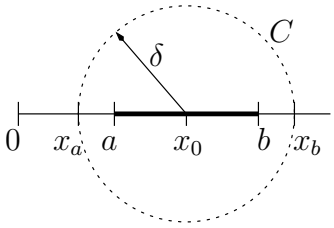
$$|\tilde{\kappa}(\hat{x},y) - \Pi_T^k [\tilde{\kappa}(\hat{x},.)](y)| \leq \exp\left[64 c_3 \frac{\eta^2(1+\eta)^2}{(1-\eta)^4}\right] \frac{1}{2^{k-1}}. \tag{3.70}$$

Beweis: Auch hier ist $\tilde{\kappa}(.,\hat{y})$ bzw. $\tilde{\kappa}(\hat{x},.)$ für festes $\hat{y} \in [a, x_{\max}]$ bzw. $\hat{x} \in [0,b]$ auf $(0,2] \times [-1,1] \subset \mathbb{C}$ holomorph, und somit existiert die behauptete Taylor-Darstellung auf jedem Intervall $[a,b] \in (0, x_{\max}]$ nach Lemma 3.7.19. Für die Abschätzung von $\|\tilde{\kappa}(.,\hat{y})\|_{C,\infty}$ bzw. $\|\tilde{\kappa}(\hat{x},.)\|_{C,\infty}$ für $\hat{y} \in [a, x_{\max}]$ bzw. $\hat{x} \in [0,b]$ beachten wir, dass der Ausdruck

$$|\exp[-c_3\, z]| = \exp[-\mathrm{Re}[c_3\, z]], \quad c_3 > 0, z \in \mathbb{C}$$

maximiert wird, wenn wir $\mathrm{Re}[z]$ minimieren. Zunächst führen wir zusätzlich die Punkte

$$x_a := \min_{z \in C}(\mathrm{Re}[z]) \quad \text{und} \quad x_b := \max_{z \in C}(\mathrm{Re}[z])$$



ein, vgl. nebenstehende Skizze. Insbesondere gilt $\delta = (x_b - x_a)/2 = \max_{z \in C}(\mathrm{Im}[z])$. Weiterhin gilt $x_0 = \delta + x_a$. Gemäß Beweis von Lemma 3.7.19 gibt es ein $\eta' \in \left(\frac{b-a}{b+a}, 1\right)$, so dass $x_0 \eta' = \delta$, da die Zulässigkeitsbedingung erfüllt ist. Somit gelten aufgrund der Zulässigkeitsbedingung $(b-a) < 2\,\eta\, a$ sowie der Definition $\eta' = 2\eta/(1+\eta)$ die nachfolgenden Beziehungen

$$\frac{\delta}{a} = \frac{(a+b)\eta'}{2a} < \frac{a(2\eta+2)\eta'}{2a} = \frac{(\eta+1)2\eta}{\eta+1} = 2\eta \tag{3.71a}$$

$$\frac{\delta}{x_a} = \frac{2\delta}{a+b-2\delta} = \frac{\eta'(a+b)}{(a+b)-\eta'(a+b)} = \frac{\eta'}{1-\eta'} = \frac{2\eta}{1-\eta} \tag{3.71b}$$

$$\begin{aligned} \frac{b}{x_a} &= \frac{2b}{a+b-2\delta} = \frac{2b}{(a+b)-\eta'(a+b)} = \frac{2b}{(a+b)(1-\eta')} \\ &< \frac{2}{1-\eta'} = \frac{2(1+\eta)}{1-\eta}. \end{aligned} \tag{3.71c}$$

a) Für $\tilde{\kappa}(x,\hat{y})$ mit $x \in C$ und $\hat{y} \in [a, x_{\max}]$ gilt

$$\begin{aligned}\mathrm{Re}\left[(2x-\hat{y})\right] &= \mathrm{Re}^2\left[2x-\hat{y}\right] - \mathrm{Im}^2\left[2x-\hat{y}\right] \geq -\mathrm{Im}^2\left[2x-\hat{y}\right] = -\mathrm{Im}^2\left[2x\right] \\ &\geq -4\delta^2\end{aligned}$$

und somit mit (3.71a)

$$\|\tilde{\kappa}(.,\hat{y})\|_{\infty,C} \leq \exp\left[c_3 \frac{4\delta^2}{a^2}\right] \leq \exp\left[16 c_3\, \eta^2\right].$$

b) Für $\tilde{\kappa}(\hat{x},.)$ haben wir für $y \in C$ und $\hat{x} \in [0,b]$ die Abschätzung

$$\begin{aligned}\mathrm{Re}\left[\left(\frac{2\hat{x}-\bar{y}}{\bar{y}}\right)^2\right] &= \mathrm{Re}\left[\left(\frac{2\hat{x}y-|y|^2}{|y|^2}\right)^2\right] \\ &= \frac{1}{|y|^4}\left(\mathrm{Re}^2\left[2\hat{x}y-|y|^2\right] - \mathrm{Im}^2\left[2\hat{x}y-|y|^2\right]\right) \\ &\geq -\frac{1}{|y|^4}\left(\mathrm{Im}^2\left[2\hat{x}y-|y|^2\right]\right) \\ &= -\frac{1}{|y|^4}\left(2x\,\mathrm{Im}[y]\right)^2 \geq -\frac{4\hat{x}^2\delta^2}{|y|^4} \geq -\frac{4b^2\delta^2}{x_a^4}\end{aligned}$$

und damit folgt mit (3.71b) und (3.71c)

$$\|\tilde{\kappa}(\hat{x},.)\|_{\infty,C} \leq \exp\left[c_3\frac{4b^2\delta^2}{x_a^4}\right] \leq \exp\left[16c_3\frac{\eta'^2}{(1-\eta')^4}\right] = \exp\left[64c_3\frac{\eta^2(1+\eta)^2}{(1-\eta)^4}\right].$$

Mit (3.61) aus Lemma 3.7.19 folgt dann die Behauptung. □

Beispiel 3.7.29: Setzen wir $\eta = 1/3$, so ergibt sich für die Abschätzungen

$\|\tilde{\kappa}(.,\hat{y})\|_{\infty,C} \leq \exp[1.8c_3]$ und

$\|\tilde{\kappa}(\hat{x},.)\|_{\infty,C} \leq \exp[64c_3]$.

Bemerkung 3.7.30: In der Abschätzung für den Dispersionskern kann besonders der Term $\|\kappa(\hat{x},.)\|_{\infty,C}$ in Abhängigkeit von der Konstanten c_3 sehr groß werden. Theoretisch kann man durch eine geeignete Wahl von k mithilfe des Terms $\frac{1}{2^{k-1}}$ den Fehler beliebig klein machen. Auch eine Anpassung der Zulässigkeitsbedingung $\mathcal{Z}_A$ wäre möglich. Wir werden in den numerischen Experimenten sehen, dass wir auch für relativ kleine k mit $\mathcal{Z}_A$ auch für den Dispersionskern gute Ergebnisse erzielen können.

Beispiel 3.7.31: In Abbildung 3.12 ist die $\mathcal{H}_V$-Matrix des linearen Operators skizziert, wobei die erweiterte Zulässigkeitsbedingung $\mathcal{E}(\mathcal{Z}_\mathcal{A})$ für Agglomerationskerne mit den zugehörigen Volterra-Funktionen $v_\mathrm{a}(x) = x$ und $v_\mathrm{b} \equiv x_{\max}$, für $x \in [0, x_{\max}]$.

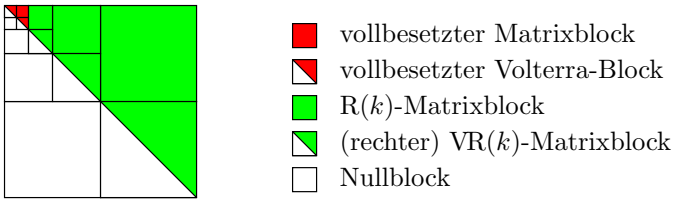


Abbildung 3.12.: $\mathcal{H}_V$-Matrix mit Zulässigkeitsbedingung $\mathcal{Z}_A$ und $\eta = 0.6$.

3.7.3. Lagrange-Interpolation

Durch die Lemmata 3.7.20, 3.7.22 und 3.7.28 haben wir für $k \in \mathbb{N}$ eine Fehlerabschätzung für die Approximation der jeweiligen Kernfunktion κ mit Hilfe der abgeschnittenen Taylor-Reihe. Die praktische Umsetzung dieser Approximation setzt allerdings die explizite Kenntnis der partiellen Ableitungen von κ bis zum Grad k voraus. Selbst wenn diese verfügbar sind, kann die Darstellung in einem Programm recht umständlich sein. Wir werden daher zur Approximation von κ den Operator der Lagrange-Approximation verwenden, welchen wir im Abschnitt 3.5.2 eingeführt hatten. Das Lemma 3.7.33 stellt den für die Fehlerabschätzung notwendigen Zusammenhang zwischen den Approximationsoperatoren her. Zuvor benötigen wir noch die folgende Definition.

Definition 3.7.32 (Lebesgue-Konstante): Sei $[a, b]$ ein reelles Intervall. Bezeichne weiter L_i^k das i-te Lagrange-Polynom bzgl. der disjunkten Stützstellen $\xi_i \in [a, b]$, $i = 0, \dots, k$, vgl. Definition 3.5.5. Dann heißt

$$\lambda_k(x) := \sum_{i=0}^{k} |L_i^k(x)|$$

die *Lebesgue-Funktion* bzgl. der Stützstellen ξ_i. Die *Lebesgue-Konstante* Λ_k bzgl. der Stützstellen ξ_i, ist dann definiert durch

$$\Lambda_k := \|\lambda_k\|_{\infty,[a,b]}. \tag{3.73}$$

Lemma 3.7.33: *Mit den Bezeichnungen aus Lemma 3.7.18 haben wir bzgl. der Zulässigkeitsbedingung (3.66) die folgende Abschätzung, wenn wir zur Approximation von f Lagrange-Polynome verwenden:*

$$\|f - \tilde{f}_k\|_{\infty,[a,b]} \leq (1 + \Lambda_k)\|f\|_{\infty,C}\frac{1}{2^{k-1}}.$$

Dabei bezeichnet Λ_k die so genannte Lebesgue-Konstante.

Beweis: Sei mit Π_T^k bzw. Π_L^k der Operator der Taylor- bzw. Lagrange-Approximation bezeichnet. Dabei lässt sich Π_L^k darstellen gemäß:

$$\Pi_L^k f(x) = \sum_{i=1}^{k} L_i^k(x) f(x_i). \tag{3.74}$$

Als Entwicklungspunkt für die Taylor-Polynome sei gerade $(a+b)/2$ gewählt. Mit (3.74), (3.73) und den Polynomeigenschaften von Π_T^k und Π_L^k gilt dann

$$\begin{aligned}\|f-\tilde{f}\|_{\infty,[a,b]} &= \|f-\Pi_T^k f+\Pi_T^k f-\Pi_L^k f\|_{\infty,[a,b]}\\ &\leq \|f-\Pi_T^k f\|_{\infty,[a,b]}+\|\Pi_L^k(\Pi_T^k f-f)\|_{\infty,[a,b]}\\ &\leq \|f-\Pi_T^k f\|_{\infty,[a,b]}+\Lambda_k\|\Pi_T^k f-f\|_{\infty,[a,b]}\\ &= (1+\Lambda_k)\,\|f-\Pi_T^k f\|_{\infty,[a,b]}.\end{aligned}$$

Lemma 3.7.19 liefert dann die Behauptung. □

Bemerkung 3.7.34: Die Wahl der Tschebyscheff-Knoten als Stützstellen für die Polynominterpolation erlaubt auf dem Referenzintervall $[-1,1]$ die Abschätzung

$$\Lambda_k \leq \frac{2}{\pi}\log k+1 \leq k+1$$

für die Lebesgue-Konstante Λ_k.

Beweis: Siehe z.B. Davis [11]. □

3.7.4. Komplexitätsabschätzung für $\mathcal{Z}_A$

Anders als bei der Standardzulässigkeit $\mathcal{Z}_S$ etwa, hängt bei $\mathcal{Z}_A$ die Zulässigkeit nicht mehr von zwei Clustern ab. Wir werden zeigen, dass dieser Umstand zur Verbesserung der Komplexität benutzt werden kann.

Voraussetzung 3.7.35: *In Abschnitt 3.6.1 haben wir geometrisch balancierte Clusterbäume vorausgesetzt. Somit nehmen wir an, dass sich, für einen Clusterbaum $\mathcal{T}_I$ zur Indexmenge I, die Durchmesser der Cluster auf dem selben Level nicht wesentlich unterscheiden. Es gebe daher ein $\gamma \geq 1$, so dass für alle Cluster τ_i und τ_j aus dem gleichen Level l*

$$\operatorname{diam}(\tau_i) \leq \gamma \operatorname{diam}(\tau_j)$$

gilt. Weiterhin seien die Indizes auf jeden Cluster τ so verteilt, dass wir deren Anzahl $|\tau|$ für jeden Level l abschätzen können durch

$$|\tau| \leq |I|\, C_{\mathrm{sz}}^l,$$

mit einer Konstanten $C_{\mathrm{sz}} \in [1/2, 1)$.

Folgerung 3.7.36: *Mit Voraussetzung 3.7.35 ist die Anzahl der zulässigen und nichtzulässigen Blockcluster bzgl. der Zulässigkeitsbedingung $\mathcal{Z}_A$ auf jedem Level l des Blockclusterbaums beschränkt[1)], so dass diese Schranke unabhängig vom Level gewählt werden kann.*

[1)] im Gegensatz zur Standardzulässigkeitsbedingung

Beweis: Die Zulässigkeit bei $\mathcal{Z}_A$ hängt nur noch von einem Cluster ab. Ist also der Cluster $\bar{\tau} \in \mathcal{L}_l(I)$ zulässig, so sind auch alle Paare $(\bar{\tau} \times \sigma), \sigma \in \mathcal{L}_l(I')$ zulässig. Dabei *verschwinden* die zu $\bar{\tau}$ gehörigen Indizes für eine Baumtiefe $> l$ aus der betrachteten Indexmenge.

Seien die Bezeichnungen aus Beispiel 3.6.8 gewählt, d.h. sei durch τ_i^l der i-te Cluster auf der l-ten Stufe bezeichnet. Es gelte $i = 1, 2, ...$ und o.B.d.A. seien die Cluster so angeordnet, dass für $i \leq j$ stets $\operatorname{dist}(\tau_i^l) \leq \operatorname{dist}(\tau_j^l)$ gilt. Außerdem nehmen wir an, dass für die von uns betrachteten Baumtiefen l, stets genügend viele Indizes in einem Cluster verbleiben, so dass eine Teilung möglich ist. Dann folgt aufgrund Voraussetzung 3.7.35 sowie $\mathcal{Z}_A$

$$\operatorname{diam}(\tau_i) \leq 2\,\eta \operatorname{dist}(0, \tau_i) \leq 2\,\eta \sum_{l=1}^{i-1} \operatorname{diam}(\tau_l) \leq 2\,\eta\,\gamma\,(i-1) \operatorname{diam}(\tau_i),$$

d.h. die Zulässigkeitsbedingung ist erfüllt, falls

$$\frac{1}{\gamma(i-1)} \leq 2\,\eta. \tag{3.75}$$

Da jeweils beim Übergang von Level l zu Level $l+1$ die nicht zulässigen Cluster halbiert werden und sich damit deren Anzahl verdoppelt[1)], gibt es aufgrund von (3.75) einen Level l_0, so dass in diesem mindestens ein Cluster bzgl. (3.75) zulässig ist. Somit existiert auch ein $\iota \in \mathbb{N}$, so dass für alle $i \leq \iota$ der Cluster $\tau_i^{l_0}$ nichtzulässig bzgl. (3.75) ist und zulässig sonst. Im Level l_0+1 haben wir dann 2ι Cluster vorliegen, von denen aufgrund der Wahl von ι wieder für alle $i \leq \iota$ der Cluster $\tau_i^{l_0+1}$ nichtzulässig bzgl. (3.75) ist und zulässig für $i > \iota$ sonst.

Somit gibt es in jedem Level $l > l_0$ genau 2ι Cluster, von denen bzgl. (3.75) jeweils ι Cluster zulässig bzw. nicht zulässig sind, insbesondere ist damit die Anzahl der zulässigen und nichtzulässigen Cluster bzgl. der alternativen Bedingung (3.75) beschränkt. Da aber (3.75) hinreichend für $\mathcal{Z}_A$ war, gilt diese Beschränkung auch bzgl. $\mathcal{Z}_A$. □

Folgerung 3.7.36 motiviert die folgende Definition.

Definition 3.7.37: Bezeichne $C_{\mathrm{cl}}(l)$ die Anzahl der zulässigen Blockcluster der Stufe l in einem Blockclusterbaum $\mathcal{T}_{I \times I'}$, dann definieren wir die Konstante

$$C_{\mathrm{cl}} := \max_l C_{\mathrm{cl}}(l).$$

Lemma 3.7.38: *Für den Speicheraufwand $N_{\mathcal{H},St}(k, \mathcal{T}, \mathcal{Z}_A)$ einer $\mathcal{H}$-Matrix bzgl. der Zulässigkeitsbedingung $\mathcal{Z}_A$ gilt*

$$N_{\mathcal{H},St}(k, \mathcal{T}, \mathcal{Z}_A) \leq C_{\mathrm{cl}}\, k \, \max\left(\frac{C_{\mathrm{sz}}}{1 - C_{\mathrm{sz}}}, \frac{C'_{\mathrm{sz}}}{1 - C'_{\mathrm{sz}}}\right) (|I| + |I'|).$$

[1)] binärer Clusterbaum

Beweis: Wenn wir beachten, dass wir auf der Ebene $l = 0$ keine zulässigen Cluster haben können, gilt:

$$\begin{aligned}
N_{\mathcal{H},St}(k,\mathcal{T},\mathcal{Z}_A) &\leq \sum_{\tau\times\sigma\in\mathcal{L}^+(\mathcal{T})} N_{L,St}(|\tau|,|\sigma|) \;+ \sum_{\tau\times\sigma\in\mathcal{L}^-(\mathcal{T})} N_{F,St}(|\tau|,|\sigma|) \\
&\leq \sum_{l\in L_T}\sum_{\tau\times\sigma\in\mathcal{L}_l(\mathcal{T})} \max(k,b_{\min})|\tau| + \sum_{l\in L_T}\sum_{\tau\times\sigma\in\mathcal{L}_l(\mathcal{T})} \max(k,b_{\min})|\sigma| \\
&\leq \max(k,b_{\min})\left[\sum_{l\in L_T}\sum_{\tau\times\sigma\in\mathcal{L}_l(\mathcal{T})} |I|C_{\mathrm{sz}}^l + \sum_{l\in L_T}\sum_{\tau\times\sigma\in\mathcal{L}_l(\mathcal{T})} |I'|C'^l_{\mathrm{sz}}\right] \\
&\leq C_{\mathrm{cl}}\ \max(k,b_{\min})\left[|I|\sum_{l=1}^{\infty} C_{\mathrm{sz}}^l + |I'|\sum_{l=1}^{\infty} C'^l_{\mathrm{sz}}\right] \\
&\leq C_{\mathrm{cl}}\ \max(k,b_{\min})\ \max\left(\frac{C_{\mathrm{sz}}}{1-C_{\mathrm{sz}}}, \frac{C'_{\mathrm{sz}}}{1-C'_{\mathrm{sz}}}\right)(|I|+|I'|).
\end{aligned}$$

□

Beispiel 3.7.39: Für Abbildung 3.11 gilt gerade $C_{\mathrm{cl}} = 3$. Wenn wir weiterhin $|I| = |I'| = 2^p$ mit einem $p \in \mathbb{N}$ annehmen und optimal clustern, so gilt weiterhin $C_{\mathrm{sz}} = 1/2$. Somit haben wir einen Speicheraufwand von $6\,k\,|I|$.

Folgerung 3.7.40: *Gemäß Lemma 3.6.26 und Lemma 3.7.38 können wir den Aufwand einer Matrix-Vektor-Multiplikation $N_{\mathcal{H},Mv}(k,\mathcal{T},\mathcal{Z}_A)$ abschätzen gemäß*

$$N_{\mathcal{H},St}(k,\mathcal{T},\mathcal{Z}_A) \leq N_{\mathcal{H},Mv}(k,\mathcal{T},\mathcal{Z}_A) \leq 2N_{\mathcal{H},St}(k,\mathcal{T},\mathcal{Z}_A).$$

Lemma 3.7.41: *Für den Speicheraufwand $N_{\mathcal{H}_V,St}(k,\mathcal{T},\mathcal{Z}_A)$ einer erweiterten $\mathcal{H}$-Matrix bzgl. der Zulässigkeitsbedingung $\mathcal{Z}_A$ gilt*

$$N_{\mathcal{H}_V,St}(k,\mathcal{T},\mathcal{Z}_A) \leq C_{\mathrm{cl}}\ \max(C_{\mathrm{v}},k,b_{\min})\ \max\left(\frac{C_{\mathrm{sz}}}{1-C_{\mathrm{sz}}}, \frac{C'_{\mathrm{sz}}}{1-C'_{\mathrm{sz}}}\right)(2|I|+|I'|).$$

Beweis: Zunächst gilt:

$$\begin{aligned}
\sum_{l\in L_T}\sum_{\tau\times\sigma\in\mathcal{L}(\mathcal{T})} C_{\mathrm{v}}|\tau| &\leq C_{\mathrm{v}}\sum_{l\in L_T}\sum_{\tau\times\sigma\in\mathcal{L}(\mathcal{T})} |I|C_{\mathrm{sz}}^l \leq C_{\mathrm{v}}C_{\mathrm{cl}}|I|\sum_{l=1}^{\infty} C_{\mathrm{sz}}^l \\
&\leq C_{\mathrm{v}}C_{\mathrm{cl}}|I|\frac{C_{\mathrm{sz}}}{1-C_{\mathrm{sz}}}.
\end{aligned}$$

Damit können wir mithilfe des Beweises von Lemma 3.7.38 den Speicheraufwand abschätzen gemäß

$$\begin{aligned}
N_{\mathcal{H}_V,St}(k,\mathcal{T},\mathcal{Z}_A) &\leq \sum_{\tau\times\sigma\in\mathcal{L}^+(\mathcal{T})} N_{V,St}(|\tau|,|\sigma|) \;+ \sum_{\tau\times\sigma\in\mathcal{L}^-(\mathcal{T})} N_{F,St}(|\tau|,|\sigma|) \\
&\leq N_{\mathcal{H},St}(k,\mathcal{T},\mathcal{Z}_A) + \sum_{l\in L_T}\sum_{\tau\times\sigma\in\mathcal{L}(\mathcal{T})} C_{\mathrm{v}}|\tau| \\
&\leq C_{\mathrm{cl}}\ \max(C_{\mathrm{v}},k,b_{\min})\ \max\left(\frac{C_{\mathrm{sz}}}{1-C_{\mathrm{sz}}}, \frac{C'_{\mathrm{sz}}}{1-C'_{\mathrm{sz}}}\right)(2|I|+|I'|).
\end{aligned} \tag{3.76}$$

□

Folgerung 3.7.42: *Gemäß Lemma 3.6.33 und Lemma 3.7.41 können wir den Aufwand einer Matrix-Vektor-Multiplikation $N_{\mathcal{H}_V,Mv}(k,\mathcal{T},\mathcal{Z}_A)$ abschätzen gemäß*

$$N_{\mathcal{H}_V,St}(k,\mathcal{T},\mathcal{Z}_A) \leq N_{\mathcal{H}_V,Mv}(k,\mathcal{T},\mathcal{Z}_A) \leq 2N_{\mathcal{H}_V,St}(k,\mathcal{T},\mathcal{Z}_A).$$

3.8. Quadratischer Operator

3.8.1. Allgemeines

Wir wollen nun den quadratischen Volterra-Operator diskretisieren, welcher im Quellterm der Koaleszenz (2.10a) auftritt. Hierbei müssen wir anders vorgehen als im linearen Fall. Der Operator hatte die Form

$$\mathcal{K}[f](x) = \int_0^x \kappa(x-y,y)\, f(x-y)\, f(y)\, dy =: g(x) \tag{3.77}$$

mit dem Quasi-Faltungskern $\kappa(x-y,y)$. Wir benutzen wieder die Galerkin-Diskretisierung und wählen als Ansatz- und Testraum den Raum, der von den stückweise konstanten Basisfunktionen $b_i, i = 1,\dots,n$, aufgespannt wird. Weiterhin legen wir eine äquidistante Zerlegung des Integrationsgebietes $[0,1]$ zugrunde, gemäß

$$0 = \xi_0 < \xi_1 < \cdots < \xi_n = 1. \tag{3.78}$$

Bemerkung 3.8.1: Insbesondere gilt:

a) $\operatorname{supp}(b_i) = [\xi_{i-1}, \xi_i], \quad \forall i = 1,\dots,n$

b) $h = \xi_i - \xi_{i-1}, \quad \forall i = 1,\dots,n,$

c) $\xi_i = ih, \quad \forall i = 0,\dots,n$

d) $\xi_i - \xi_j = \xi_{i-j}, \quad \forall i \geq j \quad \text{und} \quad i,j = 0,\dots,n,.$

Wir multiplizieren gemäß Beispiel 3.3.5 mit den Testfunktionen b_i und erhalten die Darstellung

$$\int_0^1 g(x) b_i(x)\, dx = \int_0^1 \int_0^x \kappa(x-y,y)\, f(x-y)\, f(y)\, dy\, b_i(x)\, dx \tag{3.79}$$

bzw. durch das Ausnutzen der Eigenschaften stückweise konstanter Basisfunktionen

$$\int_{\xi_{i-1}}^{\xi_i} g(x)\, dx = \int_{\xi_{i-1}}^{\xi_i} \int_0^x \kappa(x-y,y)\, f(x-y)\, f(y)\, dy\, dx, \tag{3.80}$$

für $i = 1,\dots,n$.

Wir ersetzen nun in der Darstellung (3.80) die Funktionen f und g durch ihre diskreten Darstellungen bzgl. des Ansatzraumes. Weiterhin benutzen wir wieder die finiten

Eigenschaften der b_i sowie die Tatsache, dass aufgrund des Volterra-Integrals die rechte Seite aus (3.80) für $x < y$ verschwindet und erhalten zunächst

$$\begin{aligned} g_i \int_{\xi_{i-1}}^{\xi_i} dx =& \sum_{j=1}^{i-1} f_j \int_{\xi_{i-1}}^{\xi_i} \int_{\xi_{j-1}}^{\xi_j} \kappa(x-y,y)\, f(x-y)\, dy\, dx \\ &+ f_i \int_{\xi_{i-1}}^{\xi_i} \int_{\xi_{i-1}}^{x} \kappa(x-y,y)\, f(x-y)\, dy\, dx \\ =& \sum_{k=1}^{n} \sum_{j=1}^{i-1} f_j f_k \int_{\xi_{i-1}}^{\xi_i} \int_{\xi_{j-1}}^{\xi_j} \kappa(x-y,y)\, b_k(x-y)\, dy\, dx \\ &+ f_i f_k \int_{\xi_{i-1}}^{\xi_i} \int_{\xi_{i-1}}^{x} \kappa(x-y,y)\, b_k(x-y)\, dy\, dx \end{aligned} \tag{3.81}$$

für $i = 1, \dots n$. In der rechten Seite von (3.81) kommt den Basisfunktionen $b_k(x-y)$ aufgrund der Faltung eine besondere Bedeutung zu. Wenn wir das Integral der rechten Seite für feste i, j und k betrachten, bemerken wir, da die Integrationsvariablen x bzw. y aus dem Intervall $[\xi_{i-1}, \xi_i]$ bzw. $[\xi_{j-1}, \xi_j]$ stammen, dass das Integral nur von Null verschieden sein kann für $k = i - j$ bzw. $k = i - j + 1$, vgl. hierzu auch Abbildung 3.13. Weiterhin ist aus der Abbildung ersichtlich, dass wir nun nicht mehr über das Viereck $[\xi_{i-1}, \xi_i] \times [\xi_{j-1}, \xi_j]$ integrieren müssen, sondern über die Dreiecke D_{i-j} und D_{i-j+1}.

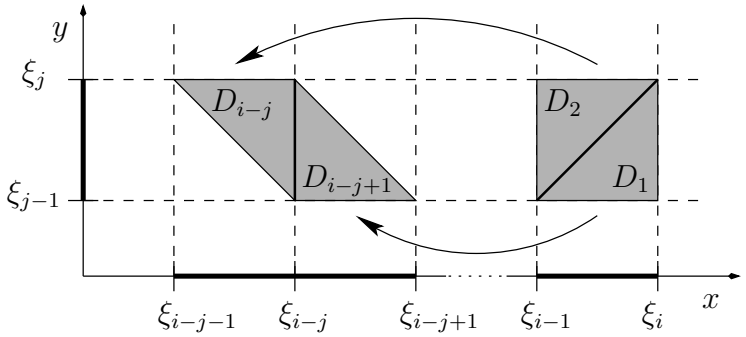


Abbildung 3.13.: Gefaltete Basisfunktionen.

Somit ergibt sich aus (3.81) und Bemerkung 3.8.1, für $i = 1, \dots, n$

$$\begin{aligned} g_i \cdot h =& \sum_{j=1}^{i-1} f_j f_{i-j} \int_{\xi_{i-1}}^{\xi_i} \int_{x-\xi_{i-j}}^{\xi_j} \kappa(x-y,y)\, dy\, dx \\ &+ f_j f_{i-j+1} \int_{\xi_{i-1}}^{\xi_i} \int_{\xi_{j-1}}^{x-\xi_{i-j}} \kappa(x-y,y)\, dy\, dx \\ &+ f_i f_1 \int_{\xi_{i-1}}^{\xi_i} \int_{\xi_{i-1}}^{x} \kappa(x-y,y)\, dy\, dx. \end{aligned} \tag{3.82a}$$

Bemerkung 3.8.2: Durch Transformation können wir (3.82a) für $i = 1, \dots, n$ in der alternativen Form darstellen

$$\begin{aligned} g_i \cdot h = &\sum_{j=1}^{i-1} f_j f_{i-j} \int_{\xi_{i-j-1}}^{\xi_{i-j}} \int_{\xi_{i-1}-x}^{\xi_j} \kappa(x,y)\, dy\, dx \\ &+ f_j f_{i-j+1} \int_{\xi_{i-j}}^{\xi_{i-j+1}} \int_{\xi_{j-1}}^{\xi_i - x} \kappa(x,y)\, dy\, dx \\ &+ f_i f_1 \int_{\xi_0}^{\xi_1} \int_{\xi_{i-1}}^{\xi_i - x} \kappa(x,y)\, dy\, dx. \end{aligned} \tag{3.82b}$$

Beweis: Wir zeigen die Transformation für den Term $\int_{\xi_{i-1}}^{\xi_i} \int_{x-\xi_{i-j}}^{\xi_j} \kappa(x-y,y)\, dy\, dx$. Durch Vertauschen der Integrationsgrenzen erhalten wir zunächst

$$\int_{\xi_{j-1}}^{\xi_j} \int_{\xi_{i-1}}^{y+\xi_{i-j}} \kappa(x-y,y)\, dx\, dy.$$

Durch die Transformation $x = \hat{x} + y$ ergibt sich

$$\int_{\xi_{j-1}}^{\xi_j} \int_{\xi_{i-1}-y}^{\xi_{i-j}} \kappa(\hat{x},y)\, d\hat{x}\, dy.$$

Durch erneutes Vertauschen und Umbenennen von $\hat{x}$ in x erhalten wir die behauptete Darstellung

$$\int_{\xi_{i-j-1}}^{\xi_{i-j}} \int_{\xi_{i-1}-x}^{\xi_j} \kappa(x,y)\, dy\, dx.$$

Die Transformation der anderen Terme erfolgt auf ähnliche Weise. □

Wir führen folgende Bezeichnungen ein

Definition 3.8.3: Sei $\boldsymbol{x} = (x_1, \dots, x_n)^\top \in \mathbb{R}^n$, dann definieren wir die Operatoren F_1 und $F_2 : \mathbb{R}^n \to \mathbb{R}^{n \times n}$ gemäß

$$F_1[\boldsymbol{x}] := \begin{pmatrix} x_1 & 0 & \dots & 0 \\ x_2 & x_1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ x_n & \dots & x_2 & x_1 \end{pmatrix} \quad \text{und} \quad F_2[\boldsymbol{x}] := \begin{pmatrix} 0 & 0 & \dots & 0 \\ x_1 & 0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ x_{n-1} & \dots & x_1 & 0 \end{pmatrix}. \tag{3.83}$$

Mit Definition 3.8.3 besitzt (3.82a) in Matrixschreibweise die folgende Gestalt

$$h \cdot \boldsymbol{g} = \big(\boldsymbol{K}_1 \circ F_1[\boldsymbol{f}] + \boldsymbol{K}_2 \circ F_2[\boldsymbol{f}]\big) \cdot \boldsymbol{f}, \tag{3.84}$$

wobei $\circ$ wieder die Hadamard-Multiplikation aus Definition 3.4.2 darstellt. Mit $\boldsymbol{f}$ bzw. $\boldsymbol{g}$ seien wieder die Koeffizientenvektoren von f bzw. g bezeichnet und für die Einträge der Matrizen $\boldsymbol{K}_1$ bzw. $\boldsymbol{K}_2$ gilt:

$$(K_1)_{ij} := \begin{cases} \int_{\xi_{i-1}}^{\xi_i} \int_{\xi_{j-1}}^{x-\xi_{i-j}} \kappa(x-y,y)\, dy\, dx & \text{für} \quad i > j \\ \int_{\xi_{i-1}}^{\xi_i} \int_{\xi_{i-1}}^{x} \kappa(x-y,y)\, dy\, dx & \text{für} \quad i = j \\ 0 & \text{für} \quad i < j, \end{cases} \tag{3.85a}$$

bzw.

$$(K_2)_{ij} := \begin{cases} \int_{\xi_{i-1}}^{\xi_i} \int_{x-\xi_{i-j}}^{\xi_j} \kappa(x-y,y)\,dy\,dx & \text{für} \quad i > j \\ 0 & \text{für} \quad i \leq j, \end{cases} \tag{3.85b}$$

für $i, j = 1, \ldots, n$, vgl. Abschnitt 3.3.2.

Definition 3.8.4: Für die Darstellung des Systems (3.84) definieren wir einen entsprechenden *diskreten quadratischen Operator* $\boldsymbol{Q} : \mathbb{R}^n \to \mathbb{R}^n$ gemäß

$$\boldsymbol{Q}\boldsymbol{f} := \big(\boldsymbol{K}_1 \circ F_1[\boldsymbol{f}] + \boldsymbol{K}_2 \circ F_2[\boldsymbol{f}]\big) \cdot \boldsymbol{f}.$$

Notation 3.8.5: Für die Anwendung der Operatoren F_1 bzw. F_2 auf den Koeffizientenvektor $\boldsymbol{f} \in \mathbb{R}^n$ nutzen wir die folgende Kurzschreibweise $\boldsymbol{F}_1 := F_1[\boldsymbol{f}]$ und $\boldsymbol{F}_2 := F_2[\boldsymbol{f}]$, wobei $\boldsymbol{F}_1$ und $\boldsymbol{F}_2$ natürlich Matrizen aus $\mathbb{R}^{n\times n}$ darstellen.

Folgerung 3.8.6: *Um den Operator $\boldsymbol{Q}$ auf einen Vektor $\boldsymbol{f}$ anzuwenden, müssen wir zunächst ein* Update *vornehmen, indem wir die Matrix $\boldsymbol{K}_1$ bzw. $\boldsymbol{K}_2$ komponentenweise mit $\boldsymbol{F}_1$ bzw. $\boldsymbol{F}_2$ multiplizieren. Erst nach der Addition von $\boldsymbol{K}_1 \circ \boldsymbol{F}_1$ und $\boldsymbol{K}_2 \circ \boldsymbol{F}_2$ können wir die entstehende Matrix mit $\boldsymbol{f}$ multiplizieren. In dieser Darstellung ist der Speicheraufwand von $\boldsymbol{Q}$ somit gegeben durch*

$$N_{Q,St} = 2\,n^2$$

und der Aufwand für eine Operatorauswertung durch

$$N_{Q,Mv} \leq 5\,n^2.$$

3.8.2. Separable Kernfunktionen

In diesem Abschnitt stellen wir ein Verfahren vor, um die Implementierung des quadratischen Operators $\boldsymbol{Q}$ effizienter zu gestalten. Wir werden zeigen, dass die Matrizen $\boldsymbol{K}_1$ und $\boldsymbol{K}_2$ durch *Toeplitz-Matrizen* approximiert werden können, wenn die zugehörige Kernfunktion separable Eigenschaften besitzt.

Definition 3.8.7 (Toeplitz-Matrix): Eine Matrix $\boldsymbol{T} \in \mathbb{R}^{m\times n}$ heißt *Toeplitz-Matrix*, wenn die Elemente t_{ij} von $\boldsymbol{T}$ nur von der Differenz $i - j$ abhängen.

Notation 3.8.8: Für eine Toeplitz-Matrix $\boldsymbol{T} \in \mathbb{R}^{m\times n}$ führen wir folgende Kurzschreibweise ein:

$$\boldsymbol{T} = \begin{pmatrix} t_0 & t_1 & \ldots & t_{n-1} \\ t_{-1} & t_0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & t_1 \\ t_{-(m-1)} & \ldots & t_{-1} & t_0 \end{pmatrix} =: \operatorname{toep}\left(t_0, t_{-1}, \ldots, t_{-(m-1)} \,|\, t_1, \ldots, t_{n-1}\right).$$

Definition 3.8.9 (periodische Toeplitz-Matrix): Eine quadratische Matrix $\boldsymbol{C} \in \mathbb{R}^{n\times n}$ heißt *periodische Toeplitz-Matrix*, *zirkulante Matrix* oder auch *Zirkulante*, wenn die Einträge c_{ij} von $\boldsymbol{C}$ nur von der Differenz $i-j$ modulo n abhängen.

Notation 3.8.10: Sei $\boldsymbol{C} \in \mathbb{R}^{n\times n}$ eine Zirkulante, so setzen wir

$$\boldsymbol{C} = \begin{pmatrix} c_0 & c_{n-1} & \dots & c_1 \\ c_1 & c_0 & \ddots & \vdots \\ \vdots & \vdots & \ddots & c_{n-1} \\ c_{n-1} & c_{n-2} & \dots & c_0 \end{pmatrix} =: \operatorname{circ}(c_0, \dots, c_{n-1}).$$

Bemerkung 3.8.11: Seien $\boldsymbol{T}_1, \boldsymbol{T}_2 \in \mathbb{K}^{m\times n}$ zwei Toeplitz-Matrizen bzw. $\boldsymbol{C}_1, \boldsymbol{C}_2 \in \mathbb{K}^{n\times n}$ zwei periodische Toeplitz-Matrizen. Aus Notation 3.8.8 bzw. 3.8.10 geht unmittelbar hervor, dass die Summe $\boldsymbol{T}_1 + \boldsymbol{T}_2 \in \mathbb{K}^{m\times n}$ bzw. $\boldsymbol{C}_1 + \boldsymbol{C}_2 \in \mathbb{K}^{n\times n}$ und das Hadamard-Produkt $\boldsymbol{T}_1 \circ \boldsymbol{T}_2 \in \mathbb{K}^{m\times n}$ bzw. $\boldsymbol{C}_1 \circ \boldsymbol{C}_2 \in \mathbb{K}^{n\times n}$ ebenfalls (periodische) Toeplitz-Matrizen sind.

Bemerkung 3.8.12: Die Matrizen $\boldsymbol{F}_1$ und $\boldsymbol{F}_2$ aus Notation 3.8.5 sind gerade Toeplitz-Matrizen:

$$\begin{aligned} \boldsymbol{F}_1 &= \operatorname{toep}(0, f_1, \dots, f_{n-1} \,|\, 0, \dots, 0) \quad \text{und} \\ \boldsymbol{F}_2 &= \operatorname{toep}(f_1, \dots, f_n \,|\, 0, \dots, 0). \end{aligned}$$

Wir wollen die aus der Numerik bekannte Tatsache benutzen, vgl. hierzu etwa Cooley und Tukey [8] und Golub und Van Loan [21], dass eine Toeplitz-Matrix schnell mit einem Vektor multipliziert werden kann, indem man sie in eine periodische Toeplitz-Matrix einbettet und dann die *schnelle Fourier-Transformation (FFT)* anwendet. Wir verweisen dazu auf den Anhang A, wo wir noch einmal die wichtigsten Eigenschaften von Zirkulanten und Toeplitz-Matrizen zusammengetragen haben. Zur Ausführung der schnellen Multiplikation vgl. besonders Bemerkung A.0.8.

Definition 3.8.13 (Einbettungsoperator): Für die Realisierung und das bessere Verständnis der schnellen Multiplikation einer Toeplitz-Matrix $\boldsymbol{T} \in \mathbb{R}^{m\times n}$ mit einem Vektor $\boldsymbol{x} \in \mathbb{R}^n$ definieren wir den Einbettungsoperator E_{eb}. Dazu sei $\hat{m} := \max(m, n)$.

a) Es ist $E_{\text{eb}}[\boldsymbol{T}] \in \mathbb{R}^{2\hat{m}\times 2\hat{m}}$ definiert gemäß

$$\begin{aligned} E_{\text{eb}}[\boldsymbol{T}] &= E_{\text{eb}}\left[\operatorname{toep}(t_0, t_{-1}, \dots, t_{-(m-1)} \,|\, t_1, \dots, t_{n-1})\right] \\ &:= \operatorname{circ}(t_0, t_1, \dots, t_{-(n-1)}, 0, \dots, 0, t_{n-1}, \dots, t_1). \end{aligned}$$

b) Es ist $E_{\text{eb}}[\boldsymbol{t}] \in R^{2n}$ definiert gemäß

$$E_{\text{eb}}[\boldsymbol{t}] = E_{\text{eb}}[(x_1, \dots, x_n)^\top] := (x_1, \dots, x_n, 0, \dots, 0)^\top.$$

Bemerkung 3.8.14: Sei wieder $\hat{m} := \max(m,n)$, dann realisiert man die Multiplikation einer Toeplitz-Matrix $\boldsymbol{T} \in \mathbb{R}^{m\times n}$ mit einem Vektor $\boldsymbol{x} \in \mathbb{R}^n$, indem man den Vektor

$$\tilde{\boldsymbol{x}} := E_{\text{eb}}[\boldsymbol{T}] \cdot E_{\text{eb}}[\boldsymbol{x}] \in \mathbb{R}^{2\hat{m}}$$

mithilfe der FFT berechnet. Die ersten m-Einträge von $\tilde{\boldsymbol{x}}$ entsprechen dann dem Produkt $\boldsymbol{T}\cdot \boldsymbol{x}$.

Bemerkung 3.8.15: Die Ausführung der FFT ist am effektivsten, wenn sich der Dimensionsparameter n in der Form $n = 2^l$, $l \in \mathbb{N}$ schreiben lässt. Ist n keine Potenz von 2, so gibt es verschiedene Ansätze, der einfachste ist die Einbettung in eine größere Matrix gemäß Bemerkung A.0.7, was im schlimmsten Fall eine Verdopplung des Aufwandes bedeutet.

Um Fallunterscheidungen vorzubeugen, benutzten wir nachfolgende Definition.

Definition 3.8.16: Sei $m \in \mathbb{N}$, dann existiert ein $p \in \mathbb{N}$, so dass $2^p \leq m < 2^{p+1}$, mit $p \in \mathbb{N}_0$. Damit definieren wir den Operator $[.]_2 : \mathbb{N} \to \mathbb{N}$ gemäß

$$[m]_2 = \begin{cases} m & \text{für} \quad m = 2^p \\ 2^{p+1} & \text{sonst.} \end{cases}$$

Folgerung 3.8.17: *Für den Speicheraufwand bzw. den Aufwand der Matrix-Vektor-Multiplikation einer Toeplitz-Matrix $\boldsymbol{T} \in \mathbb{R}^{m\times n}$ mittels FFT gilt*

$$N_{T,St}(m,n) \leq m+n \leq 2\max(m,n) \quad \textit{bzw.}$$

$$N_{T,Mv}(m,n) \leq 9[\max(m,n)]_2 \log_2[\max(m,n)]_2 + 13[\max(m,n)]_2,$$

wenn wir $[\max(m,n)]_2 \geq 4$ *voraussetzen gilt außerdem*

$$N_{T,Mv}(m,n) \leq 16\,[\max(m,n)]_2 \log_2[\max(m,n)]_2.$$

Beweis: Sei $\hat{m} := \max(m,n)$. Um eine Toeplitz-Matrix $\boldsymbol{T} \in \mathbb{R}^{m\times n}$ eindeutig zu beschreiben, genügen aufgrund ihrer Struktur $m + (n-1) \leq m+n \leq 2\hat{m}$ Einträge, vgl. etwa Notation 3.8.8. Der Aufwand der Multiplikation einer Zirkulanten $\boldsymbol{C} \in \mathbb{R}^{l\times l}$ mit einem Vektor ist nach Folgerung A.0.6 aus Anhang A gegeben durch $9/2\, l \log_2 l + 2l$, falls $l = 2^p$, $p \in \mathbb{N}$. Für eine Matrix-Vektor-Multiplikation betten wir $\boldsymbol{T}$ in eine $\mathbb{R}^{2\hat{m}\times 2\hat{m}}$ Zirkulante ein und haben somit eine Aufwand von

$$9[\hat{m}]_2 \log_2 2[\hat{m}]_2 + 4[\hat{m}]_2 = 9[\hat{m}]_2 \log_2[\hat{m}]_2 + 13[\hat{m}]_2 \leq 16[\hat{m}]_2 \log_2[\hat{m}]_2,$$

wegen $7[\hat{m}]_2 \log_2[\hat{m}]_2 \geq 13[\hat{m}]_2$ für $[\hat{m}]_2 \geq 4$. □

Um $\boldsymbol{Q}$ mit Hilfe von Toeplitz-Matrizen darzustellen, nehmen wir zunächst wieder an, dass die Kernfunktion κ gemäß Definition 3.5.1 bereits separiert vorliegt, d.h. in der Form

$$\kappa(x,y) = \sum_{l=1}^{k_{\text{sep}}} \Phi_l(x)\Psi_l(y). \tag{3.86}$$

Für die Möglichkeiten, die sich aus dieser Darstellung ergeben, betrachten wir exemplarisch die Einträge von $\boldsymbol{K}_1$ auf der Diagonalen. Diese sind durch (3.85) gegeben und besitzen bei separabler Kernfunktion die Gestalt

$$K_{ii} = \sum_{l=1}^{k_{\text{sep}}} \int_{\xi_{i-1}}^{\xi_i} \int_{\xi_{i-1}}^{x} \Phi_l(x-y)\Psi_l(y)\, dy\, dx. \tag{3.87}$$

Die Idee ist nun, die Integrale, die sich hinter den Einträgen von $\boldsymbol{K}_1$ und $\boldsymbol{K}_2$ verbergen, durch Quadraturverfahren zu ersetzen und anschließend die separable Darstellung (3.86) auszunutzen. Um einen Vorteil aus der Translationsinvarianz von $\Phi_l(x-y)$ ziehen zu können, benötigen wir dabei ein Quadraturverfahren, welches auf äquidistanten Stützstellen beruht. Wir benutzen hier die zweidimensionale *Trapezregel*, welche für eine genügend glatte Kernfunktionen und kleine Integrationsgebiete gute Ergebnisse liefert.

Gemäß Abschnitt 3.8.1 integrieren wir nicht über Quadrate, sondern über Dreiecke, daher müssen wir die Trapezmethode entsprechend modifizieren. Eine ausführliche Darstellung dazu findet man im Anhang C.

Lemma 3.8.18 (TrapezmethodeI): *Es gelte $\kappa \in C^2([\xi_{i-j}, \xi_{i-j+1}]\times[\xi_{j-1}, \xi_j])$. Dann folgt:*

$$\begin{aligned}
&\int_{\xi_{i-1}}^{\xi_i} \int_{\xi_{j-1}}^{x-\xi_{i-j}} \kappa(x-y,y)\, dy\, dx = \int_{\xi_{i-j}}^{\xi_{i-j+1}} \int_{\xi_{j-1}}^{\xi_i - x} \kappa(x,y)\, dy\, dx \\
&= \frac{h^2}{24}\Big[5\,\kappa(\xi_{i-j},\xi_{j-1}) + 3\,\kappa(\xi_{i-j+1},\xi_{j-1}) + 3\,\kappa(\xi_{i-j},\xi_j) + \kappa(\xi_{i-j+1},\xi_j)\Big] \\
&\quad - \frac{h^4}{24}\left[\frac{\partial^2}{\partial x^2}\kappa(\xi,\zeta) + \frac{\partial^2}{\partial y^2}\kappa(\tilde{\xi},\tilde{\zeta})\right],
\end{aligned} \tag{3.88}$$

für $\xi, \tilde{\xi} \in [\xi_{i-j}, \xi_{i-j+1}]$ und $\zeta, \tilde{\zeta} \in [\xi_{j-1}, \xi_j]$, wobei der Integrationsbereich gerade dem Dreieck $\tilde{D}$ aus Abbildung 3.14 a) entspricht. Der erste Term der rechten Seite beschreibt dabei die modifizierte Trapezregel*, der zweite Term stellt das Fehlerglied dar.*

Beweis: Dies ist ein Spezialfall von Lemma C.0.3, vgl. im Anhang C die Beweise zu Lemma C.0.2 und Lemma C.0.3. □

Eine analoge Darstellung erhalten wir für Integrale über dem Dreieck D gemäß Abbildung 3.14 b).

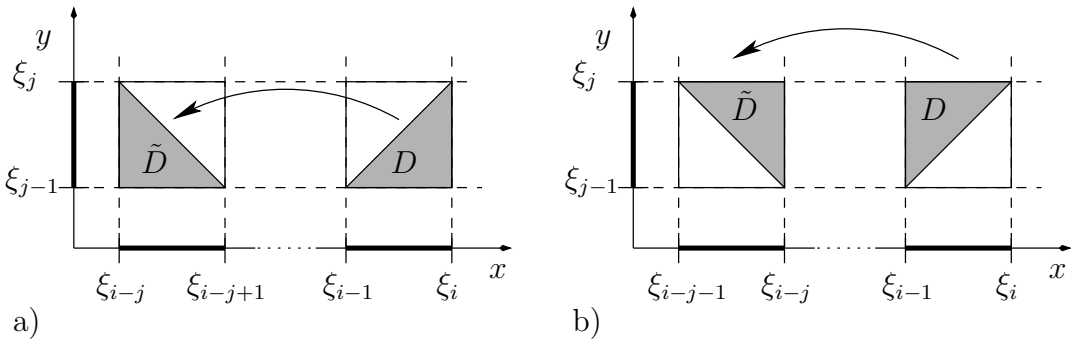


Abbildung 3.14.: Transformation des Integrationsgebietes.

Lemma 3.8.19 (Trapezmethode II): *Es gelte* $\kappa \in C^2([\xi_{i-j-1}, \xi_{i-j}] \times [\xi_{j-1}, \xi_j])$. *Dann folgt:*

$$\begin{aligned}
&\int_{\xi_{i-1}}^{\xi_i} \int_{x-\xi_{i-j}}^{\xi_j} \kappa(x-y,y)\,dy\,dx = \int_{\xi_{i-j-1}}^{\xi_{i-j}} \int_{\xi_{i-1}-x}^{\xi_j} \kappa(x,y)\,dy\,dx \\
&= \frac{h^2}{24}\Big[\kappa(\xi_{i-j-1},\xi_{j-1}) + 3\,\kappa(\xi_{i-j},\xi_{j-1}) + 3\,\kappa(\xi_{i-j-1},\xi_j) + 5\,\kappa(\xi_{i-j},\xi_j)\Big] \qquad (3.89)\\
&\quad - \frac{h^4}{24}\left[\frac{\partial^2}{\partial x^2}\kappa(\xi,\zeta) + \frac{\partial^2}{\partial y^2}\kappa(\tilde{\xi},\tilde{\zeta})\right],
\end{aligned}$$

für $\xi, \tilde{\xi} \in [\xi_{i-j-1}, \xi_{i-j}]$ *und* $\zeta, \tilde{\zeta} \in [\xi_{j-1}, \xi_j]$, *wobei der Integrationsbereich gerade dem Dreieck* $\tilde{D}$ *aus Abbildung 3.14 b) entspricht. Der erste Term der rechten Seite beschreibt dabei die* modifizierte Trapezregel, *der zweite Term stellt das Fehlerglied dar.*

Beweis: Dies ist ein Spezialfall von Lemma C.0.3, vgl. im Anhang C die Beweise zu Lemma C.0.2 und Lemma C.0.3. □

Notation 3.8.20: Wir führen die folgenden abkürzenden Schreibweisen ein:

$$\Phi_{l,i} := \Phi_l(\xi_i) \quad \text{und} \quad \Psi_{l,i} := \Psi_l(\xi_i).$$

Mithilfe der Trapezregel können wir die Matrix $\boldsymbol{K}_1$ bzw. $\boldsymbol{K}_2$ (vgl. (3.85)) durch die angenäherte Matrix $\tilde{\boldsymbol{K}}_1$ bzw. $\tilde{\boldsymbol{K}}_2$ ausdrücken. Mit Notation 3.8.20 gilt gemäß Lemma 3.8.18, Lemma 3.8.19 sowie der separablen Darstellung der Kernfunktion κ gemäß (3.86) für die Einträge der Matrix $\tilde{\boldsymbol{K}}_1$ bzw. $\tilde{\boldsymbol{K}}_2$:

$$(\tilde{K}_1)_{ij} := \frac{h^2}{24}\sum_{l=1}^{k_{\text{sep}}} 5\,\Phi_{l,i-j}\Psi_{l,j-1} + 3\,\Phi_{l,i-j+1}\Psi_{l,j-1} + 3\,\Phi_{l,i-j}\Psi_{l,j} + \Phi_{l,i-j+1}\Psi_{l,j} \quad (3.91a)$$

für $i \geq j$ und $\tilde{K}^1_{ij} = 0$ sonst bzw.

$$(\tilde{K}_2)_{ij} := \frac{h^2}{24}\sum_{l=1}^{k_{\text{sep}}} \Phi_{l,i-j-1}\Psi_{l,j-1} + 3\,\Phi_{l,i-j}\Psi_{l,j-1} + 3\,\Phi_{l,i-j-1}\Psi_{l,j} + 5\,\Phi_{l,i-j}\Psi_{l,j} \quad (3.91b)$$

für $i > j$ und $(\tilde{K}_2)_{ij} = 0$ sonst, für $i, j = 1, \dots, n$. Wir ersetzen nun den diskreten Operator $\boldsymbol{Q}$ durch durch einen approximierten Operator.

Definition 3.8.21: Für die approximative Darstellung des Systems (3.84) definieren wir den *approximierten Operator* $\boldsymbol{Q}_{T(k_{\text{sep}})} : \mathbb{R}^n \to \mathbb{R}^n$ vom Grad k_{sep} gemäß

$$\boldsymbol{Q}_{T(k_{\text{sep}})} \boldsymbol{f} := \left(\tilde{\boldsymbol{K}}_1 \circ \boldsymbol{F}_1 + \tilde{\boldsymbol{K}}_2 \circ \boldsymbol{F}_2\right) \cdot \boldsymbol{f}.$$

Ist der Bezug eindeutig, so schreiben wir auch nur $\boldsymbol{Q}_T$.

Unter der Annahme, dass jeweils die zweiten Ableitungen von κ verschwinden, stimmt $\boldsymbol{Q}_T$ sogar mit $\boldsymbol{Q}$ überein.

Wir werden im Folgenden zeigen, dass wir den Operator $\boldsymbol{Q}_T$ in günstiger Weise darstellen können, bezogen auf Auswertung und Speicherbedarf. Dazu führen wir zunächst die folgenden Matrizen und Vektoren ein.

Definition 3.8.22: Es sei der Vektor $\boldsymbol{x} = (x_1, \dots, x_n)^\top \in \mathbb{R}^n$ gegeben, dann definieren wir die *Spaltenmatrix* $Sp_m[\boldsymbol{x}] \in \mathbb{R}^{m \times n}$ gemäß

$$Sp_m[\boldsymbol{x}] = \begin{pmatrix} x_1 & x_2 & \dots & x_n \\ x_1 & x_2 & \dots & x_n \\ \vdots & \vdots & \vdots & \vdots \\ x_1 & x_2 & \dots & x_n \end{pmatrix}.$$

Bemerkung 3.8.23: Es sei $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ eine Matrix und $\boldsymbol{a} = (a_1, \dots, a_n)^\top$, $\boldsymbol{x} = (x_1, \dots, x_n)^\top \in \mathbb{R}^n$ zwei Vektoren, dann gilt

$$\left(\boldsymbol{A} \circ Sp_m[\boldsymbol{a}]\right) \boldsymbol{x} = \boldsymbol{A} \left(\boldsymbol{a} \circ \boldsymbol{x}\right).$$

Beweis: Seien mit a_{ij}, $i = 1, \dots, m$, $j = 1, \dots, n$ die Einträge von $\boldsymbol{A}$ bezeichnet, dann gilt

$$\left(\boldsymbol{A} \circ Sp_m[\boldsymbol{a}]\right) \boldsymbol{x} = \begin{pmatrix} \sum_{j=1}^n a_{1j} a_j x_j \\ \vdots \\ \sum_{j=1}^n a_{mj} a_j x_j \end{pmatrix} = \boldsymbol{A} \left(\boldsymbol{a} \circ \boldsymbol{x}\right).$$

□

Notation 3.8.24: Wir definieren die folgenden Toeplitz-Matrizen, wobei wir die Bezeichnungen aus den Notationen 3.8.20 und 3.8.8 benutzen:

$$\begin{aligned}
\boldsymbol{T}_{01}^l &:= \text{toep}\left(0, \Phi_{l,0}, \dots, \Phi_{l,n-2} \,|\, 0, \dots, 0\right), \\
\boldsymbol{T}_{02}^l &:= \text{toep}\left(0, \Phi_{l,1}, \dots, \Phi_{l,n-1} \,|\, 0, \dots, 0\right), \\
\boldsymbol{T}_{03}^l &:= \text{toep}\left(\Phi_{l,0}, \dots, \Phi_{l,n-1} \,|\, 0, \dots, 0\right) \quad \text{und} \\
\boldsymbol{T}_{04}^l &:= \text{toep}\left(\Phi_{l,1}, \dots, \Phi_{l,n} \,|\, 0, \dots, 0\right).
\end{aligned}$$

Notation 3.8.25: Desweiteren definieren wir die Toeplitz-Matrizen (vgl. Bemerkung 3.8.11)

$$\begin{aligned}
\boldsymbol{T}^l_{11} &:= \boldsymbol{T}^l_{01} + 3\,\boldsymbol{T}^l_{02},\\
\boldsymbol{T}^l_{12} &:= 5\,\boldsymbol{T}^l_{03} + 3\,\boldsymbol{T}^l_{04},\\
\boldsymbol{T}^l_{13} &:= 3\,\boldsymbol{T}^l_{01} + 5\,\boldsymbol{T}^l_{02},\\
\boldsymbol{T}^l_{14} &:= 3\,\boldsymbol{T}^l_{03} + \boldsymbol{T}^l_{04}.
\end{aligned}$$

Notation 3.8.26: Es seien die Vektoren $\boldsymbol{t}^l_1$ und $\boldsymbol{t}^l_2$ definiert gemäß

$$\boldsymbol{t}^l_1 := (\Psi_{l,0}, \Psi_{l,1}, \dots, \Psi_{l,n-1})^\top \quad \text{und} \quad \boldsymbol{t}^l_2 := (\Psi_{l,1}, \Psi_{l,2}, \dots, \Psi_{l,n})^\top.$$

Mit Bemerkung 3.8.23 sowie den Notationen 3.8.24 und 3.8.26 bzw. 3.8.25 können wir $\tilde{\boldsymbol{K}}_1$ und $\tilde{\boldsymbol{K}}_2$ darstellen gemäß

$$\begin{aligned}
\tilde{\boldsymbol{K}}_1 &= \frac{h^2}{24}\sum_{l=1}^{k_{\text{sep}}} 5\boldsymbol{T}^l_{03} \circ Sp[\boldsymbol{t}^l_1] + 3\boldsymbol{T}^l_{04} \circ Sp[\boldsymbol{t}^l_1] + 3\boldsymbol{T}^l_{03} \circ Sp[\boldsymbol{t}^l_2] + \boldsymbol{T}^l_{04} \circ Sp[\boldsymbol{t}^l_2]\\
&= \frac{h^2}{24}\sum_{l=1}^{k_{\text{sep}}} (5\boldsymbol{T}^l_{03} + 3\boldsymbol{T}^l_{04}) \circ Sp[\boldsymbol{t}^l_1] + (3\boldsymbol{T}^l_{03} + \boldsymbol{T}^l_{04}) \circ Sp[\boldsymbol{t}^l_2]\\
&= \frac{h^2}{24}\sum_{l=1}^{k_{\text{sep}}} \boldsymbol{T}^l_{12} \circ Sp[\boldsymbol{t}^l_1] + \boldsymbol{T}^l_{14} \circ Sp[\boldsymbol{t}^l_2]
\end{aligned}$$

und

$$\begin{aligned}
\tilde{\boldsymbol{K}}_2 &= \frac{h^2}{24}\sum_{l=1}^{k_{\text{sep}}} \boldsymbol{T}^l_{01} \circ Sp[\boldsymbol{t}^l_1] + 3\boldsymbol{T}^l_{02} \circ Sp[\boldsymbol{t}^l_1] + 3\boldsymbol{T}^l_{01} \circ Sp[\boldsymbol{t}^l_2] + 5\boldsymbol{T}^l_{02} \circ Sp[\boldsymbol{t}^l_2]\\
&= \frac{h^2}{24}\sum_{l=1}^{k_{\text{sep}}} (\boldsymbol{T}^l_{01} + 3\boldsymbol{T}^l_{02}) \circ Sp[\boldsymbol{t}^l_1] + (3\boldsymbol{T}^l_{01} + 5\boldsymbol{T}^l_{02}) \circ Sp[\boldsymbol{t}^l_2]\\
&= \frac{h^2}{24}\sum_{l=1}^{k_{\text{sep}}} \boldsymbol{T}^l_{11} \circ Sp[\boldsymbol{t}^l_1] + \boldsymbol{T}^l_{13} \circ Sp[\boldsymbol{t}^l_2].
\end{aligned}$$

Damit können wir $\boldsymbol{Q}_T\boldsymbol{f}$ darstellen gemäß

$$\begin{aligned}
&\frac{h^2}{24}\sum_{l=1}^{k_{\text{sep}}} \Big[\big[\boldsymbol{T}^l_{12} \circ Sp_n[\boldsymbol{t}^l_1] + \boldsymbol{T}^l_{14} \circ Sp_n[\boldsymbol{t}^l_2]\big] \circ \boldsymbol{F}_1 + \big[\boldsymbol{T}^l_{11} \circ Sp_n[\boldsymbol{t}^l_1] + \boldsymbol{T}^l_{13} \circ Sp_n[\boldsymbol{t}^l_2]\big] \circ \boldsymbol{F}_2\Big]\boldsymbol{f}\\
&= \frac{h^2}{24}\sum_{l=1}^{k_{\text{sep}}} \big[\boldsymbol{T}^l_{12} \circ \boldsymbol{F}_1 + \boldsymbol{T}^l_{11} \circ \boldsymbol{F}_2\big] \circ Sp_n[\boldsymbol{t}^l_1]\boldsymbol{f} + \big[\boldsymbol{T}^l_{14} \circ \boldsymbol{F}_1 + \boldsymbol{T}^l_{13} \circ \boldsymbol{F}_2\big] \circ Sp_n[\boldsymbol{t}^l_2]\boldsymbol{f}.
\end{aligned}$$

Mit Bemerkung 3.8.23 stellen wir $\boldsymbol{Q}_T\boldsymbol{f}$ schließlich in der Form dar

$$\boldsymbol{Q}_T\boldsymbol{f} = \frac{h^2}{24}\sum_{l=1}^{k_{\text{sep}}} \underbrace{\big[\boldsymbol{T}^l_{11} \circ \boldsymbol{F}_1 + \boldsymbol{T}^l_{12} \circ \boldsymbol{F}_2\big]}_{=:\,\boldsymbol{T}^l_1} \cdot \big[\boldsymbol{t}^l_1 \circ \boldsymbol{f}\big] + \underbrace{\big[\boldsymbol{T}^l_{13} \circ \boldsymbol{F}_1 + \boldsymbol{T}^l_{14} \circ \boldsymbol{F}_2\big]}_{=:\,\boldsymbol{T}^l_2} \cdot \big[\boldsymbol{t}^l_2 \circ \boldsymbol{f}\big], \tag{3.92}$$

wobei alle Matrizen Toeplitz-Struktur besitzen, vgl. Bemerkung 3.8.12. Bei der Berechnung des Systems (3.92) gehen wir so vor, dass wir für $l = 1, \dots, k_{\text{sep}}$ zunächst $\boldsymbol{T}_{11}^l, \dots, \boldsymbol{T}_{14}^l$ entsprechend mit $\boldsymbol{F}_1$ oder $\boldsymbol{F}_2$ Hadamard-multiplizieren. Dann können wir die Matrizen $\boldsymbol{T}_1^l$ und $\boldsymbol{T}_2^l$ berechnen, welche gemäß Bemerkung 3.8.11 ebenfalls Toeplitz-Struktur besitzen. Außerdem werden $\boldsymbol{t}_1^l$ und $\boldsymbol{t}_2^l$ mit $\boldsymbol{f}$ Hadamard-multipliziert. Die resultierenden Vektoren multiplizieren wir dann mithilfe der FFT mit $\boldsymbol{T}_1^l$ und $\boldsymbol{T}_2^l$ und addieren die entstehenden Vektoren.

Folgerung 3.8.27: *Sei der Operator $\boldsymbol{Q}_{T(k_{\text{sep}})}$ gemäß (3.92) vorgelegt. Dann gilt:*

a) der Speicheraufwand für den Operator $\boldsymbol{Q}_T$ ist gegeben durch

$$N_{Q_T,St} \leq 10\, k_{\text{sep}} n,$$

b) der Aufwand der Operatorauswertung ist gegeben durch

$$N_{Q_T,Op} \leq k_{\text{sep}}(18[n]_2 \log_2[n]_2 + 41[n]_2).$$

Beweis: a) Für den Operator $\boldsymbol{Q}_T$ müssen wir für $i = 1, \dots, k_{\text{sep}}$ die Toeplitz-Matrizen $\boldsymbol{T}_{11}^l, \dots, \boldsymbol{T}_{14}^l$ speichern und außerdem die Vektoren $\boldsymbol{t}_1^l$ und $\boldsymbol{t}_2^l$. Nach Folgerung 3.8.17 beträgt der Speicheraufwand für die Toeplitz-Matrizen $8k_{\text{sep}}n$. Für die Vektoren $\boldsymbol{t}_1^l$ und $\boldsymbol{t}_2^l$ müssen wir jeweils $k_{\text{sep}}n$ Einträge speichern, somit haben wir einen Gesamtspeicherbedarf von $10k_{\text{sep}}n$.

b) Für $i = 1, \dots, k_{\text{sep}}$ müssen wir nacheinander die folgenden Operationen mit dem entsprechenden Aufwand durchführen:

1. Das Update der vier Toeplitz-Matrizen $\boldsymbol{T}_{11}^l, \dots, \boldsymbol{T}_{14}^l$ und der beiden Vektoren $\boldsymbol{t}_1^l$ und $\boldsymbol{t}_2^l$ benötigt gemäß Bemerkung 3.8.11 $10\,n$ Operationen.
2. Die Addition jeweils zwei der Toeplitz-Matrizen nach dem Update um die Toeplitz-Matrizen $\boldsymbol{T}_1^l$ und $\boldsymbol{T}_2^l$ zu berechnen, benötigt nach Bemerkung 3.8.11 insgesamt $4\,n$ Operationen.
3. Die Ausführung der FFT von $\boldsymbol{T}_1^l$ und $\boldsymbol{T}_2^l$ benötigt nach Folgerung 3.8.17 je $9[n]_2 \log_2[n]_2 + 13[n]_2$, d.h. insgesamt $18[n]_2 \log_2[n]_2 + 26[n]_2$ Operationen.
4. Die Addition der resultierenden Vektoren miteinander benötigt n Operationen.

Somit benötigen wir insgesamt $\leq k_{\text{sep}}(18[n]_2 \log_2[n]_2 + 41[n]_2)$ Operationen.

□

3.8.3. Quadratischer Operator und $\mathcal{H}$-Matrix-Kalkül

Die globale Approximation des Operators $\boldsymbol{Q}$ durch $\boldsymbol{Q}_T$ in ausreichender Genauigkeit wird i.A. nicht möglich sein. Daher benutzen wir wieder die Idee des $\mathcal{H}$-Matrix-Kalküls, nur dass jetzt die zulässigen Blöcke keine R(k)-Matrizen darstellen, sondern lokal die Struktur des Operators $\boldsymbol{Q}_{T(k)}$ aufweisen. Dazu müssen wir zunächst entsprechende Zulässigkeitsbedingungen für die von uns betrachteten Kerne finden.

Wir beginnen unsere Untersuchung wieder am modifizierten Smoluchowski-Kern (3.49). Wenn wir auf diesen die Faltung anwenden, besitzt er die Form

$$\kappa(x-y,y) = \frac{x^2}{c_\mathrm{s} + (x-y)y},$$

wobei $\kappa \equiv 0$ gilt, für $y > x$. Wenn c_s gegen Null strebt, haben wir Singularitäten für $y \to 0$ und $x \to y$. Wie im linearen Fall formulieren wir ein Zulässigkeitskriterium, welches gerade die Singularitäten von der Kernentwicklung ausschließt.

Lemma 3.8.28: *Sei f eine Funktion auf $[0,1]$ mit Singularitäten höchstens in 0 und im Punkt $p \in (0,1]$. Es lasse sich f komplex fortsetzen, so dass f jeweils auf den Gebieten $G_1 = (0,p) \times [-1,1] \subset \mathbb{C}$ und $G_2 = (p,2] \times [-1,1] \subset \mathbb{C}$ holomorph ist. Dann lässt sich f auf jedem Intervall $[a,b] \in (0,1]$, $p \notin [a,b]$ in eine Taylor-Reihe entwickeln. Sei weiter $x_0 := (a+b)/2$ und $C \subset \mathbb{C}$ eine Kreiskurve mit dem Mittelpunkt x_0 und dem Radius $\delta > (b-a)/2$.*

a) Falls für $p < a$ die Intervallgrenzen die Bedingung

$$(b-a) < 2\eta\,(a-p) \quad \textit{für} \quad \eta \in \left(\frac{b-a}{2(a-p)}, 1\right) \tag{3.93}$$

erfüllen, dann können wir δ so wählen, dass folgende Fehlerabschätzung gilt:

$$|f(x) - \Pi_T^k(f)[x]| \le \|f\|_{\infty,C} \frac{1}{2^{k-1}}. \tag{3.94}$$

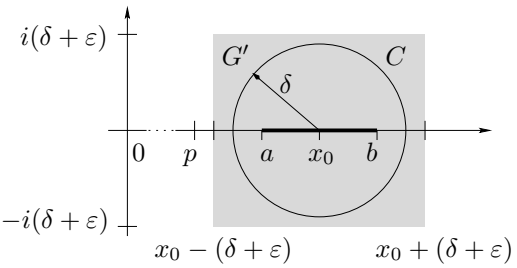
b) Falls für $0 < a$ und $b < p$ die Intervallgrenzen die Bedingungen

$$(b-a) < 2\eta\,(p-b) \quad \textit{für} \quad \eta \in \left(\frac{b-a}{2(p-b)}, 1\right) \quad \textit{und} \tag{3.95a}$$

$$(b-a) < 2\eta\,a \quad \textit{für} \quad \eta \in \left(\frac{b-a}{2a}, 1\right) \tag{3.95b}$$

erfüllen, können wir δ so wählen, dass ebenfalls die Fehlerabschätzung (3.94) gilt.

Beweis: Sei $[a,b]$ zunächst beliebig. Wegen $0, p \notin [a,b]$ gilt genau einer der Fälle $0 < a < b < p$ bzw. $p < a < b$, vgl. b) bzw. a). Wir zeigen zunächst a). Wir wählen wieder einen Kreis C mit dem Mittelpunkt x_0 und dem Radius δ gemäß Voraussetzung. Außerdem muss $\delta < (a-p)+(b-a)/2$ gelten, vgl. auch nebenstehende Abbildung, da sonst C die Singularität in p mit einschließen würde. Wir stellen δ in der Form $\delta + 2\varepsilon = (a-p) + (b-a)/2$ dar, damit ist C in dem Gebiet



$$G' = [x_0 - (\delta+\varepsilon), x_0 + (\delta+\varepsilon)] \times [-(\delta+\varepsilon), \delta+\varepsilon]$$

vollständig enthalten. Wegen $G' \subset G_2$ existiert eine entsprechende holomorphe Fortsetzung von f, so dass wir Lemma 3.7.18 anwenden können, somit existiert im Fall a) eine Taylor-Entwicklung von f auf dem Intervall $[a,b]$. Sei nun zusätzlich Bedingung (3.93) erfüllt, dann gilt

$$\frac{b-a}{2} + \eta\frac{b-a}{2} < \eta(a-p) + \eta\frac{b-a}{2} \quad \text{bzw.} \quad \frac{b-a}{2}(1+\eta) < \eta\left[\frac{b-a}{2} + (a-p)\right]$$

und damit wegen $|x - x_0| \leq (b-a)/2$

$$|x - x_0| < \frac{\eta}{1+\eta}\left[\frac{b-a}{2} + (a-p)\right], \quad \forall x \in [a,b]. \tag{3.96}$$

Wir können nun δ in der alternativen Form $\delta = \eta'[(a-p) + (b-a)/2]$ darstellen, wobei aus den Bedingungen an δ

$$(b-a)/2 < \delta = \eta'\,[\,(a-p) + (b-a)/2\,] < (a-p) + (b-a)/2$$

folgt, dass $\eta' \in \left(\frac{b-a}{a+b-2p}, 1\right)$. Wir setzen dann

$$\eta' = 2\frac{\eta}{1+\eta}, \quad \text{mit} \quad \eta \in \left(\frac{b-a}{(b+3a)-4p}, 1\right)$$

und erhalten aus (3.96)

$$|x - x_0| \leq \frac{1}{2}\delta \quad \text{bzw.} \quad \frac{|x-x_0|}{\delta} \leq \frac{1}{2}, \quad \forall x \in [a,b].$$

Damit ergibt sich mithilfe von (3.59) aus dem Lemma 3.7.18 die gesuchte Abschätzung (3.94) für den Fall a). Wegen Bedingung (3.93) gilt insbesondere $(b-a)/2(a-p) < \eta$ und somit folgt aus $(b-a)/(b+3a-4p) < (b-a)/2(a-p)$, dass

$$\eta \in \left(\frac{b-a}{2(a-p)}, 1\right).$$

Für den Fall b) kombinieren wir die Vorgehensweisen aus dem Lemma 3.7.19 und dem Fall a), um die Existenz der Taylor-Entwicklung zu beweisen. Mithilfe von Lemma 3.7.19 und Fall a) finden wir auch ein entsprechendes δ, so dass die gewünschte Abschätzung (3.94) aus den Bedingungen (3.95) folgt. □

Bemerkung 3.8.29: Wir wollen das Lemma 3.8.28 für Funktionen in zwei Variablen verallgemeinern, wobei wir auf die Approximation der gefalteten Aggregationskerne κ hinarbeiten. Sei nun $[a,b] \times [c,d]$ ein zu betrachtendes Entwicklungsgebiet. Wir fordern für dessen Zulässigkeit, dass für $\kappa(x,y)$ stets $x \neq y$ und $y \neq 0$ gilt, für alle $x \in [a,b]$, $y \in [c,d]$, gemäß den Voraussetzungen von Lemma 3.8.28. Dann können wir die Tatsache ausnutzen, dass aufgrund des Volterra-Integrals stets

$$\kappa(x,y) \equiv 0 \quad \text{für} \quad y > x \tag{3.97}$$

gilt, indem wir zwei Fälle unterscheiden.

a) Entwickeln wir auf dem Intervall $[a,b]$ und halten ein $\hat{y}$ fest, dann gilt $\hat{y} \notin [a,b]$, wegen $x \neq \hat{y}$ für alle $x \in [a,b]$. Von den zwei möglichen Fällen $\hat{y} < a$ und $\hat{y} > b$ ist folglich nur der erste von Bedeutung, da im zweiten $\kappa \equiv 0$ gilt, wegen (3.97).

b) Entwickeln wir hingegen in y auf dem Intervall $[c,d]$ und halten $\hat{x}$ fest, so ist wegen (3.97) nur der Fall $\hat{x} > d$ interessant.

Folgerung 3.8.30: *Sei $f(x,y)$ eine Funktion auf $[0,1] \times [0,1] \subset \mathbb{R}^2$, welche höchstens für $y = 0$ und $x = y$ Singularitäten besitzt.*

a) Es lasse sich $f(.,\hat{y})$ für festes $\hat{y} \in (0,1]$ holomorph fortsetzen auf $(\hat{y},2] \times [-1,1] \subset \mathbb{C}$. Dann lässt sich $f(.,\hat{y})$ auf jedem Intervall $[a,b] \subset (\hat{y},1]$ in eine Taylor-Reihe entwickeln. Sei weiter $x_0 := (a+b)/2$ und $C \subset \mathbb{C}$ ein Kreis mit dem Mittelpunkt x_0 und dem Radius $\delta > (b-a)/2$. Wenn a und b außerdem der Bedingung

$$(b-a) < 2\eta(a-\hat{y}) \quad \textit{für} \quad \eta \in \left(\frac{b-a}{2(a-\hat{y})}, 1\right) \tag{3.98}$$

genügen, dann können wir δ so wählen, dass die abgeschnittene Taylor-Reihe die Funktion $f(.,\hat{y})$ auf $[a,b]$ mit dem Fehler

$$|f(x,\hat{y}) - \Pi_T^k[f(.,\hat{y})](x)| \leq \|f(.,\hat{y})\|_{\infty,C} \frac{1}{2^{k-1}} \tag{3.99}$$

approximiert.

b) Es lasse sich $f(\hat{x},.)$ für festes $\hat{x} \in [0,1]$ holomorph fortsetzen auf $(0,\hat{x}) \times [-1,1] \subset \mathbb{C}$. Dann lässt sich $f(\hat{x},.)$ auf jedem Intervall $[a,b] \subset (0,\hat{x})$ in eine Taylor-Reihe entwickeln. Sei $y_0 := (a+b)/2$ und $\hat{x} \in [0,1]$ fest, weiter sei $C \subset \mathbb{C}$ ein Kreis mit dem Mittelpunkt y_0 und dem Radius δ. Wenn zusätzlich für a und b die Bedingungen

$$\begin{aligned} &(b-a) < 2\eta(\hat{x}-b) \quad \textit{für} \quad \eta \in \left(\frac{b-a}{2(\hat{x}-b)}, 1\right) \\ &(b-a) < 2\eta\, a \quad \textit{für} \quad \eta \in \left(\frac{b-a}{2a}, 1\right) \end{aligned} \tag{3.100}$$

erfüllt sind, dann können wir δ so wählen, dass die abgeschnittene Taylor-Reihe die Funktion $f(\hat{x},.)$ auf $[a,b]$ mit einem Fehler

$$|f(\hat{x},y) - \Pi_T^k[f(\hat{x},.)](y)| \leq \|f(\hat{x},.)\|_{\infty,C} \frac{1}{2^{k-1}} \tag{3.101}$$

approximiert.

Beweis: Direkte Folgerung aus dem Lemma 3.8.28, wobei für a) bzw. b) jeweils $\hat{x}$ bzw. $\hat{y}$ die Rolle von p übernimmt. □

Lemma 3.8.31: *Sei der modifizierte Smoluchowski-Kern (3.49) als Faltungskern vorgelegt, d.h. in der Form*

$$\kappa(x,y) = \frac{x^2}{c + (x-y)y}, \quad c > 0,$$

auf dem Definitionsbereich $[0, x_{\max}] \times [0, x_{\max}] \subset [0,1] \times [0,1]$.

a) Die Funktion $\kappa(.,\hat{y})$ *lässt sich auf jedem Intervall* $[a,b] \subset (\hat{y}, x_{\max}]$ *für ein festes* $\hat{y} \in (0, x_{\max}]$ *in eine Taylor-Reihe entwickeln. Gilt außerdem die Bedingung (3.98) aus Folgerung 3.8.30, so kann man den Approximationsfehler der abgeschnittenen Taylor-Reihe abschätzen durch*

$$\left| \kappa(x,\hat{y}) - \Pi_T^k[\kappa(.,\hat{y})](x) \right| \leq \frac{5x_{\max}^2}{c} \frac{1}{2^{k-1}}.$$

b) Die Funktion $\kappa(\hat{x},.)$ *lässt sich auf jedem Intervall* $[a,b] \subset (0,\hat{x})$ *für ein festes* $\hat{x} \in [0, x_{\max}]$ *in eine Taylor-Reihe entwickeln. Gilt außerdem die Bedingung (3.100) aus Folgerung 3.8.30, so kann man den Approximationsfehler der abgeschnittenen Taylor-Reihe abschätzen durch*

$$\left| \kappa(\hat{x},y) - \Pi_T^k[\kappa(\hat{x},.)](y) \right| \leq \frac{x_{\max}^2}{c} \frac{1}{2^{k-1}}.$$

Beweis: Wir zeigen zunächst a). Die Funktion $\kappa(.,\hat{y})$ ist auf dem Gebiet $G = (\hat{y}, 2] \times [-1,1] \subset \mathbb{C}$ holomorph. Somit existiert nach Folgerung 3.8.30 a) die Taylor-Entwicklung auf jedem Intervall $[a,b] \subset (\hat{y}, x_{\max}]$. Gilt außerdem die Bedingung (3.98), so ist auch die Abschätzung (3.99) gültig. Für die Abschätzung des Terms $\|\kappa(.,\hat{y})\|_{\infty,C}$ nutzen wir erneut aus, dass $|z| \leq 2x_{\max}$ für alle $z \in C$ sowie $\hat{y} \leq \mathrm{Re}[z]$ aufgrund von Bedingung (3.98). Folglich gilt

$$|\kappa(z,\hat{y})| = \left| \frac{z^2}{c + (z-\hat{y})\hat{y}} \right| \leq \frac{5x_{\max}^2}{c} \quad \text{bzw.} \quad \|\kappa(.,\hat{y})\|_{\infty,C} \leq \frac{5x_{\max}^2}{c}.$$

Aus (3.99) folgt damit die restliche Behauptung von a).

Für den Beweis des Teils b) bemerken wir, dass die Funktion $\kappa(\hat{x},.)$ auf dem Gebiet $G = (0,\hat{x}) \times [-1,1] \subset \mathbb{C}$ holomorph ist. Nach Folgerung 3.8.30 b) existiert somit die Taylor-Entwicklung von $\kappa(\hat{x},.)$ auf jedem Intervall $[a,b] \subset (0,\hat{x})$. Wenn die Bedingung (3.100) erfüllt ist, können wir die Abschätzung (3.101) benutzen. Für die Darstellung des Terms $\|\kappa(\hat{x},.)\|_{\infty,C}$ benutzen wir $\mathrm{Im}[z] \leq \delta$ sowie den Umstand, dass wegen Bedingung (3.100) $\mathrm{Re}[z] \leq \hat{x} \leq x_{\max}$ gilt, für alle $z \in C$. Daher haben wir

$$|\kappa(\hat{x},z)| = \left| \frac{\hat{x}^2}{c + (\hat{x}-z)z} \right| \leq \frac{x_{\max}^2}{c} \quad \text{bzw.} \quad \|\kappa(\hat{x},.)\|_{\infty,C} \leq \frac{x_{\max}^2}{c}.$$

Damit folgt aus (3.101) der Teil b). □

Lemma 3.8.32: *Sei der Emulsionskern (3.50) als Faltungskern vorgelegt, d.h. in der Form*

$$\kappa(x,y) = c_1 \left((x-y)^{\frac{2}{3}} + y^{\frac{2}{3}} \right) \left((x-y)^{\frac{2}{9}} + y^{\frac{2}{9}} \right)^{\frac{1}{2}} \exp\left[-c_2 \left(\frac{(x-y)^{\frac{1}{3}} y^{\frac{1}{3}}}{(x-y)^{\frac{1}{3}} + y^{\frac{1}{3}}} \right)^4 \right],$$

auf dem Definitionsbereich $[0, x_{\max}] \times [0, x_{\max}] \subset [0,1] \times [0,1]$.

a) Die Funktion $\kappa(., \hat{y})$ *lässt sich für ein festes* $\hat{y} \in (0, x_{\max}]$ *auf dem jedem Intervall* $[a,b] \in (\hat{y}, x_{\max})$ *in eine Taylor-Reihe entwickeln. Ist außerdem die Bedingung (3.98) aus Folgerung 3.8.30 erfüllt, so kann man den Approximationsfehler der abgeschnittenen Taylor-Reihe abschätzen durch*

$$\left| \kappa(x, \hat{y}) - \Pi_T^k[., \hat{y}](x) \right| \leq c_1 \, 6 \, x_{\max}^{\frac{7}{9}} \exp\left[c_2 \, x_{\max}^{\frac{4}{3}} \right] \frac{1}{2^{k-1}}.$$

b) Die Funktion $\kappa(\hat{x}, .)$ *lässt sich für ein festes* $\hat{x} \in [0, x_{\max}]$ *auf dem jedem Intervall* $[a,b] \in (0, \hat{x})$ *in eine Taylor-Reihe entwickeln. Ist außerdem die Bedingung (3.100) aus Folgerung 3.8.30 erfüllt, so kann man den Approximationsfehler der abgeschnittenen Taylor-Reihe abschätzen durch*

$$\left| \kappa(\hat{x}, y) - \Pi_T^k[\hat{x}, .](y) \right| \leq c_1 \, 5 \, x_{\max}^{\frac{7}{9}} \exp\left[c_2 \, 3 \, x_{\max}^{\frac{4}{3}} \right] \frac{1}{2^{k-1}}.$$

Beweis: Die Existenz der Taylor-Entwicklungen zeigen wir im Fall a) bzw. b) auf die gleiche Weise wie im Beweis von Lemma 3.8.31. Sei nun die Bedingung (3.98) bzw. (3.100) aus Folgerung 3.8.30 für a) bzw. b) erfüllt, dann gilt die Abschätzung (3.99) bzw. (3.101). Es bleiben noch die Terme $\|\kappa(., \hat{y})\|_{\infty, C}$ bzw. $\|\kappa(\hat{x}, .)\|_{\infty, C}$ für a) bzw. b) zu bestimmen. Für a) benutzen wir wieder $|z| \leq \sqrt{5} x_{\max}$ und $\hat{y} \leq \mathrm{Re}[z]$ für alle $z \in C$. Damit gilt

$$\left| c_1 \left[(z-\hat{y})^{\frac{2}{3}} + \hat{y}^{\frac{2}{3}} \right] \left[(z-\hat{y})^{\frac{2}{9}} + \hat{y}^{\frac{2}{9}} \right]^{\frac{1}{2}} \right| \leq c_1 \left[2\,|z|^{\frac{2}{3}} \right] \left[2\,|z|^{\frac{2}{9}} \right]^{\frac{1}{2}} \leq c_1 \, 6 \, x_{\max}^{\frac{7}{9}}$$

sowie

$$\left| \exp\left[-c_2 \left(\frac{(z-\hat{y})^{\frac{1}{3}} \hat{y}^{\frac{1}{3}}}{(z-\hat{y})^{\frac{1}{3}} + \hat{y}^{\frac{1}{3}}} \right)^4 \right] \right| \leq \exp\left[c_2 \left| \frac{(z-\hat{y})^{\frac{1}{3}} x_{\max}^{\frac{1}{3}}}{(z-\hat{y})^{\frac{1}{3}}} \right|^4 \right] \leq \exp\left[c_2 \, x_{\max}^{\frac{4}{3}} \right].$$

Folglich haben wir

$$\|\kappa(., \hat{y})\|_{\infty, C} \leq c_1 \, 6 \, x_{\max}^{\frac{7}{9}} \exp\left[c_2 \, x_{\max}^{\frac{4}{3}} \right].$$

Für den Fall b) haben wir wieder $\mathrm{Re}[z] \leq \hat{x} \leq x_{\max}$ und $\mathrm{Im}[z] \leq \delta$, somit gilt

$$\begin{aligned} \left| c_1 \left[(\hat{x}-z)^{\frac{2}{3}} + z^{\frac{2}{3}} \right] \left[(\hat{x}-z)^{\frac{2}{9}} + z^{\frac{2}{9}} \right]^{\frac{1}{2}} \right| &\leq c_1 \left[|z|^{\frac{2}{3}} + x_{\max}^{\frac{2}{3}} \right] \left[|z|^{\frac{2}{9}} + x_{\max}^{\frac{2}{9}} \right]^{\frac{1}{2}} \\ &\leq c_1 \, 5 \, x_{\max}^{\frac{7}{9}} \end{aligned}$$

sowie

$$\left|\exp\left[-c_2\left(\frac{(\hat{x}-z)^{\frac{1}{3}}z^{\frac{1}{3}}}{(\hat{x}-z)^{\frac{1}{3}}+z^{\frac{1}{3}}}\right)^4\right]\right| \leq \exp\left[c_2\left|\frac{(\hat{x}-z)^{\frac{1}{3}}z^{\frac{1}{3}}}{z^{\frac{1}{3}}}\right|^4\right] \leq \exp\left[c_2\,|z|^{\frac{4}{3}}\right]$$
$$\leq \exp\left[c_2\,3\,x_{\max}^{\frac{4}{3}}\right].$$

Somit folgt

$$\|\kappa(\hat{x},.)\|_{\infty,C} \leq c_1\,5\,x_{\max}^{\frac{7}{9}}\exp\left[c_2\,3\,x_{\max}^{\frac{4}{3}}\right].$$

□

Mit den Bedingungen (3.98) und (3.100) aus Folgerung 3.8.30 sind wir in der Lage ein Zulässigkeitskriterium zu formulieren, auf welchen Blöcken eine Darstellung des Operators $\boldsymbol{Q}$ als $\boldsymbol{Q}_{T(k)}$ Operator möglich ist. Dabei sehen wir die jeweiligen Intervalle $[a,b]$ als Träger von Clustern an. Die Größe der Ableitungen der Kernfunktion κ bestimmt nicht nur den Approximationsfehler, sondern auch den Quadraturfehler, siehe Lemmata 3.8.18 und 3.8.19. Beim Quadraturfehler gehen gemäß (3.88) und (3.89), im Unterschied zur Kernentwicklung, sowohl Ableitungen nach der ersten als auch nach der zweiten Variable ein. Deshalb fordern wir, dass stets beide Bedingungen (3.98) und (3.100) erfüllt sind.

Definition 3.8.33 (Zulässigkeit für gefaltete Aggregationskerne): Zwei Cluster τ und σ genügen der Zulässigkeitsbedingung $\mathcal{Z}_T(\eta)$, falls gilt

$$\max(\operatorname{diam}(\tau),\operatorname{diam}(\sigma)) < 2\,\eta\,\operatorname{dist}(\tau,\sigma) \quad \text{und} \quad \operatorname{diam}(\sigma) < 2\,\eta\,\operatorname{dist}(0,\sigma), \tag{3.102}$$

mit einem η

$$\max\left(\frac{\operatorname{diam}(\tau)}{2\operatorname{dist}(\tau,\sigma)},\frac{\operatorname{diam}(\sigma)}{2\operatorname{dist}(\tau,\sigma)},\frac{\operatorname{diam}(\sigma)}{2\operatorname{dist}(0,\sigma)}\right) < \eta < 1.$$

(Dabei entwickeln wir auf dem Cluster mit dem kleineren Durchmesser.)

Definition 3.8.34 ($\mathcal{H}_T$-Operator): Seien I, I' zwei Indexmengen und $\mathcal{T}_{I\times I'}$ ein Blockclusterbaum bzgl. der Zulässigkeitsbedingung $\mathcal{Z}_T$. Eine Matrix $\boldsymbol{M} \in \mathbb{R}^{I\times I'}$ bezeichnen wir als *$\mathcal{H}_T$-Operator* mit blockweisem Rang k, falls für jedes zulässige Blatt $b \in \mathcal{L}(\mathcal{T}_{I\times I'})$ der korrespondierende Matrixblock $M_b = (M_{ij})_{(i,j)\in b}$ lokal die Struktur des diskreten Operators $\boldsymbol{Q}_{T(k)}$ aufweist oder eine Nullmatrix ist. Die nichtzulässigen Blöcke besitzen lokal entsprechend die Struktur des diskreten Operators $\boldsymbol{Q}$.

Auch hier werden wir die Kernentwicklung mithilfe der Lagrange-Interpolation durchführen. Den benötigten Zusammenhang zwischen Lagrange- und Taylor-Approximation haben wir bereits in Lemma 3.7.33 hergeleitet.

Notation 3.8.35: Wenn im weiteren Verlauf der Arbeit von dem Begriff $\mathcal{H}$-Matrix die Rede ist, so wollen wir darunter die Gesamtheit aller bisher eingeführten Konzepte, die auf dem $\mathcal{H}$-Matrix-Kalkül beruhen, verstanden wissen. Speziell sollen damit also auch die $\mathcal{H}_V$-Matrizen sowie den eben eingeführten $\mathcal{H}_T$-Operator bezeichnet seien, vgl. Definition 3.6.30 sowie 3.8.34.

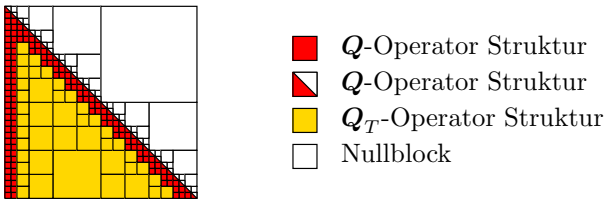


Abbildung 3.15.: $\mathcal{H}_T$-Operator

3.8.4. Komplexitätsabschätzung für $\mathcal{Z}_T$

Die Zulässigkeitsbedingung $\mathcal{Z}_T$ ist, genau wie die Standardzulässigkeit $\mathcal{Z}_S$, von beiden Clustern abhängig. Leider haben wir keine so einfache Beziehung zwischen Speicher und Rechenaufwand wie im Fall der $\mathcal{H}$-Matrizen.

Lemma 3.8.36: *Für den Speicheraufwand $N_{\mathcal{H}_T,St}(k,\mathcal{T},\mathcal{Z}_T)$ eines $\mathcal{H}_T$-Operators zum Blockclusterbaum $\mathcal{T}=\mathcal{T}_{I\times I'}$, der Zulässigkeitsbedingung $\mathcal{Z}_T$ und konstantem Rang k gilt*

$$N_{\mathcal{H}_T,St}(k,\mathcal{T},\mathcal{Z}_T) \leq 2\,|L_T|\,C_{\mathrm{sp}}\max(5k,b_{\min})\,\max(|I|,|I'|)$$

bzw. für $|I|=|I'|$

$$N_{\mathcal{H}_T,St}(k,\mathcal{T},\mathcal{Z}_T) \leq 2\,|L_T|\,C_{\mathrm{sp}}\max(5k,b_{\min})\,|I|.$$

Beweis:

$$\begin{aligned}
N_{\mathcal{H}_T,St}(k,\mathcal{T},\mathcal{Z}_T) &= \sum_{\tau\times\sigma\in\mathcal{L}^+(\mathcal{T})} N_{Q_T,St}(|\tau|,|\sigma|) \;+\; \sum_{\tau\times\sigma\in\mathcal{L}^-(\mathcal{T})} N_{Q,St}(|\tau|,|\sigma|)\\
&\leq \sum_{l\in L_T}\sum_{\tau\times\sigma\in\mathcal{L}_l^+(\mathcal{T})} 10k\max(|\tau|,|\sigma|) \;+\; \sum_{l\in L_T}\sum_{\tau\times\sigma\in\mathcal{L}_l^-(\mathcal{T})} 2|\tau||\sigma|\\
&\leq \sum_{l\in L_T}\left[10\,k\sum_{\tau\times\sigma\in\mathcal{L}_l^+(\mathcal{T})}\max(|\tau|,|\sigma|) \;+\; 2\,b_{\min}\sum_{\tau\times\sigma\in\mathcal{L}_l^-(\mathcal{T})}\max(|\tau|,|\sigma|)\right]\\
&\leq 2\max(5k,b_{\min})\sum_{l\in L_T}\sum_{\tau\times\sigma\in\mathcal{L}_l(\mathcal{T})}\max(|I|,|I'|)\\
&\leq 2\,|L_T|\,C_{\mathrm{sp}}\max(5k,b_{\min})\,\max(|I|,|I'|)
\end{aligned}$$

□

Lemma 3.8.37: *Bei der Anwendung eines $\mathcal{H}_T$-Operators zum Blockclusterbaum $\mathcal{T}=\mathcal{T}_{I\times I'}$, der Zulässigkeitsbedingung $\mathcal{Z}_T$ und konstantem Rang k auf einen Vektor gilt für den Aufwand*

$$\begin{aligned}
N_{\mathcal{H}_T,Mv}(k,\mathcal{T},\mathcal{Z}_T) \leq |L_T|\,C_{\mathrm{sp}}\max([|I|]_2,[|I'|]_2)\big[&18k\log_2\max([|I|]_2,[|I'|]_2)\\
&+\max(41k,5b_{\min})\big]
\end{aligned}$$

bzw. für $|I|=|I'|$

$$N_{\mathcal{H}_T,Mv}(k,\mathcal{T},\mathcal{Z}_T) \leq |L_T|\,C_{\mathrm{sp}}[|I|]_2\big[18k\log_2[|I|]_2+\max(41k,5b_{\min})\big].$$

Beweis: Es sei $\hat{m}_{\tau,\sigma} = \max\big([|\tau|]_2, [|\sigma|]_2\big)$ und $\hat{I} = \max([|I|]_2, [|I'|]_2)$ mit Folgerung 3.8.27 gilt dann

$$\begin{aligned}
N_{\mathcal{H}_T,Mv}(k,\mathcal{T},\mathcal{Z}_T) &= \sum_{\tau\times\sigma\in\mathcal{L}^+(\mathcal{T})} N_{Q_T,Op}(|\tau|,|\sigma|) \;+ \sum_{\tau\times\sigma\in\mathcal{L}^-(\mathcal{T})} N_{Q,Op}(|\tau|,|\sigma|) \\
&\le \sum_{l\in L_T} \sum_{\tau\times\sigma\in\mathcal{L}_l^+(\mathcal{T})} 18k\hat{m}_{\tau,\sigma}\log_2\hat{m}_{\tau,\sigma} + 41k\hat{m}_{\tau,\sigma} \;+ \sum_{l\in L_T} \sum_{\tau\times\sigma\in\mathcal{L}_l^-(\mathcal{T})} 5|\tau||\sigma| \\
&\le \sum_{l\in L_T} \Bigg[18\,k \sum_{\tau\times\sigma\in\mathcal{L}_l^+(\mathcal{T})} \hat{m}_{\tau,\sigma}\log_2\hat{m}_{\tau,\sigma} \;+ 41k \sum_{\tau\times\sigma\in\mathcal{L}_l^+(\mathcal{T})} \hat{m}_{\tau,\sigma} + 5\,b_{\min} \sum_{\tau\times\sigma\in\mathcal{L}_l^-(\mathcal{T})} \hat{m}_{\tau,\sigma} \Bigg] \\
&\le \sum_{l\in L_T} \Bigg[18\,k \sum_{\tau\times\sigma\in\mathcal{L}_l^+(\mathcal{T})} \hat{I}\log_2\hat{I} + \max(41k, 5b_{\min}) \sum_{\tau\times\sigma\in\mathcal{L}_l(\mathcal{T})} \hat{I} \;\Bigg] \\
&\le |L_T|\, C_{\mathrm{sp}}\, \hat{I}\big[18k\log_2\hat{I} + \max(41k, 5b_{\min})\big].
\end{aligned}$$

□

3.9. Fehlerbetrachtung

Bei allen Berechnungen werden Fehler unterschiedlichster Art gemacht, nur selten wird man auf Probleme stoßen, die man exakt lösen kann. Nachfolgend werden wir auf Fehler eingehen, die unser Problem betreffen, dabei bezeichne h die Gitterweite, vgl. Bemerkung 3.3.4.

3.9.1. Modellfehler

Wie in Abschnitt 2.6 bereits angedeutet, sind Modelle, die bestimmte physikalische (und/oder chemische) Vorgänge beschreiben, zum Teil hochkomplex, in dem Sinne, dass sehr viele Effekte betrachtet werden. Bei der Modellierung eines Problems gibt es verschiedene Herangehensweisen. Viele Modelle werden aus bekannten physikalischen Gesetzmäßigkeiten abgeleitet. Größtenteils macht man hier vereinfachende Annahmen, um die Komplexität des Modells zu reduzieren. Andere Modelle wiederum beruhen auf Messungen von physikalischen Größen, die beim Ablauf des entsprechenden Vorgangs im Experiment gewonnen wurden. Nachdem man Messwerte für verschiedene Zeitschritte vorliegen hat, sucht man nach einer Funktion, die besonders gut mit diesen Messwerten übereinstimmt. Mit diesen Funktionen kann man nun ein Modell erstellen.

Die meisten Modelle beschreiben den zugrundeliegenden Vorgang mit einem Modellfehler von $1\% - 10\%$, was für die meisten Anwendungen völlig ausreicht. Leider ist der Modellfehler nicht immer angegeben, bzgl. des Emulsionsmodells kann man in Coulaloglou und Tavlarides [9] auf einen Modellfehler zwischen 2% und 4% schließen.

3.9.2. Diskretisierungsfehler

Um nun ein Modell berechnen zu können, ist bis auf wenige Ausnahmen eine Diskretisierung notwendig. Hierbei macht man einen Fehler, der vom gewählten Diskretisierungsverfahren und von Annahmen über die zu erwartende Lösung abhängt. In unserem Falle

haben wir das Galerkinverfahren und stückweise konstante Ansatzfunktionen gewählt. Damit liegt der Diskretisierungsfehler im Bereich von $\mathcal{O}(h)$. Bei stückweise linearen Ansatzfunktionen liegt er beispielsweise im Bereich von $\mathcal{O}(h^2)$, siehe etwa Abschnitt 3.3.

3.9.3. Quadraturfehler

Bei der Berechnung der Einträge der Systemmatrizen bzw. bei der Erstellung der Operatoren $\boldsymbol{Q}$ sowie $\boldsymbol{Q}_T$ werden Integrale ausgewertet. Bei den Systemmatrizen verwenden wir ein Gauß-Verfahren, dessen Ordnung im Prinzip an eine vorgegebene Genauigkeit angepasst werden kann.

Bei der Kernapproximation im Fall des quadratischen Operators geben wir für die lokale Darstellung als $\boldsymbol{Q}_T$-Operator eine feste Quadraturordnung durch die Wahl der Trapezmethode vor. Der Fehler ist gemäß der Abschätzungen der Lemmata 3.8.18 und 3.8.19 gegeben und liegt demnach im Bereich von $\mathcal{O}(h^2)$. Er kann nur durch Verkleinerung von h verbessert werden.

3.9.4. Maschinenfehler

Einem Computer steht nur endlicher Speicherplatz zur Verfügung und somit gibt es gewisse Beschränkungen in der Darstellung von Zahlen. Wir interessieren uns besonders für die Darstellung der reellen Zahlen. Die reellen Zahlen werden in einem Programm i.Allg. durch so genannte *Fließkommazahlen* bis zu einer bestimmten Genauigkeit ε_{m} approximiert. Die Zahl ε_{m} bezeichnet man als *Maschinengenauigkeit*. Sie kann definiert werden gemäß

$$\varepsilon_{\mathrm{m}} = \min_{\varepsilon>0}\{\varepsilon | 1+\varepsilon >_{\mathrm{m}} 1\}, \tag{3.103}$$

wobei $>_{\mathrm{m}}$ den auf Computerebene beschränkten Vergleichsoperator $>$ bezeichnet.

Den für eine Fließkommazahl bereitgestellten Speicher bezeichnet man auch als *Bitlänge* dieser Zahl. Von dieser Bitlänge wird ein Bit für das Vorzeichen der Zahl benötigt, die restlichen Bits benutzt man für die Darstellung der Mantisse und des Exponenten der Zahl. Zur eindeutigen Darstellung gibt es die Festlegung, dass die erste Stelle der Mantisse nicht Null sein darf (Normierungsbedingung), damit steht sogar ein Bit mehr für die Darstellung der Mantisse zur Verfügung. In Tabelle 3.1 haben wir eine Übersicht, über die z.B. in der Programmiersprache C bzw. C++ benutzten Fließkommazahlen, wobei der Typ *long double* nicht immer zur Verfügung steht. Die

Bezeichnung	Bitlänge	Vorzeichen	Mantisse	Exponent	ε_{m}
float (single)	32 Bit (4 Byte)	1 Bit	23 Bit	8 Bit	$\sim 6 \cdot 10^{-8}$
double	64 Bit (8 Byte)	1 Bit	52 Bit	11 Bit	$\sim 1 \cdot 10^{-16}$
long double	96 Bit (12 Byte)	1 Bit	80 Bit	15 Bit	$\sim 4 \cdot 10^{-25}$

Tabelle 3.1.: Darstellung von Fließkommazahlen.

Maschinengenauigkeit ergibt sich aus der Länge der Mantisse und der binären Darstellung innerhalb des Computers. Für den Typ *float* haben wir z.B. eine Länge von 23

bzw. aufgrund der Normierungsbedingung 24 Bit zur Verfügung und somit eine Genauigkeit von $2^{-24} \sim 6 \cdot 10^{-8}$. Man spricht im Zusammenhang mit *single* oder *double* auch von Zahlen mit einfacher oder doppelter Genauigkeit.

Bemerkung 3.9.1: Obwohl es für die Zahl Null eine spezielle interne Darstellung gibt[1)], sollte man sich bewusst sein, dass Zahlen, deren Betrag kleiner als ε_{m} ist, im Prinzip nicht mehr von Null unterscheidbar sind.

3.9.5. Kompressionsfehler

Wenn wir die Systemmatrix durch eine $\mathcal{H}$-Matrix ersetzen, bzw. den quadratischen Operator $\boldsymbol{Q}$ durch den $\mathcal{H}_T$-Operator ersetzen, machen wir einen Fehler, welchen wir als *Kompressionsfehler* bezeichnen. Dieser kommt im wesentlichen durch die Kernapproximation zustande (Ausnahme: quadratischer Operator, siehe Abschnitt 3.9.3). Entsprechend der Abschätzungen für Kernapproximation siehe Abschnitte 3.7.1, 3.7.2 und 3.8.3 fällt der Fehler exponentiell mit $\frac{1}{2^{k-1}}$, wobei k die Interpolationsordnung bezeichnet. In den numerischen Experimenten, vgl. Kapitel 5, werden wir sehen, dass der Kompressionfehler für relativ kleine k bereits im Bereich der einfachen oder sogar doppelten Maschinengenauigkeit liegt, vgl. besonders Bemerkung 3.9.1.

3.9.6. Toeplitzabschätzungen

Die Berechnung durch Toeplitz- bzw. zirkulante Matrizen ist exakt. Bei der numerischen Berechnung spielt lediglich die Maschinengenauigkeit ε_{m} eine Rolle. Es gelten die Abschätzungen

$$\|\tilde{\boldsymbol{T}}\boldsymbol{x} - \boldsymbol{T}\boldsymbol{x}\| \leq \varepsilon_{\mathrm{m}} \|\boldsymbol{T}\boldsymbol{x}\|$$

und

$$\|\tilde{\boldsymbol{T}}\boldsymbol{x} - \boldsymbol{T}^F\boldsymbol{x}\| \leq \varepsilon_{\mathrm{m}} \|\boldsymbol{T}\boldsymbol{x}\|,$$

wobei $\boldsymbol{T}$ die exakte Matrix, $\tilde{\boldsymbol{T}}$ die Matrix in Computerdarstellung und $\boldsymbol{T}^F$ die Fourier-Darstellung der Toeplitzmatrix ist.

[1)] man setzt den Exponenten Null

4. Objektorientierte Implementierung

Die Programme, die im Verlauf dieser Arbeit entstanden, setzen auf der Softwarebibliothek LIBBEM auf, die von Christian Lage im Rahmen von Lage [32] erstellt wurde. Sie wurde in der Programmiersprache C++ verfasst, mit der Bestimmung, Randintegralgleichungen zu diskretisieren und zu lösen. Die Ränder gehören dabei zu Gebieten des $\mathbb{R}^3$, und die Kernfunktionen der Integrale sind die Laplace-Kerne.

Die Programmiersprache C++ erlaubt in hohem Maße die Anwendung objektorientierter Modellierung. Wir geben zunächst eine Übersicht über dieses Konzept. Im weiteren Verlaufe dieses Kapitels werden wir uns mit den neu entwickelten Klassen für LIBBEM beschäftigen. Dabei geben wir eine Beschreibung der Klassen, verzichten aber auf eine vollständige Klassendefinition. Unsere Angaben beschränken sich auf die Daten, die für ein Verständnis der Klassen benötigt werden. Dabei werden wir immer wieder auf das objektorientierte Konzept stoßen. Auf die Originalklassen aus LIBBEM werden wir nur eingehen, sofern wir sie für die neuen Klassen benötigen. Zum Schluss erklären wir den Einsatz der Bibliothek an einem Beispiel.

4.1. Objektorientierte Modellierung

Objektorientierte Modellierung hat das Ziel, Strukturen in einem gestellten Problem zu erkennen und diese in eine (objektorientierte) Programmiersprache zu übertragen. Oft werden Daten und deren Verarbeitung getrennt betrachtet. Bei der objektorientierten Modellierung hingegen steht das *Objekt* im Mittelpunkt, welches Daten und Operationen auf den Daten zu einer Einheit verbindet. Wir geben hier eine kurze Einführung der zentralen Ideen.

Notation 4.1.1: a) *Identität* beinhaltet die Unterteilung der betrachteten Daten in wohlunterschiedene *Objekte*. Diese Objekte können sehr unterschiedlicher Art sein, z.B. eine Zahl, eine Struktur oder auch eine Strategie.

b) *Klassen* fassen mehrere Objekte zusammen, die gemeinsame Eigenschaften (Datenstrukturen und Operationen) besitzen. Jedes Objekt gehört zu einer Klasse und wird in diesem Zusammenhang *Instanz* dieser Klasse genannt.

c) Klassen, die teilweise dieselben oder sehr ähnliche Eigenschaften besitzen, können diese miteinander teilen. Diese gemeinsamen Datenstrukturen und Methoden fasst man in einer *Ober-* oder *Basisklasse* zusammen. Klassen, die Datenstrukturen und Operationen von einer solchen Basisklasse übernehmen, nennt man von dieser *abgeleitet*, sie *erben* die Eigenschaften der Basisklasse. Dabei kann eine abgeleitete Klasse selbst wieder Basisklasse sein oder eine Klasse auch von mehreren Basisklassen abgeleitet werden. Auf diese Weise entsteht eine Klassenhierarchie.

d) Mit *Polymorphismus* bezeichnen wir den Umstand, dass dieselbe Operation unterschiedlich auf verschiedene Klassen wirken kann. Z.B. kann die Operation + auf Zahlen angewendet die gewöhnliche Addition bewirken, auf Zeichenketten angewandt jedoch die Konkatenation. Eine solche konkrete Zuordnung einer Operation zu einer Klasse nennt man *Methode*.

e) Es gibt Klassen, bei denen Operationen existieren, welche nicht durch Methoden konkretisiert wurden. Diese Klassen bezeichnen wir als *abstrakte* Klassen und derartige Operationen als *abstrakte* Methoden. Abstrakte Klassen beschreiben, welches Verhalten ein Objekt dieser Klasse besitzen soll, ohne zu spezifizieren, wie dieses Verhalten erzielt wird. Eine abstrakte Klasse eignet sich speziell als Basisklasse, eine Instanz kann auf ihr nicht erzeugt werden. Soll eine Instanz einer Klasse gebildet werden, welche von einer abstrakten Basisklasse abgeleitet wurde, so müssen die abstrakten Methoden durch konkrete Methoden überschrieben werden. (Beispiel: Matrixklasse)

f) Von *Aggregation* sprechen wir, wenn eine Klasse aus einer oder mehreren Klassen zusammengesetzt ist.

g) *Kapselung* beschreibt, ob Eigenschaften eines Objektes für andere Objekte sichtbar sein sollen oder nicht. Durch das Verstecken von Eigenschaften, wie Implementierungsdetails oder Teilen der Datenstruktur sind Änderungen der Implementierung leichter und sicherer durchzuführen.

Bemerkung 4.1.2: Punkt e) aus Notation 4.1.1 ist ein entscheidendes Kriterium für die Erweiterungsfähigkeit eines Modells. Die Bibliothek LIBBEM und auch die hier beschriebene Erweiterung können nicht alle Wünsche abdecken, die ein Benutzer vielleicht haben könnte. Durch die Verwendung von (abstrakten) Basisklassen ist eine Anpassung jedoch problemlos möglich bzw. sogar erwünscht. Voraussetzung hierbei ist allerdings ein durchdachtes objektorientiertes Modell.

Bemerkung 4.1.3: Eine Methode, die den selben Namen wie die Klasse trägt, bezeichnen wir als *Konstruktor*. Durch den Aufruf des Konstruktors wird eine Instanz erzeugt. Einem Konstruktor kann man die Daten übergeben, die für die Initialisierung der Instanz notwendig sind. Meist wird im Konstruktor auch benötigter Speicher reserviert. Es ist möglich, mehr als einen Konstruktor zu definieren.

Das Gegenstück zu einem Konstruktor ist der *Destruktor*, dieser gibt gegebenenfalls reservierten Speicher wieder frei.

Um objektorientierte Modellierungen zu beschreiben und zu veranschaulichen, wurde eine spezielle Sprache entwickelt, die *Unified Modeling Language* (UML). Mit dieser Sprache lässt sich ein objektorientiertes Modell graphisch sehr übersichtlich und beliebig detailliert darstellen, siehe z.B. Larman [33]. Wir benutzen hier nur die Konventionen für die Klassenbeschreibung, die Darstellung von abgeleiteten Klassen und die Darstellung der Aggregation. Gemäß Abbildung 4.1 stellen wir Klassen als rechteckige Boxen mit drei Feldern dar. Dabei steht im obersten Feld der Klassenname, im zweiten Feld die Daten und im letzten Feld die Methoden. Für abstrakte Klassen bzw. Methoden geben wir hinter dem Namen noch ein „(abstrakt)“ an. Gegebenenfalls lassen wir

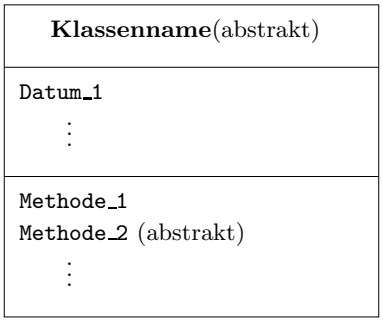


Abbildung 4.1.: Klassendarstellung.

Felder oder Namen weg, deren Existenz bereits in der Basisklasse erkennbar oder für das Verständnis nicht relevant sind.

Ableitungen kennzeichnen wir durch einen Pfeil, der von der abgeleiteten Klasse zur Basisklasse zeigt. Die Aggregation wird mit einer Raute bezeichnet, die sich an der Box der Klasse befindet, die eine andere enthält. Wir verwenden hier ... anstelle eines Klassennamens, um anzudeuten, dass hier eine Erweiterung der Bibliothek durch den Nutzer beabsichtigt ist, siehe auch Bemerkung 4.1.2.

Das Beispiel der Abbildung 4.2 stellt eine mögliche Konstellation dar. Dabei ist z.B. **Subklasse_1** von **Basisklasse_1** abgeleitet. Die abstrakte Methode `Methode_1` aus **Basisklasse_1** wird in **Subklasse_1** durch eine konkrete Methode überschrieben. Die Methode `Methode_2` wird so wie sie ist von **Subklasse_1** übernommen, d.h. sie ist gemäß Notation 4.1.1 e) ebenfalls abstrakt. Erst in **Subklasse_2** wird auch `Methode_2` konkretisiert. Die Methode `Methode_1` wird so wie sie ist von **Subklasse_1** geerbt, d.h. bei einem Aufruf dieser Methode in **Subklasse_2** wird eigentlich die Methode aus **Subklasse_1** aufgerufen. Da nun alle Methoden in **Subklasse_2** konkret definiert sind, kann man Instanzen dieser Klasse bilden, was bei **Subklasse_1** noch nicht möglich war. Weiter ist z.B. die Klasse **Subklasse_3** von den zwei Klassen **Basisklasse_2** und **Basisklasse_3** abgeleitet und erbt somit auch die Daten und Methoden beider Klassen. Letztendlich ist die **Subklasse_2** in der **Subklasse_3** enthalten bzw. ist ein Teil von **Subklasse_3**.

Für die Beziehung, **Subklasse_1** ist abgeleitet von **Basisklasse_1**, schreiben wir auch kurz **Subklasse_1 : Basisklasse_1**.

Das Angebot an Büchern, die detailliert auf objektorientiertes Design und objektorientierte Programmierung eingehen, ist groß, hier sei besonders auf Rumbauh et.al. [42] verwiesen.

4.2. Implementierung

Im Folgenden werden wir die objektorientierte Modellierung und Programmierung beschreiben, die zur Umsetzung der Verfahren notwendig sind, die wir im vorhergehenden Kapitel vorgestellt haben.

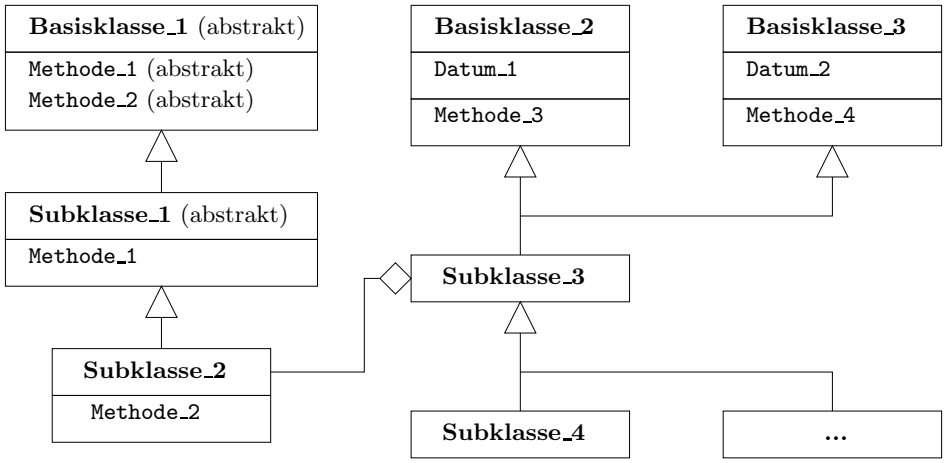


Abbildung 4.2.: Klassenbeziehungen.

4.2.1. Konzepte

Das gemeinsame Konzept von LIBBEM und der hier vorgestellten Erweiterung ist die Behandlung von Integralgleichungen. Aus diesem Grund wurde auch keine eigenständige Bibliothek erstellt. Dennoch haben wir zwei der in der Original-LIBBEM verfolgten Konzepte aufgrund der speziellen Strukturen unseres Lösungsansatzes nicht beibehalten.

In der Original-LIBBEM werden vom objektorientierten Standpunkt keine Matrizen verwendet. Stattdessen modelliert man einen Operator, welcher eine Funktion von einem Raum in einen anderen abbildet. Dabei werden die zum Operator gehörigen Kernfunktionen[1)] bereits fest eingebaut. Die Struktur der Matrix, die den diskretisierten Operator repräsentiert, wird hier verborgen (Kapselung).

Wir haben uns entschieden von diesem Ansatz abzuweichen, da die Matrixstruktur in unserer Modellierung eine so große Rolle spielt, dass wir sie nicht hinter einem Operator verbergen wollen. Somit haben wir eine Programmebene, in der wir explizit eine Matrix-Vektor-Multiplikation modellieren, mehr dazu später in Abschnitt 4.2.9. Außerdem sieht unsere Erweiterung den Einsatz verschiedener Kernfunktionen vor.

Sowohl in der Original-LIBBEM als auch in dieser Erweiterung stehen wir vor der Aufgabe, das Definitionsgebiet $G \subset \mathbb{R}^n$ der Integrale zu diskretisieren. Das führt uns auf eine disjunkte Zerlegung von G in geometrische Objekte. Gewöhnlich bevorzugt man hier einfache Strukturen, wie z.B. Dreiecke. Diese Objekte bezeichnen wir ganz allgemein als *Paneele*. Die Paneele sind gerade die Träger der Basisfunktionen aus dem Abschnitt 3.3.1.

Es gibt nun zwei Vorgehensweisen beim Aufstellen eines diskreten Operators. Man kann die Basisfunktionen durchlaufen, so erhält man jeweils einen Eintrag in der Matrix, man übernimmt also direkt das mathematische Modell. Es sei daran erinnert, dass sich gemäß Abschnitt 3.3.1 die Träger zweier Basisfunktionen überschneiden können und

[1)] Laplace-Kerne

somit ein Paneel Träger mehrerer Basisfunktionen sein kann. Der Nachteil dabei ist, dass wir ein Paneel mehrfach aufrufen müssen, was programmtechnisch nicht so effizient ist.

Die zweite Möglichkeit ist das Durchlaufen der Paneele. Auf diese Weise ruft man jedes Paneel nur einmal auf und addiert die Anteile, die das Paneel an den verschiedenen Basisfunktionen bzw. Matrixeinträgen hat.

Die Original-LIBBEM präferierte die zweite Vorgehensweise, die bei vollbesetzen bzw. unstrukturierten Matrizen Effizienzvorteile bietet.

Aufgrund der Hierarchie in der Organisation der $\mathcal{H}$-Matrizen, vgl. Abschnitt 3.6, ist das Suchen der Anteile eines Paneels im Blockclusterbaum aufwendiger als der Effizienzvorteil der zweiten Möglichkeit. Somit haben wir uns für das erste Konzept entschieden.

4.2.2. Geometrie

Die geometrischen Grundbausteine der LIBBEM sind die Paneele, die wir im vorigen Abschnitt beschrieben haben. Die in der Original-LIBBEM verwendeten Paneele waren nur für den $\mathbb{R}^3$ bzw. $\mathbb{R}^2$ geplant. Dabei wurden für jede Dimension entsprechende spezialisierte Klassen geschrieben. Für die Erweiterung benötigen wir aber eindimensionale, bzw. für einen weiteren Ausbau sogar vierdimensionale Paneele. Hier standen wir vor der Frage, ob wir ebenfalls spezielle Klassen anlegen oder in Hinblick auf eine spätere Erweiterungen einen allgemeineren Zugang wählen sollten.

Wir haben uns für den allgemeinen Zugang entschieden, da uns im Zusammenhang objektorientierter Modellierung mit den so genannten *generischen* Klassen und *generischen* Funktionen ein mächtiges Werkzeug zur Verfügung steht. Im Kontext der Programmiersprache C++ spricht man von *Template*-Klassen und *Template*-Funktionen oder auch allgemein von *Templates*. Hierbei bezeichnen wir mit dem Terminus „Funktionen“ Methoden, die keiner speziellen Klasse angehören.

Der Template-Mechanismus erlaubt es, einen Typen als Parameter[1)] für eine Klasse oder eine Funktion anzugeben.

Beispiel 4.2.1: Möchte man eine Klasse konstruieren, die n Instanzen eines Typs in einem Array verwaltet, so ist der prinzipielle Aufbau immer gleich. Ohne Templates müsste man für jeden Typen, den man in diesem Array speichern wollte, eine spezielle Klasse schreiben. Diese Klassen unterscheiden sich jedoch nur in der Angabe des Typs. Verwendet man hier ein Template, ist nur eine einzige Klasse notwendig, den entsprechenden Typen übergibt man als Parameter.

Ergänzend zu unserer bereits eingeführten Bezeichnung kennzeichnen wir Templateklassen, indem wir hinter dem Klassennamen den Parameter in spitzen Klammern angeben, gemäß: „<*Parameter*>“. In unserem Fall übergeben wir als Parameter die Dimension des Paneels, so dass im Prinzip eine Erweiterung auf beliebige Dimensionen möglich ist.

Der Nachteil von Templates ist der höhere programmtechnische Aufwand bei ihrer Anwendung. Da wir im Moment nur eindimensionale Modellfälle vorliegen haben, wurde

[1)] mitunter spricht man auch von *parametrisierten* Klassen und Funktionen

dieses Konzept zunächst nur auf Paneele und die für die Paneele notwendigen Klassen, d.h. auf die Basisbausteine der Bibliothek, angewendet.

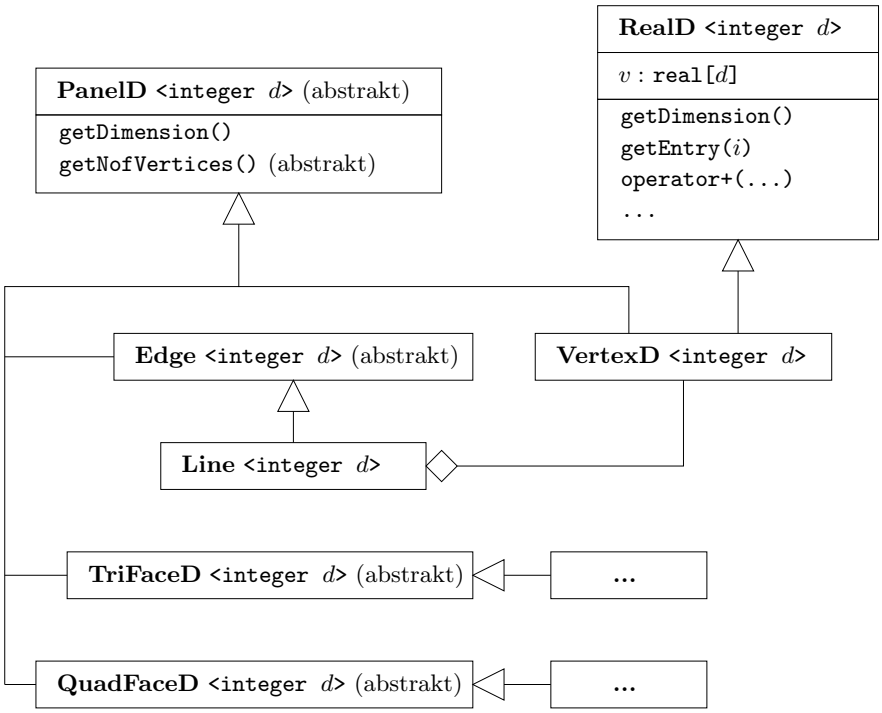


Abbildung 4.3.: Geometrische Grundbausteine - Paneele.

Klassenbeschreibung 4.2.2 (PanelD): Durch diese Basisklasse wird ein geometrisches Objekt dargestellt, welches seine Dimension und die Anzahl seiner Ecken kennt. Allen Paneelen wird über ihren Konstruktor eine eindeutige Identifizierungsnummer (Id) zugeordnet. In Abbildung 4.4 ist eine Übersicht skizziert.

a) Das einfachste Paneel, die Klasse **VertexD**, ist gerade eine solche Ecke. Außer von **PanelD** ist sie noch von **RealD** abgeleitet. Die Klasse **RealD** stellt einen Vektor im $\mathbb{R}^d$ dar. Dieser kennt seine Dimension, kann seine i-te Komponente ausgeben und es sind vektorübliche Operationen möglich. Vom mathematischen Standpunkt gibt es keinen Unterschied zwischen **VertexD** und **RealD**. Aus objektorientierter Sicht gehört die Ecke zu einer anderen Hierarchie und besitzt insbesondere eine Id.

b) Die Klasse **Edge** modelliert eine beliebige Kante im $\mathbb{R}^d$, welche insbesondere auch gekrümmt sein kann. Von ihr abgeleitet ist die Klasse **Line**, diese stellt eine Kante zur Verfügung, deren Eckpunkte durch eine gerade Linie verbunden sind.

c) Die Klassen **TriFaceD** bzw. **QuadFaceD** stellen wieder Basisklassen für d-dimensionale Objekte dar, die drei bzw. vier Seiten besitzen.

Aufgrund der Bauweise unserer Operatoren können wir die Geometrie auf Intervalle der Form $I = [x_{\min}, x_{\max}]$ beschränken. Die Zerlegung von I in Teilintervalle kann beliebig

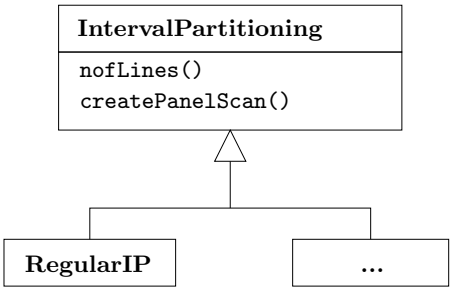


Abbildung 4.4.: Intervallzerlegung.

sein, sie wird durch die Basisklasse **IntervalPartitioning** realisiert.

Klassenbeschreibung 4.2.3 (IntervalPartitioning): Diese (nichtabstrakte) Basisklasse verwaltet ein Array von **Line**s. Dazu werden zwei Methoden zur Verfügung gestellt.

- Durch `nofLines()` wird die Anzahl der Intervalle der Partition angegeben.
- Die Methode `createPanelScan()` wird benutzt, um die **Line**s nacheinander aus dem Array zu holen.

Da wir sowohl die Anzahl der **Line**s kennen als auch die Möglichkeit sie auszulesen, ist dies die einzige nichtabstrakte Basisklasse. Ihr Konstruktor initialisiert allerdings nur die Anzahl der **Line**s und nicht das Array selbst. D.h. man kann zwar eine Instanz dieser Klasse anlegen, dies ist aber nicht sinnvoll, da das Array leer bleibt. Es ist also auch hier notwendig, eine konkrete Klasse abzuleiten. Dabei liegt der Schwerpunkt auf der Konstruktion des Arrays.

Im Moment stellen wir nur nur eine konkrete Klasse zu Verfügung, die von **IntervalPartition** abgeleitet wird. Es ist die **RegularIP**, welche eine äquidistante Zerlegung der Intervalles I modelliert, siehe Abbildung 4.4. Alle Klassen, mit Ausnahme der für die Implementierung des quadratischen Operators notwendigen, sind nicht an eine äquidistante Zerlegung gebunden. Somit kann der Benutzer beliebige Intervallzerlegungen implementieren, die optimal an seine Problemstellung angepasst sind.

Sämtliche Berechnungen dieser Arbeit erfolgten mit Hilfe von **RegularIP** und somit auf äquidistantem Gitter.

Für n-dimensionale Erweiterungen sind zunächst Konstruktionen der Form $I_1 \times I_2 \times ... \times I_n$ vorgesehen.

4.2.3. Räume

Für die Realisierung der Projektionsverfahren benötigen wir eine Modellierung von Ansatz- und Testräumen. Wie im Abschnitt 4.2.1 bereits erwähnt, erfolgt die Organisation unsereres Raumes bzgl. der Basisfunktionen.

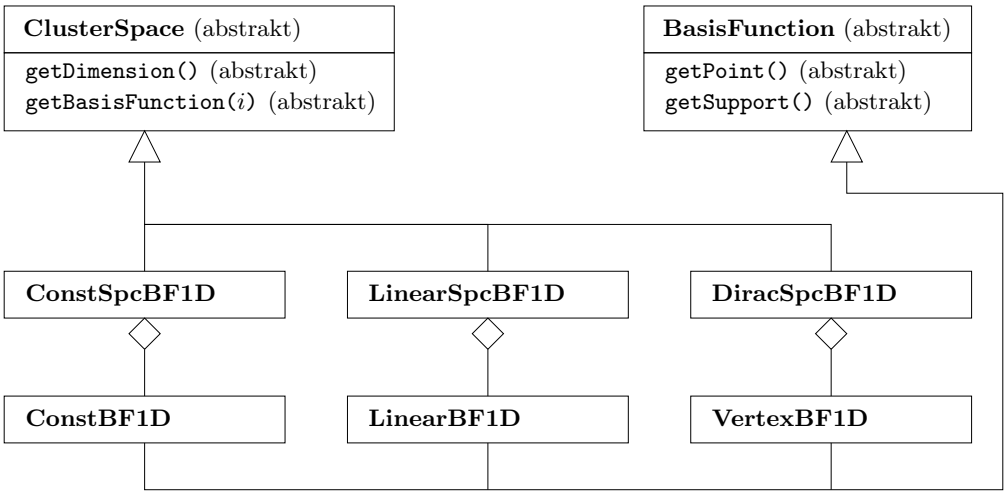


Abbildung 4.5.: Räume.

Klassenbeschreibung 4.2.4: Die Basisklasse **ClusterSpace** ist die Grundlage für die Funktionenräume, siehe Abbildung 4.5. Die Elemente dieser Räume werden durch die Basisklasse **BasisFunction** realisiert. **ClusterSpace** stellt zwei Methoden bereit:

- `getDimension()` gibt die Dimension n des Raumes zurück.

- `getBasisFunction(`i`)` liefert die i-te Basisfunktion des Raumes. Somit hat insbesondere jede Basisfunktion einen eindeutigen Index, was später bei der Konstruktion des Clusterbaumes bzw. der $\mathcal{H}$-Matrizen benötigt wird.

Die Methoden der Klasse **BasisFunction** sind:

- `getPoint()` gibt den Referenzpunkt der Basisfunktion zurück, vgl. Bemerkung 3.3.4 a).

- `getSupport()` gibt den Träger der Basisfunktion in Form eines eindimensionalen **PanelD**s an.

Wir haben die folgenden Räume als konkreten Klassen implementiert:

a) **ConstSpcBF1D** ist der Raum der stückweise konstanten Basisfunktionen. Die Elemente dieses Raumes werden gebildet aus Instanzen der Klasse **ConstBF1D**.

b) **LinearSpcBF1D** modelliert den Raum der stückweise linearen Basisfunktionen zusammen mit **LinearBF1D**.

c) **DiracSpc1D** stellt den Raum der Dirac-Funktionale dar, zusammen mit **VertexBF1D**.

4.2.4. Funktionen

Die Darstellung von reellen Funktionen realisieren wir jeweils durch die Klassen **AnalyticFunction1D**, **AnalyticFunction2D** und **AnalyticFunction4D** gemäß:

Klassenbeschreibung 4.2.5 (AnalyticFunction): Die Klasse modelliert eine Funktion $f : \mathbb{R}^d \to \mathbb{R}$ mit $d \in \{1, 2, 4\}$. Diese Klasse verfügt über die Methode:

- `evaluate(`$\boldsymbol{x}$`)`, diese wertet f an der Stelle $\boldsymbol{x} \in \mathbb{R}^d, d \in \{1, 2, 4\}$ aus und gibt den entsprechenden Funktionswert zurück.

Wir haben diese Funktionen jeweils von **Integrand_1D**, **Integrand_2D** und **Integrand_4D** abgeleitet. Diese Klassen stammen aus der Original-LIBBEM und stellen Methoden zur Integration bereit, die auch für die neuen Klassen nutzbar sind.

Mit den Klassen **AnalyticFunction** haben wir eine Schnittstelle, die es dem Benutzer ermöglicht, auf einfache Weise auch eigene Funktionen zu implementieren.

4.2.5. Kernfunktionen

Unser Lösungsansatz beruht im Wesentlichen auf der Idee, die Variablen in der Kernfunktion zu trennen. Mit der Basisklasse **HKernelFunction1D**, siehe Abbildung 4.6, modellieren wir eine Kernfunktion κ, deren abgeleitete Klassen dieses Konzept unterstützen.

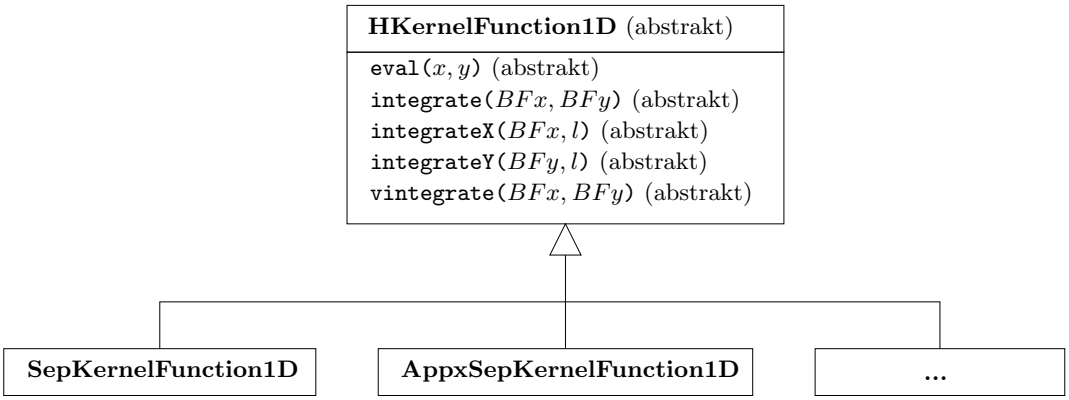


Abbildung 4.6.: Kernfunktionen.

Klassenbeschreibung 4.2.6 (HKernelFunction1D): Die Basisklasse für die Kernfunktionen ist charakterisiert durch die folgenden Methoden:

- `eval(`x, y`)` gibt den Wert $\kappa(x, y)$ zurück.
- `integrate(`BFx, BFy`)` integriert die Kernfunktion im Falle des Galerkin-Verfahrens über die Träger der Basisfunktionen BFx bzgl. x und BFy bzgl. y. Im Falle des Kollokationsverfahrens integrieren wir über den Träger von BFy bzgl. y, mit dem Referenzpunkt von BFx als Kollokationspunkt.

- Für `integrateX(`BFx, l`)` bzw. `integrateY(`BFy, l`)` erfolgt die Integration in Abhängigkeit des gewählten Verfahrens, bzgl. der getrennten Variablen x bzw. y, über den jeweiligen Träger der Basisfunktionen BFx bzw. BFy. Im Fall des Kollokationsverfahrens wird allerdings durch die Methode `integrateX(`BFx, l`)` nur eine Punktauswertung vorgenommen. Durch die Variable l wird gerade der l-te Summand aus der Darstellung separierter Kernfunktionen ausgewählt, siehe etwa Gleichungen (3.27) und (3.35).

- `vintegrate(`BFx, BFy`)` entspricht im Prinzip der bereits beschriebenen Methode `integrate(`BFx, BFy`)`, nur dass in diesem Fall über Volterra-Grenzen integriert wird und somit eine andere Vorgehensweise erforderlich ist.

Im Moment gibt es zwei konkrete Klassen, die von **HKernelFunction1D** abgeleitet sind:

a) Durch **SepKernelFunction** wird eine Kernfunktion modelliert, die bereits separabel ist,

b) während durch **AppxSepKernelFunction1D** die Separation der Variablen mit Hilfe von Interpolationspolynomen erreicht wird.

Bemerkung 4.2.7: Vom mathematischen Standpunkt aus könnte man sagen, dass es kein Unterschied ist, ob man z.B. eine **AnalyticFunction2D** oder eine **KernelFunction1D** vorliegen hat. Kernfunktionen spielen jedoch eine wichtige eigenständige Rolle, so dass vom Standpunkt objektorientierten Programmierens eine strikte Trennung beider Konzepte sinnvoll ist. Insbesondere versuchen wir Kernfunktionen in eine Form zu bringen, die eine Trennung der Variablen x und y erlaubt.

4.2.6. Interpolation

Um eine Kernfunktion κ gemäß (3.35) darzustellen, benötigen wir eine Klasse, die Interpolationspolynome modelliert. Wir haben dazu die Klasse **InterpolationOperator1D** implementiert. Die entsprechenden Interpolationspunkte werden durch die Klasse **Knots** bereit gestellt, siehe auch Abbildung 4.7.

Klassenbeschreibung 4.2.8 (InterpolationOperator1D): Diese Basisklasse stellt Interpolationspolynome bereit. Dabei stehen folgenden Methoden zur Verfügung:

- `getDeg()` gibt den Grad des Polynoms zurück.

- `getKnot(`i`)` gibt den i-ten Interpolationspunkt (Knoten) aus.

- `eval(`i, x`)` wertet das i-te Polynom an der Stelle x aus und gibt den entsprechenden Wert zurück.

Im Moment haben wir als konkrete Klasse **LagrangePolynomial1D** von **InterpolationOperator1D** abgeleitet. Wir interpolieren hier mit Hilfe von Lagrange-Polynomen, wobei wir als Interpolationspunkte gerade Tschebyscheff-Knoten verwenden, vgl. (3.32) aus Abschnitt 3.5.2 und Anhang B sowie Klassenbeschreibung 4.2.9.

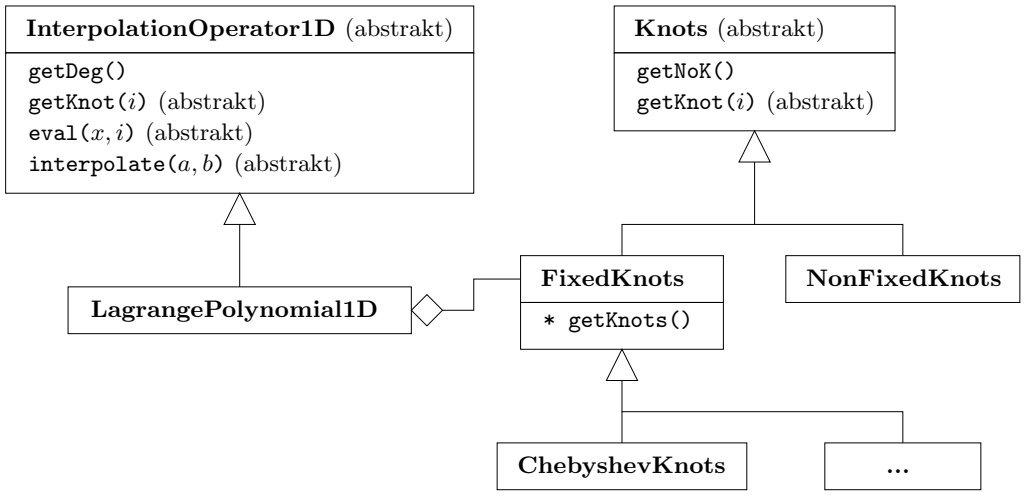


Abbildung 4.7.: Interpolation.

Klassenbeschreibung 4.2.9 (Knots): Mit dieser Klasse erzeugen wir Interpolationspunkte bzw. Knoten auf dem Intervall $[0, 1]$. Wir haben hier die Methoden:

- `getNoK()` liefert die Anzahl der Knoten.

- `getKnot(`i`)` gibt den i-ten Knoten zurück.

Von **Knots** haben wir zwei Klassen abgeleitet, die wiederum Basisklassen bilden.

In der Klasse **FixedKnots** verwalten wir ein Array mit Knoten auf dem Intervall $[0, 1]$. Wenn eine Instanz dieser Klasse erzeugt wird, werden die Knoten erzeugt bzw. dem Konstruktor übergeben und dann im Array abgelegt. Die Tschebyscheff-Knoten werden durch eine entsprechende Klasse **ChebysevKnots:FixedKnots** realisiert.

In der zweiten von **Knots** abgeleiteten Klasse, **NonFixedKnots**, erzeugen wir den i-ten Knoten jedesmal, wenn wir ihn benötigen.

4.2.7. Cluster

Weiterhin benötigen wir eine Klasse, die Cluster gemäß Abschnitt 3.6.1 modelliert.

Klassenbeschreibung 4.2.10 (Cluster1D): Die Klasse **Cluster1D** modelliert einen Cluster, welcher genau zwei Söhne hat oder gar keine. Sie enthält entsprechende Verweise[1)] auf die beiden Söhne, bzw. leere Verweise[2)]. Außerdem kennt die Klasse die Indizes der zum Cluster gehörigen Basisfunktionen und somit auch ihren Träger und ihre Boundingbox.

Nachfolgend die Methoden dieser Klasse, wobei τ_1 und τ_2 die Söhne des modellierten Clusters bezeichnen und das Intervall $[x_{\min}, x_{\max}]$ seine (eindimensionale) Boundingbox.

[1)] man spricht auch von Zeigern oder auch Pointern

[2)] hier spricht man entsprechend auch von Zeigern auf Null bzw. Null-Pointern

- Mit der Methode `getIndices()` können wir nacheinander die Indizes der zugehörigen Basisfunktionen abrufen.

- Die Methode `setBBox(`$x_{\min}$`,`$x_{\max}$`)` bzw. `getBBox()` weist die Werte der Boundingbox zu bzw. gibt die Boundingbox zurück.

- Die Methode `setSon(`i`,` τ_i`)` bzw. `getSon(`i`)` setzt bzw. liest den Verweis auf den i-ten Sohn, $i \in \{1, 2\}$.

4.2.8. Konditionen

Für die Konstruktion der $\mathcal{H}$-Matrizen benötigen wir unter anderem eine Zulässigkeitsbedingung. Außerdem müssen wir wissen, welche Volterra-Grenze benutzt wird und wir brauchen Konditionen für die Integration.

Klassenbeschreibung 4.2.11: Für die obengenannte Konditionen haben wir die folgenden Hilfsklassen implementiert:

a) Die Klasse **ConditionAdmissible** ist die Basisklasse für mögliche Zulässigkeitsbedingungen, siehe Abbildung 4.8. Die Klasse bekommt zwei Cluster τ und σ und entscheidet mit Hilfe der Methode `checkAdmiss(`τ`,`σ`)`, ob diese zulässig sind. Außerdem entscheidet die Klasse vermittels der Methode `developXorY(`τ`,`σ`)` bzgl. welcher Variablen die Kernfunktion $\kappa(.,.)$ interpoliert wird. Wir haben bereits die folgenden Zulässigkeitsbedingungen implementiert:

   1. Die Klasse **IsAdmissible** bzw. **NotAdmissible** deklariert ein Clusterpaar stets als zulässig bzw. als nicht zulässig. Diese Bedingungen benötigen wir, um eine globale R(k)-Matrix bzw. um eine vollbesetzte Matrix aufzustellen.
   2. Die Klasse **StandardAdmisibility** realisiert die Zulässigkeitsbedingung gemäß Definition 3.6.12 mit dem entsprechenden Parameter η.
   3. Durch die Klasse **LinAdmissibility** modellieren wir die Zulässigkeitsbedingung für den linearen und quasilinearen Operatoren aus Definition 3.7.24, wobei die Variable η den zugehörigen Parameter darstellt.
   4. Die Klasse **QuadAdmissibilty** modelliert die Zulässigkeitsbedingung aus Definition 3.8.33 für den quadratischen Operator, ebenfalls mit zugehörigem Parameter η.

b) Die Klasse **VolterraCondition** enthält alle nötigen Informationen für die Darstellung des Volterra-Typs gemäß Abschnitt 3.4.2. Unter anderem testen wir hier, ob wir einen Null-Matrixblock vorliegen haben.

c) In der Klasse **IntegrationCondition** werden Einstellungen für eine numerische Integration gespeichert, wie z.B. die Anzahl der Gauß-Punkte.

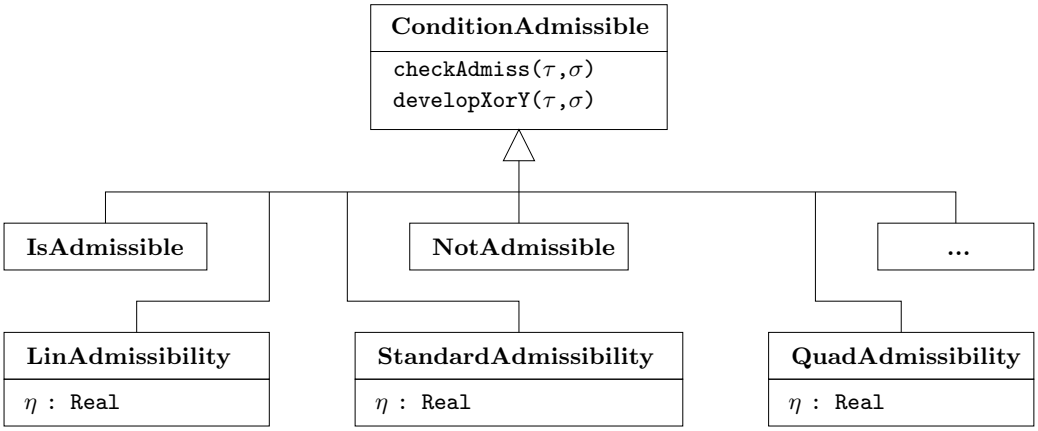


Abbildung 4.8.: Zulässigkeitsbedingungen.

4.2.9. Matrizen

Das Kernstück unserer Bibliothek bildet zweifellos die Implementierung der Matrizen. Hier konnten wir die Möglichkeiten der objektorientierten Programmierung besonders gut nutzen. Eine Übersicht über die Klassenhierarchie gibt Abbildung 4.9 auf Seite 120.

Klassenbeschreibung 4.2.12 (PDMatrix): Dies ist die Basisklasse für Matrizen in der LIBBEM-Erweiterung. Sie enthält als Daten `rows` bzw. `cols`, welche die Anzahl der Zeilen bzw. Spalten der darzustellenden Matrix repräsentieren.

Nachfolgend erläutern wir die Methoden dieser Klasse, wobei mit $\boldsymbol{M} \in \mathbb{R}^{m\times n}$ eine allgemeine Matrix bezeichnet sei.

- `getRows()` bzw. `getCols()` gibt die Zeilen- bzw. Spaltenanzahl der Matrix $\boldsymbol{M}$ aus.

- `evalMatrix(`$\alpha, \boldsymbol{x}, \beta, \boldsymbol{y}$`)` multipliziert $\boldsymbol{M}$ mit dem Vektor $\boldsymbol{x} \in \mathbb{R}^n$ nach dem Schema $\alpha\boldsymbol{M}\boldsymbol{x} + \beta\boldsymbol{y}$, wobei $\alpha, \beta \in \mathbb{R}$ sind. Das Ergebnis wird in den Vektor $\boldsymbol{y} \in \mathbb{R}^m$ geschrieben.

- `evalTransposedMatrix(`$\alpha, \boldsymbol{x}, \beta, \boldsymbol{y}$`)` analog zu `evalMatrix`, nur dass hier mit der transponierten Matrix gemäß $\alpha\boldsymbol{M}^\top\boldsymbol{x} + \beta\boldsymbol{y}$ gerechnet wird, wobei natürlich entsprechend $\boldsymbol{x} \in \mathbb{R}^m$ bzw. $\boldsymbol{y} \in \mathbb{R}^n$ gilt.

- Wenn wir eine Instanz einer von **PDMatrix** abgeleiteten Klasse bilden, legen wir zuerst den für die Matrixelemente erforderlichen Speicher an, der zunächst jedoch keine Werte enthält. Mit der Methode `makeMatrix()` füllen wir den Speicher entsprechend dem zugehörigen Matrixformat. Nicht für jede Matrix werden Matrixelemente angelegt, siehe Klassenbeschreibungen 4.2.13 a) und h).

- Die Methode `updateMatrix()` ist speziell für die Implementierung des $\mathcal{H}_T$-Operators gedacht, siehe Klassenbeschreibung 4.2.13 g).

Im Weiteren beschreiben wir die von **PDMatrix** abgeleiteten Klassen.

Klassenbeschreibung 4.2.13: Es bezeichne wieder $\boldsymbol{M} \in \mathbb{R}^{m \times n}$ eine allgemeine Matrix. Weiterhin sind nicht alle Matrixklassen direkt von **PDMatrix** abgeleitet, gemäß unserer Konvention **Basisklasse : Subklasse** haben wir diese Klassen zusätzlich gekennzeichnet.

a) Aufgrund des Volterra-Charakters der hinter den Matrizen stehenden Operatoren können einige Blöcke der $\mathcal{H}$-Matrizen Nullmatrizen sein, siehe Abschnitt 3.6.4. Für die Darstellung von Matrizen, deren Einträge ausschließlich aus Nullen bestehen, haben wir die Klasse **NullMatrix**[1)] vorgesehen. Aufgrund der überaus einfachen Struktur dieser Matrizen ist es nicht notwendig, Speicherplatz für Matrixeinträge anzulegen.

b) Die Klasse **DenseMatrix** repräsentiert eine vollbesetzte Matrix. Der Konstruktor reserviert Speicherplatz für mn Matrixeinträge.

c) Ein Spezialfall von **DenseMatrix** ist die Klasse **vDenseMatrix : DenseMatrix**. Beim Aufruf der `MakeMatrix`-Routine wird berücksichtigt, dass die Matrix Volterra-Einträge enthält bzw. entsprechende Nulleinträge. Der Konstruktor legt den gleichen Speicher wie bei **DenseMatrix** an, allerdings wird eine entsprechende Volterra-Bedingung mitgegeben. Alle anderen Methoden werden nicht neu implementiert, d.h. es werden direkt die Methoden von **DenseMatrix** aufgerufen.

d) Durch die Klasse **LowRank** werden Niedrigrang- bzw. R(k)-Matrizen implementiert. Durch den Konstruktor wird Speicherplatz für die Matrizen $\boldsymbol{S}$ und $\boldsymbol{T}$ angelegt, siehe Folgerung 3.5.4 a), d.h. für $k(m+n)$ Einträge. Die Klasse besitzt zusätzlich die Methode `getRank`, welche den Rang k der Matrix zurückgibt. In den Methoden `evalMatrix` und `evalTransposedMatrix` benutzen wir den Algorithmus gemäß Folgerung 3.5.4 b).

e) Die Klasse **vLowRank : LowRank** ist ein Spezialfall der Niedrigrang-Matrizen. Auch hier müssen wir die Volterra-Einträge berücksichtigen. Im Konstruktor wird daher zusätzlich ein Array angelegt, welches die $m \cdot C_{\mathrm{vt}}$ Volterra-Einträge gemäß Folgerung 3.5.25 enthalten soll. Beim Aufruf der `MakeMatrix`-Routine wird dieses Array mit Werten gefüllt. Die Multiplikation wird entsprechend des Algorithmus 3.5.27 angepasst.

f) Aufgrund der besonderen Struktur des quadratischen Operators $\boldsymbol{Q}$, vgl. Definiton 3.8.4, benutzen wir zu seiner Modellierung nicht die Klasse **DenseMatrix**, sondern erstellen eine eigene Klasse **DenseTMatrix**. Durch die Methode `updateMatrix` führen wir die Hadamard-Multiplikation mit den Matrizen $\boldsymbol{F}_1$ und $\boldsymbol{F}_2$ durch, vgl. Definitionen 3.8.4 und 3.8.21.

[1)] Wir könnten eine Nullmatrix z.B. auch mit Hilfe einer vollbesetzten Matrix darstellen, indem wir einfach alle Einträge Null setzen. Dies wäre aber keine saubere objektorientierte Implementierung, von unnötig reserviertem Speicherplatz ganz abgesehen.

g) **TMatrix** repräsentiert die zum Operator $\boldsymbol{Q}_T$ gehörigen Toeplitz-Matrizen und Vektoren. Die Matrix-Vektor-Multiplikation realisieren wir durch FFT, siehe Folgerung A.0.6. Die Methode `updateMatrix` realisiert wieder die Hadamard-Multiplikation mit den Matrizen $\boldsymbol{F}_1$ und $\boldsymbol{F}_2$.

h) Schließlich wird durch die Klasse **BlockMatrix** ein Block in einer Matrix repräsentiert, welcher wiederum Unterblöcke enthält. Im eigentlichen Sinne stellt die Klasse keine Matrix dar, sondern einen Container für Matrizen. Mit dieser Klasse realisieren wir die Hierarchie im Aufbau der $\mathcal{H}$-Matrizen (sowohl der $\mathcal{H}_V$- als auch $\mathcal{H}_T$-Matrizen). Jeder Unterblock kann eine der speziellen von **PDMatrix** abgeleiteten Klasse sein, insbesondere auch wieder eine **BlockMatrix**. Bei der Ausführung der Methoden der Basisklasse werden alle Informationen an die Unterblöcke weitergegeben. Aufgrund der speziellen Struktur von **BlockMatrix** wird hier ebenfalls kein Speicherplatz für Matrixeinträge benötigt.

4.2.10. $\mathcal{H}$-Matrizen

Wir haben nun alle für den Aufbau einer $\mathcal{H}$-Matrix notwendigen Klassen vorgestellt. Zusätzlich benötigen wir noch einige spezielle Funktionen, die keiner Basisklasse angehören.

a) Die Funktion `split` erstellt gemäß Algorithmus 3.6.6 einen Clusterbaum. Beim Ausführen kreieren wir zunächst den Root-Cluster, der die Wurzel des Baumes repräsentiert. Durch den rekursiven Aufruf von `split` werden die weiteren Cluster erzeugt und somit der Clusterbaum. Da jeder Cluster auf seine Söhne verweist, haben wir über den Root-Cluster Zugriff auf alle andern Cluster. Ein direkter Zugriff ist nicht möglich, aber auch nicht notwendig.

b) Die Funktion `makeBoundingBox` erstellt für jeden Cluster des Baumes die zugehörige Boundingbox. Dabei starten wir die Funktion wiederum für den Root-Cluster und erreichen durch deren rekursiven Charakter alle anderen Cluster des Baumes.

c) Durch den Aufruf der Funktion `makeBlockCluster` modellieren wir den Blockclusterbaum nach Algorithmus 3.6.15. Dabei benutzen wir keine expliziten Blockcluster, sondern stellen gleich die zugehörigen Matrizen auf, siehe Abschnitt 3.6.2. Nichtzulässige Blöcke, die keine Blätter sind, werden dabei durch die Klasse **BlockMatrix** repräsentiert. Zum Aufstellen der Blockcluster benötigen wir außerdem die Konditionen aus Abschnitt 4.2.8. Beim Aufruf der Funktion wird entsprechend jeweils eine Instanz der Klassen **ConditionAdmissible**, **VolterraCondition** und **IntegrationCondition** übergeben.

4.2.11. Massematrix

Bei der Verwendung von Projektionsverfahren treten zusätzlich zu den Operatormatrizen auf der rechten Seite Massematrizen auf, siehe (3.13). Diese passen nicht in das Konzept der **PDMatrizen** und bekommen daher eine eigene Klasse.

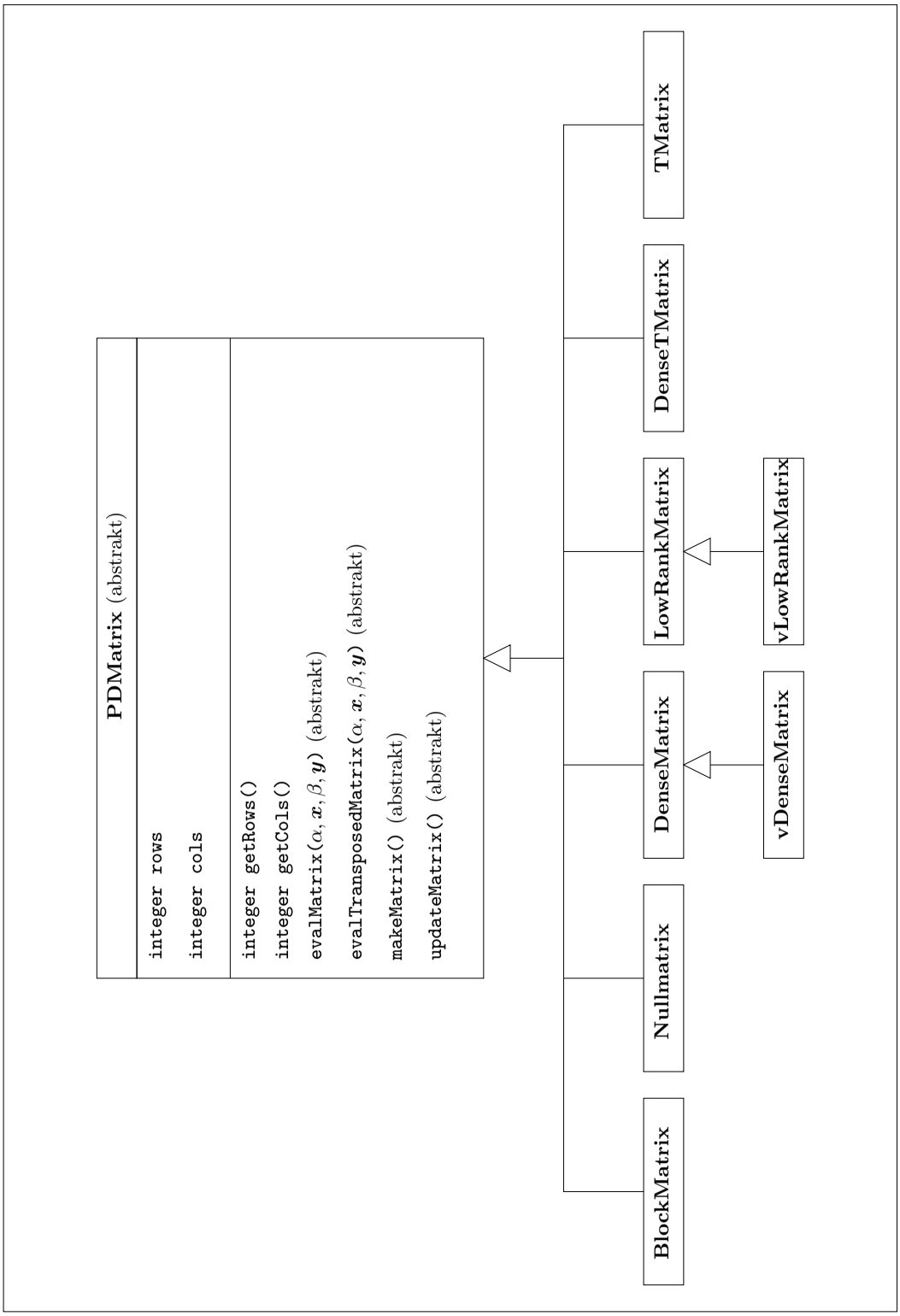


Abbildung 4.9.: Klassenhierarchie: Matrixstrukturen.

Klassenbeschreibung 4.2.14 (MassMatrix): Basisklasse der Massematrizen. Methoden: `virtual void solve(const double* b, double* x)`. Damit wird das Gleichungssystem $\boldsymbol{x} = \boldsymbol{M}\boldsymbol{y}$ nach $\boldsymbol{y}$ aufgelöst. Massematrizen sind stets symmetrisch und regulär. Wir haben entsprechend unseren Bedürfnissen Klassen für Diagonal- und Tridiagonal-Matrizen abgeleitet. Für die `solve`-Methode nutzen wir bei den Tridiagonal-Matrizen aus, dass diese positiv definit sind. Natürlich kann der Benutzer auch hier weitere von ihm benötigte Klassen ableiten.

a) **DiagonalMatrix : MassMatrix** Spezielle Form der Massematrizen tritt bei stückweise konstanten Basisfunktionen auf.

b) **SymTriDiagonalMatrix : MassMatrix** Spezielle Form der Massematrizen tritt bei stückweise linearen Basisfunktionen auf.

4.3. Ein Beispielprogramm

```
int main(int argc, char** argv)
{
  // Variablen
  unsigned gauss = 3;                                  {Anzahl der Gauß-Knoten}
  unsigned deg = 4;                                    {Grad des Interpolationsoperators}
  unsigned bmin = 3*deg;                               {Parameter b_min}
  unsigned vdepth = 1;                                 {für stückweise konstante Basisfunktionen}
  double eta = 1.0;                                    {Parameter η}
  int n = 1024;                                        {Dimension des Problems}
  int a = 0.0; int b = 1.0;                            {maximale Integrationsgrenzen}
  bool right = true; bool up = true;                   {für rechtes oberes System}

  // Interpolation
  ChebyshevKnots knots(deg);
                          {Stützstellen für Interpolation (hier Tschebyscheff-Knoten)}
  LagrangePolynomial1D iop(a, b, knots);
                                 {Interpolationsoperator (hier Lagrange-Polynome)}

  // Geometrie
  RegularIP ip(a, b, n);                               {Intervallpartition}
  ConstSpc1D_nd cs(ip);                  {benötigen wir für LIBBEM-Kompatibilität}
  ConstSpcBF1D cspc(cs);       {Raum der stückweise konstanten Basisfunktionen}

  // Funktionen
  AnalyticFunction1D* af_v;                            {Volterra-Grenze}
  AnalyticFunction1D* af_ini;                          {Anfangsbedingung}
  AppxSepKernelFunction1D* kf;                         {Kernfunktion}

  // Bedingungen
```

```
  IntCondition ic(gauss, deg);                    {Integrationsbedingungen}
  VolterraCondition vc(af_v, vdepth, n, right, up);
                                                  {Darstellung des Volterra-Typs}
  LinAdmissibility la(eta);                       {Zulässigkeitsbedingung}

  // Konstruktion der Matrizen
  cluster1D* root = split(cspc, bmax);            {Aufbau Clusterbaum}
  PDMatrix* hmatrix = split(root, root, la, vc, ic);
                                  {Konstruktion H-Matrix: Blockclusterbaum}
  hmatrix->makeMatrix(cspc, cspc, kf, ic);
                              {Konstruktion H-Matrix: Berechnung der Einträge}
  MassMatrix* mmatrix = makeMassMatrix(cspc, cspc);
                                                  {Konstruktion Massematrix}

  // Berechnungen
  double* f = new double[n];
  double* g = new double[n];                      {Arrays für Basiskoeffizienten}
  double* t = new double[n];
  for (int i = 0; i < n; i++)                     {Initialisierung von f}
    f[i] = af_ini->eval((cspc.getBF(i))->getPoint());

  hmatrix->evalMatrix(1.0, f, 0.0, t);        {Multiplikation mit H-Matrix}
  mmatrix->solve(t, g);                                {Auflösen nach g}

  // weitere Berechnungen...
  return 0;
}
```

5. Numerische Experimente

In diesem Kapitel werden die Ergebnisse der numerischen Experimente präsentiert, die mithilfe der Theorie aus Kapitel 3 und der Bibliothek aus Kapitel 4 erzielt wurden.

Dabei vergleichen wir den Aufwand von vollbesetzten und $\mathcal{H}$-Matrizen. Weiterhin interessieren wir uns für den Approximationsfehler, der entsteht, wenn wir vollbesetzte Matrizen durch die jeweilige $\mathcal{H}$-Matrix ersetzen.

Wir beginnen mit einem kleinen Exkurs über Rechnerarchitektur und werden anschließend auf die Experimente eingehen.

5.1. Ein kleiner Exkurs

An dieser Stelle soll noch einmal das Potential der effizienten Darstellung am Beispiel des benötigten Speichers eingegangen werden.

Zunächst sei bemerkt, dass Berechnungen nur dann wirklich effektiv sind, solange sie im Arbeitsspeicher eines Computers ausgeführt werden können. Durch Auslagerung von Daten auf die Festplatte verlangsamen sich die Berechnungen außerordentlich. Zum Vergleich, eine moderne Festplatte hat Zugriffszeiten um die 10 Millisekunden, moderner Arbeitspeicher hingegen um die 10 Nanosekunden, d.h. eine Festplatte ist ca. um den Faktor 10^6 langsamer.

Den meisten heutigen PCs liegt eine 32 Bit Architektur zugrunde. Damit besitzen die Speicheradressen ebenfalls eine Länge von 32 Bit und somit ist die Größe des nutzbaren Arbeitsspeichers auf 2^{32} Byte = 4 GB pro Prozessor beschränkt. Ein Programm kann aber nur einen Teil dieses Arbeitsspeichers nutzen, da es nicht autonom agiert, sondern bei seiner Ausführung mit anderen Programmen (insbesondere mit dem Betriebssystem), die ebenfalls Speicher beanspruchen, kommuniziert. Diese 4 GB sind jedoch eher eine theoretische Größe, denn die meisten PCs besitzen nur einen Prozessor und können außerdem aus Platzgründen nur bis zu 2-3 GB Speicher aufnehmen.

Eine Größe von 2 GB ist im Moment für die meisten Anwendungen vollkommen ausreichend, aber besonders für wissenschaftliche Berechnungen, speziell Simulationen, wird oft sehr viel mehr Speicher benötigt. Großrechner (und neuerdings auch PCs) bieten hier mit einer 64 Bit Architektur einen Lösungsansatz. Theoretisch lassen sich hier 2^{64} Byte = $1.7 \cdot 10^{10}$ GB (= 16 Exabyte) verwalten. Effektiv benutzt man aber „nur“ 40 Bit zur Speicheradressierung, was einem theoretischen Arbeitspeicher von 2^{40} Byte = 1024 GB = 1 Terabyte (TB) entspricht. Zum Vergleich, auf Platz eins der Top Ten der Supercomputer steht im Moment der *Earth Simulator* des japanischen *Institute for Earth Science*. Dieser besitzt 640 Rechenknoten, dies entspricht 640 eigenständigen Rechnern, die miteinander vernetzt sind. Jeder dieser Knoten besteht aus 8 Prozessoren

und verwaltet 16 GB Arbeitsspeicher (shared memory[1)]), siehe EMMEN [13]. Damit ist man noch weit unter dem theoretischem Limit von 1024 GB = 1 TB. Insgesamt stehen also 5120 Prozessoren und 10 TB Speicher zur Verfügung.

Die größten Matrizen aus unseren Experimenten haben das Format 16384×16384 bzw. $2^{14} \times 2^{14}$. Zusammen mit dem Speicherbedarf von 8 Byte für eine Zahl mit doppelter Genauigkeit (siehe Abschnitt 3.9.4, Tabelle 3.1) entspricht dies gerade einem Speicherbedarf von 2 GB für eine vollbesetzte Matrix, d.h. ein normaler PC würde hier bereits an seine Grenzen stoßen. Erhöhten wir nun die Dimension z.B. auf 2^{20}, so würden wir für eine vollbesetzte Matrix schon 8 TB benötigen, so dass wir bereits den *Earth Simulator* bemühen müssten und auch hier wäre der Speicher schon zu 80% belegt. Verallgemeinern wir die Ergebnisse aus Tabelle 5.1, so benötigte die entsprechende $\mathcal{H}_V$-Matrix mit $k = 12$ hingegen etwas weniger als 461 MB und kann somit durchaus noch von einem normalen PC bewältigt werden.

5.2. Allgemeines

In den Experimenten betrachten wir für die von uns benutzten $\mathcal{H}$-Matrizen den Speicheraufwand, den Aufwand für das Aufstellen und den Aufwand für die Matrix-Vektor-Multiplikation im Vergleich zu den vollbesetzten Matrizen.

Weiterhin interessieren wir uns für den Fehler, der entsteht, wenn wir die vollbesetzte Matrix $\boldsymbol{K}$ durch die approximierte ($\mathcal{H}$-)Matrix $\tilde{\boldsymbol{K}}$ ersetzen. Wir geben dabei den relativen Fehler

$$\varepsilon_{\text{rel}} = \frac{\|\boldsymbol{K} - \tilde{\boldsymbol{K}}\|_2}{\|\boldsymbol{K}\|_2}$$

in der Spektralnorm an. Dabei strebten wir ein ε_{rel} in der Größenordnung der Maschinengenauigkeit ε_{m} für *single* an, siehe Abschnitt 3.9.4, d.h. $\varepsilon_{\text{rel}} \sim 6 \cdot 10^{-8}$, vgl. Tabelle 3.1. Damit wäre $\boldsymbol{K}$ im Bereich der einfachen Genauigkeit nicht mehr von $\tilde{\boldsymbol{K}}$ zu unterscheiden, vgl. Bemerkung 3.9.1.

Die Berechnungen erfolgten auf den folgenden Rechnern:

a) einem Single-Prozessorsystem Athlon XP 3000+ Prozessor (2.1 GHz) mit 512 kB L2-Cache und 3 GB Arbeitsspeicher,

b) einer Sun Enterprise E6000 mit 16 Prozessoren mit jeweils 400 MHz und 16 GB Arbeitsspeicher (shared memory).

Die Approximationsfehler wurden mithilfe der E6000 bestimmt, die Effizienzmessungen (Speicher, Zeiten) hingegen auf dem Athlon-System durchgeführt.

Der Speicherbedarf der $\mathcal{H}$-Matrizen wurde anhand des vom Programm allokierten Speichers ermittelt. Somit berücksichtigen wir ebenfalls den Speicher, der durch (im Vergleich zur vollbesetzten Matrix) zusätzliche Maßnahmen[2)] erforderlich ist. Der für

[1)] gemeinsamer Speicher, die Prozessoren teilen sich den Arbeitsspeicher, theoretisch kann ein Prozessor den gesamten Speicher für sich beanspruchen

[2)] wie z.B. Aufstellen des Clusterbaums

die vollbesetzten Matrizen benötigte Speicher wurde mit Hilfe von Tabelle 3.1 berechnet, dabei ist nur der durch die Matrixeinträge verursachte Speicher enthalten.

Die graphische Darstellung der Ergebnisse der Effizienzmessungen wurde jeweils mithilfe zweier Bilder vorgenommen. Das Koordinatensystem des jeweils linken Bildes besitzt eine lineare Teilung und das des rechten Bildes eine logarithmische. Dadurch wird im linken Bild das asymptotische Verhalten der dargestellten Größen besser deutlich, während im rechten Bild die Vergleichbarkeit von vollbesetzter und $\mathcal{H}$-Matrix im Vordergrund steht.

Die Abschätzungen für die Approximation der Kernfunktionen durch separable Kerne hängen teilweise sehr stark von den auftretenden Konstanten ab, vgl. etwa Abschnitt 3.7. Daher wurden die Tests jeweils für verschiedenen Konstanten durchgeführt, insbesondere jenen aus Tabelle 2.2.

Alle Operatoren wurden mithilfe des Galerkin-Verfahrens und stückweise konstanten Basisfunktionen diskretisiert. Bei der Diskretisierung des jeweiligen Operators mithilfe des $\mathcal{H}$-Matrix-Kalküls bezeichnen wir mit k wieder den Grad der dem Lagrangeschen Interpolationsoperator zugrundeliegenden Lagrange-Polynome. Zu k gehören jeweils $\mathcal{H}_V$-Matrizen bzw. im Fall des quadratischen Operators $\mathcal{H}_T$-Operatoren mit blockweisem Rang k.

Wie schon in der Theorie, betrachten wir den linearen und quasilinearen Operator getrennt vom quadratischen.

5.3. Linearer und quasilinearer Operator

5.3.1. Allgemeines

Da sich linearer und der quasilinearer Operator numerisch sehr ähnlich verhalten, betrachten wir sie gemeinsam in einem Abschnitt.

Dem linearen Operator liegt gemäß (3.22) aus Abschnitt 3.4.2 die Darstellung

$$\mathcal{K}[f](x) = \int_x^{x_{\max}} \kappa(x,y)\ f(y)\ dy,$$

zugrunde. Als zugehörigen Dispersionskern betrachten wir den Emulsionskern, welcher definiert wurde gemäß

$$\kappa(x,y) = \varphi(x,y) \cdot \beta_{\mathrm{br}}(y) = c_{\varphi_1}\, c_{\mathrm{br},3}\, y^{-\frac{11}{9}} \exp\left[-c_{\varphi_2} \frac{(2x-y)^2}{y^2} - c_{\mathrm{br},4}\, y^{-\frac{5}{9}}\right],$$

vgl. (2.45) und (2.46).

Der quasilineare Operator besitzt nach (3.25) aus Abschnitt 3.4.2 die Form

$$\mathcal{K}[f](x) = f(x) \int_0^{x_{\max}-x} \kappa(x,y)\, f(y)\, dy.$$

Dabei legen wir als Aggregationskerne, den Smoluchowski-Kern (vgl. (2.35))

$$\kappa(x,y) = \frac{(x+y)^2}{c_{\mathrm{smol}} + xy}$$

und den Emulsionskern (vgl. (2.56))

$$\kappa(x,y) = c_{\mathrm{co},3}(x^{\frac{2}{3}} + y^{\frac{2}{3}})(x^{\frac{2}{9}} + y^{\frac{2}{9}})^{\frac{1}{2}} \exp\left[-c_{\mathrm{co},4}\left(\frac{x^{\frac{1}{3}}y^{\frac{1}{3}}}{x^{\frac{1}{3}} + y^{\frac{1}{3}}}\right)^4\right]$$

zugrunde.

Beim Aufstellen der zum linearen sowie quasilinearen Operator gehörigen $\mathcal{H}_V$-Matrizen, kam jeweils die Zulässigkeitsbedingung $\mathcal{Z}_A(\eta)$ mit $\eta = 1.0$ zum Tragen, vgl. Definition 3.7.24. Für die Gestalt der jeweiligen Matrizen erinnern wir an Abbildung 3.11 sowie 3.12. Der Parameter $b_{\min}$ wurde dabei entsprechend Bemerkung 3.6.16 gewählt, d.h. $b_{\min} = 3k$, wobei k den (blockweisen) Rang der $\mathcal{H}_V$-Matrix bezeichnet. Als Integrationsintervall wurde $[0, x_{\max}] = [0, 1]$ gewählt, dabei liegt eine äquidistante Zerlegung zugrunde.

5.3.2. Speicherbedarf

Hier erwarten wir ein lineares Verhalten der $\mathcal{H}_V$-Matrizen, gemäß der Abschätzung aus Lemma 3.7.41. Aufgrund der unterschiedlichen Volterra-Grenzen, wird sich der Speicheraufwand für die $\mathcal{H}_V$-Matrix des linearen und des quasilinearen Operators leicht unterscheiden. Auf die vollbesetzen Matrizen hat die Volterra-Grenze keinen Einfluss, d.h. sie sind in beiden Fällen gleichgroß. In Tabelle 5.1 finden wir den Speicherbedarf für den diskretisierten linearen Operator aufgelistet. Dabei enthält die zu $k = 1, \ldots, 12$ gehörige Spalte, jeweils den für $\mathcal{H}_V$-Matrizen (mit blockweisem Rang k) bzw. den für die vollbesetzte Matrix benötigten Speicher in Megabyte. Die Prozentangaben in jeder Spalte belegen, wieviel Speicher im Vergleich zur vollbesetzten Matrix (100%) benötigt wird. Wir finden hier das vorhergesagte lineare Verhalten des Speicheraufwandes der

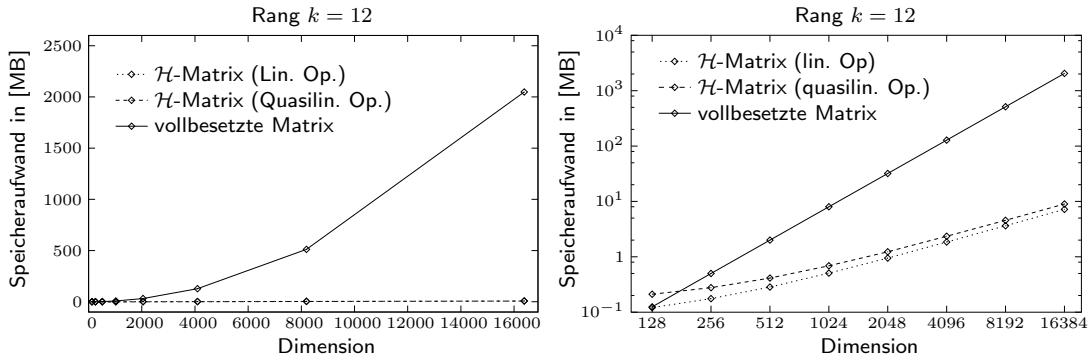


Abbildung 5.1.: Linearer und Quasilinearer Operator: Speicherbedarf.

entsprechenden $\mathcal{H}_V$-Matrizen bestätigt. Damit erhalten wir im Vergleich zur vollbesetzten Matrix mit steigender Dimension eine drastische Reduzierung des Aufwandes, da der Speicheraufwand der vollbesetzten Matrix quadratisch wächst.

Natürlich steigt der Aufwand gemäß Lemma 3.7.41 für die $\mathcal{H}_V$-Matrix auch mit größer werdendem k, aber wir benötigen bereits bei einer Dimensionsgröße von 256 für

Dimension	128		256		512		1024	
$k = 1$	0.094	75.0%	0.117	23.4%	0.156	7.8%	0.246	3.1%
$k = 2$	0.094	75.0%	0.113	22.6%	0.152	7.6%	0.234	2.9%
$k = 3$	0.094	75.0%	0.117	23.4%	0.160	8.0%	0.246	3.1%
$k = 4$	0.098	78.1%	0.125	25.0%	0.172	8.6%	0.277	3.5%
$k = 5$	0.102	81.6%	0.129	25.8%	0.184	9.2%	0.301	3.8%
$k = 6$	0.105	84.0%	0.137	27.4%	0.199	10.0%	0.328	4.1%
$k = 7$	0.105	84.0%	0.144	28.8%	0.215	10.8%	0.359	4.5%
$k = 8$	0.109	87.2%	0.152	30.4%	0.230	11.5%	0.391	4.9%
$k = 9$	0.113	90.4%	0.156	31.2%	0.242	12.1%	0.418	5.2%
$k = 10$	0.117	93.6%	0.164	32.8%	0.254	12.7%	0.445	5.6%
$k = 11$	0.117	93.6%	0.168	33.6%	0.270	13.5%	0.476	6.0%
$k = 12$	0.121	96.8%	0.176	35.2%	0.285	14.2%	0.508	6.4%
vollbesetzt	0.125	100.0%	0.5	100.0%	2	100.0%	8	100.0%
Dimension	2048		4096		8192		16384	
$k = 1$	0.414	1.3%	0.762	0.6%	1.449	0.3%	2.844	0.1%
$k = 2$	0.398	1.2%	0.730	0.6%	1.387	0.3%	2.719	0.1%
$k = 3$	0.422	1.3%	0.777	0.6%	1.492	0.3%	2.922	0.1%
$k = 4$	0.484	1.5%	0.902	0.7%	1.742	0.3%	3.422	0.2%
$k = 5$	0.527	1.6%	0.988	0.8%	1.914	0.4%	3.766	0.2%
$k = 6$	0.590	1.8%	1.113	0.9%	2.164	0.4%	4.262	0.2%
$k = 7$	0.652	2.0%	1.250	1.0%	2.430	0.5%	4.773	0.2%
$k = 8$	0.715	2.2%	1.375	1.1%	2.680	0.5%	5.277	0.3%
$k = 9$	0.766	2.4%	1.480	1.2%	2.887	0.6%	5.699	0.3%
$k = 10$	0.824	2.6%	1.605	1.2%	3.137	0.6%	6.199	0.3%
$k = 11$	0.887	2.8%	1.726	1.4%	3.387	0.7%	6.699	0.3%
$k = 12$	0.949	3.0%	1.852	1.4%	3.637	0.7%	7.199	0.4%
vollbesetzt	32	100.0%	128	100.0%	512	100.0%	2048	100.0%

Tabelle 5.1.: Linearer Operator: Benötigter Speicher in Megabyte.

$k = 12$ weniger als die Hälfte einer vollbesetzten Matrix, bei 8192 brauchen wir weniger als ein Prozent der ursprünglichen Matrix.

Im Fall des quasilinearen Operators ist das Speicherverhalten der zugehörigen $\mathcal{H}_V$-Matrix etwas schlechter, aber ebenfalls linear. Das Obengesagte ist auch für diesen Fall gültig. In Abbildung 5.1 haben wir das Verhalten für $k = 12$ und „vollbesetzt“ für linearen und quasilinearen Fall graphisch dargestellt. Im linken Bild (bei linearer Teilung), wird das lineare Verhalten der $\mathcal{H}_V$-Matrizen (für $k = 12$) im Vergleich zur vollbesetzten Matrix deutlich. Im Anhang D.2 findet man in Tabelle D.4 noch einmal das genaue Speicherverhalten des quasilinearen Operators.

5.3.3. Zeiten für das Aufstellen

Der zeitliche Aufwand, den das Aufstellen der diskretisierten Operatoren erfordert, ist proportional zum Speicheraufwand. Wir können daher ebenfalls ein quadratisches Verhalten für die vollbesetzte Matrix und lineares Verhalten für die entsprechende $\mathcal{H}_V$-Matrix erwarten. Diese Erwartungen finden wir in Abbildung 5.2 für den linea-

ren Operator bestätigt. Im Anhang D.1 haben wir zusätzlich die Resultate in D.1 in tabellarischer Form aufgelistet.

Der quasilineare Operator verhält sich ebenso wie der lineare. Die vollbesetzten Matrizen sind aber nicht direkt vergleichbar, da die Zeiten aufgrund der unterschiedlichen Kernfunktionen abweichen. Daher haben wir hier auf eine explizite Darstellung verzichtet, sie ist im Anhang D.2 in Tabelle D.5 bzw. Abbildung D.4 für den Emulsionskern zu finden.

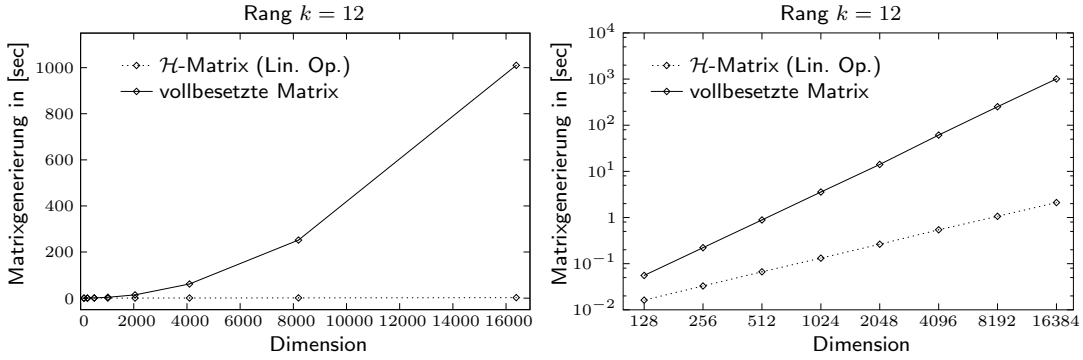


Abbildung 5.2.: Linearer Operator: Verhalten der Zeiten für das Aufstellen.

5.3.4. Zeiten für die Auswertung

Für die Anwendung der entsprechenden $\mathcal{H}_V$-Matrizen erwarten wir gemäß Folgerung 3.7.42 ebenfalls lineares Verhalten. Für $k = 12$ haben wir die entsprechenden Multiplikationszeiten mit denen der vollbesetzten Systemmatrix für linearen bzw. quasilinearen Operator verglichen, wie in Abbildung 5.3 dargestellt und finden hier ebenfalls unsere Abschätzungen bestätigt. Schon bei einer Dimension von 256 sind die Multiplikationszeiten für $\mathcal{H}_V$-Matrizen der linearen und des quasilinearen Operators um eine Größenordnung kleiner.

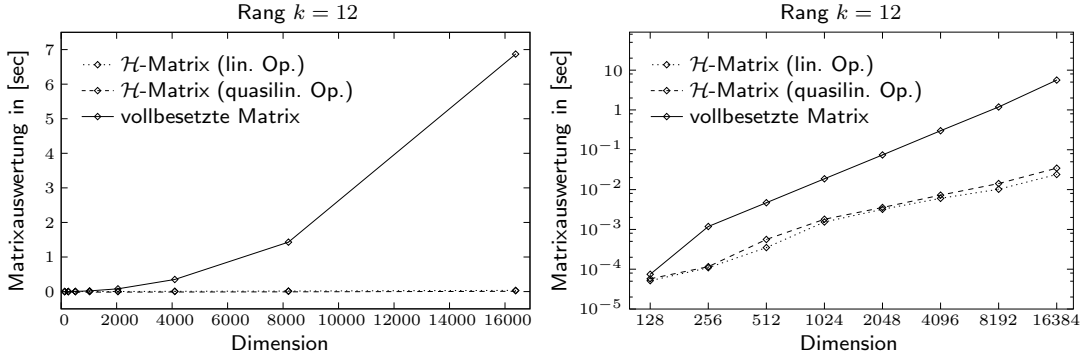


Abbildung 5.3.: Linearer und quasilinearer Operator: Verhalten der Zeiten für die Auswertung.

In Abbildung 5.3 (links) ist noch zu bemerken, dass der Anstieg der Kurve, die das

Zeitverhalten darstellt, nicht kontinuierlich ist. So ist z.B. für die vollbesetzte Matrix erst ab der Dimension 256 ein gleichmäßiger Anstieg zu beobachten. Der Grund dafür ist in der Rechnerarchitektur zu finden. Neben dem bereits erwähnten Festplatten- und Arbeitsspeicher gibt es noch einen weiteren speziellen Speicher, den so genannten *Cache*. Für uns interessant ist dabei der *L2-Cache*, der nach den anfangs gemachten Spezifikationen über die benutzte Hardware, die Größe von 512 kB besitzt. Dieser kann von dem Prozessor direkt angesprochen werden und ist ca. um eine Größenordnung schneller als der Arbeitsspeicher. Eine vollbesetzte Matrix der Dimension 128 beansprucht aber nach Tabelle 5.1 125 kB Speicher und kann somit offenbar noch vollständig in den Cache geladen werden. Daher beschleunigt sich die Auswertung durch entsprechend kürzere Zugriffszeiten. Eine vollbesetzte Matrix der Dimension 256 hingegen benötigt bereits 512 kB, damit ist das Limit des Caches bereits erreicht und es kann daher ab dieser Dimension nicht mehr direkt aus dem Cache gerechnet werden.

5.3.5. Approximationsfehler

Der Approximationfehler ε_{rel} kommt im Wesentlichen durch die separable Approximation der Kerne zustande. Für die eingangs erwähnten Kernfunktionen, den Smoluchowski-Kern und die beiden Emulsionskerne, liefern die Lemmata 3.7.20, 3.7.22 aus dem Abschnitt 3.7.1 sowie das Lemma 3.7.28 aus dem Abschnitt 3.7.2 die entsprechenden Abschätzungen. Somit erwarten wir exponentielle Konvergenz bzgl. k bzw. eine Halbierung des Fehlers beim Übergang von k zu $k+1$. Desweiteren sollte der Fehler unabhängig von der Dimension sein.

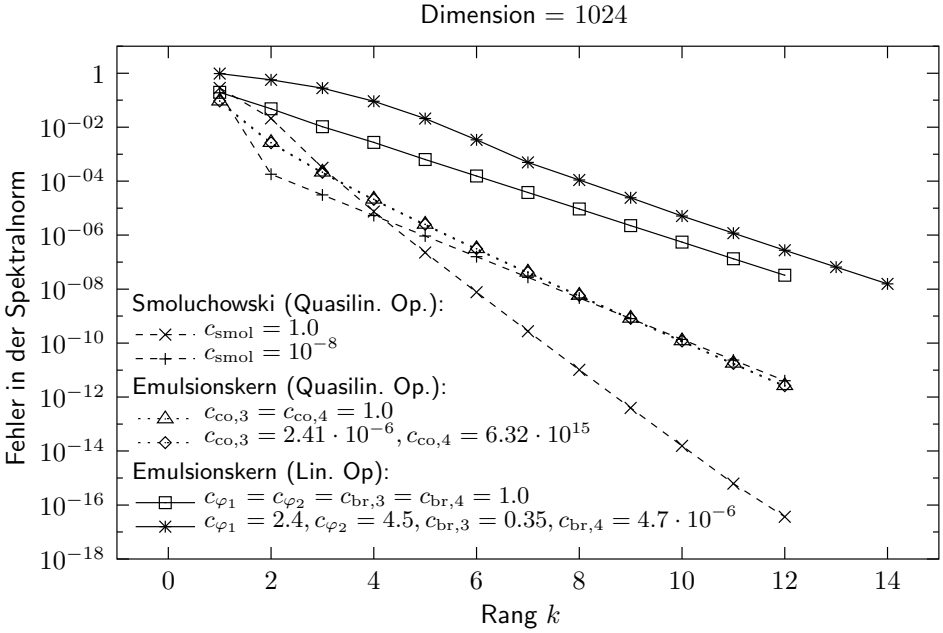


Abbildung 5.4.: Linearer und quasilinearer Operator: Verhalten des relativen Fehlers ε_{rel} für verschiedene Kernfunktionen.

In Tabelle 5.2 haben wir das Fehlerverhalten von ε_{rel} für den Fall des linearen Opera-

tors aufgelistet, dem der Emulsionskern mit den Konstanten aus Tabelle 2.2 zugrunde liegt. Zunächst erkennt man, dass der Fehler tatsächlich mit wachsendem k abnimmt. Die kleinere Zahl hinter jeder Fehlerangabe kennzeichnet dabei die Verbesserungsrate. Ab $k = 3$ erreichen wir das vorhergesagte Fehlerverhalten und sogar eine Verbesserung des Fehlers von über einem Viertel in jedem Schritt, in dem k erhöht wird. Dabei erreichen wir bei $k = 13$ die einfache Genauigkeit. Vergleicht man außerdem die Zeilen für ein festes k, so stellt man fest, dass ε_{rel} in allen Dimensionen im Wesentlichen gleich ist.

Dimension	128		256		512		1024	
$k = 1$	9.681e−01	–	9.681e−01	–	9.681e−01	–	9.681e−01	–
$k = 2$	5.697e−01	0.59	5.701e−01	0.59	5.703e−01	0.59	5.703e−01	0.59
$k = 3$	2.769e−01	0.49	2.773e−01	0.49	2.775e−01	0.49	2.776e−01	0.49
$k = 4$	9.107e−02	0.33	9.134e−02	0.33	9.143e−02	0.33	9.147e−02	0.33
$k = 5$	2.082e−02	0.23	2.093e−02	0.23	2.097e−02	0.23	2.098e−02	0.23
$k = 6$	3.352e−03	0.16	3.383e−03	0.16	3.394e−03	0.16	3.399e−03	0.16
$k = 7$	4.867e−04	0.14	4.936e−04	0.15	4.961e−04	0.15	4.971e−04	0.15
$k = 8$	1.082e−04	0.22	1.094e−04	0.22	1.099e−04	0.22	1.100e−04	0.22
$k = 9$	2.354e−05	0.22	2.390e−05	0.22	2.402e−05	0.22	2.406e−05	0.22
$k = 10$	4.976e−06	0.21	5.073e−06	0.21	5.105e−06	0.21	5.117e−06	0.21
$k = 11$	1.157e−06	0.23	1.180e−06	0.23	1.188e−06	0.23	1.190e−06	0.23
$k = 12$	2.661e−07	0.23	2.725e−07	0.23	2.747e−07	0.23	2.754e−07	0.23
$k = 13$	6.284e−08	0.24	6.446e−08	0.24	6.504e−08	0.24	6.523e−08	0.24
$k = 14$	1.504e−08	0.24	1.546e−08	0.24	1.561e−08	0.24	1.566e−08	0.24
Dimension	2048		4096		8192		16384	
$k = 1$	9.681e-01	–	9.681e-01	–	9.681e-01	–	9.681e-01	–
$k = 2$	5.704e-01	0.59	5.704e-01	0.59	5.704e-01	0.59	5.704e-01	0.59
$k = 3$	2.776e-01	0.49	2.776e-01	0.49	2.776e-01	0.49	2.776e-01	0.49
$k = 4$	9.149e-02	0.33	9.150e-02	0.33	9.150e-02	0.33	9.149e-02	0.33
$k = 5$	2.099e-02	0.23	2.099e-02	0.23	2.099e-02	0.23	2.099e-02	0.23
$k = 6$	3.401e-03	0.16	3.402e-03	0.16	3.402e-03	0.16	3.402e-03	0.16
$k = 7$	4.975e-04	0.15	4.977e-04	0.15	4.978e-04	0.15	4.977e-04	0.15
$k = 8$	1.101e-04	0.22	1.101e-04	0.22	1.101e-04	0.22	1.101e-04	0.22
$k = 9$	2.408e-05	0.22	2.409e-05	0.22	2.409e-05	0.22	2.408e-05	0.22
$k = 10$	5.121e-06	0.21	5.123e-06	0.21	5.124e-06	0.21	5.122e-06	0.21
$k = 11$	1.191e-06	0.23	1.192e-06	0.23	1.192e-06	0.23	1.191e-06	0.23
$k = 12$	2.757e-07	0.23	2.758e-07	0.23	2.758e-07	0.23	2.757e-07	0.23
$k = 13$	6.530e-08	0.24	6.533e-08	0.24	6.534e-08	0.24	6.530e-08	0.24
$k = 14$	1.568e-08	0.24	1.569e-08	0.24	1.569e-08	0.24	1.568e-08	0.24

Tabelle 5.2.: Linearer Operator: Approximationsfehler ε_{rel} für den Emulsionskern (Dispersion) mit $c_{\varphi_1} = 2.4$, $c_{\varphi_2} = 4.5$, $c_{\text{br},3} = 0.35$ und $c_{\text{br},4} = 4.7 \cdot 10^{-6}$.

In Abbildung 5.4 sehen wir für eine Dimension von 1024[1)], dass für die anderen verwendeten Kerne ein ähnliches Fehlerverhalten vorliegt, wobei verschiedenen Konstanten benutzt wurden. Tatsächlich liegt der Fehler bei $k = 12$ zum Teil sogar sehr weit unter der vorgegebenen einfachen Genauigkeit.

[1)] Dies stellt keine Einschränkung dar, da der Fehler unabhängig von der Dimension ist.

Im Anhang D kann man außerdem in den Tabellen D.3, D.7 und D.8 das genaue Fehlerverhalten zu Kernen aus Abbildung 5.4 finden.

5.4. Quadratischer Operator

5.4.1. Allgemeines

Der quadratischen Operator besitzt nach (3.77) aus Abschnitt 3.8 die Gestalt

$$\mathcal{K}[f](x) = \int_0^x \kappa(x-y,y)\, f(x-y)\, f(y)\, dy.$$

Dabei betrachten wir, wie beim quasilinearen Operator, als Dispersionskern den Smoluchowski-Kern

$$\kappa(x,y) = \frac{(x+y)^2}{c_{\text{smol}} + xy}$$

und den Emulsionskern der Dispergierung

$$\kappa(x,y) = c_{\text{co},3}(x^{\frac{2}{3}} + y^{\frac{2}{3}})(x^{\frac{2}{9}} + y^{\frac{2}{9}})^{\frac{1}{2}} \exp\left[-c_{\text{co},4}\left(\frac{x^{\frac{1}{3}}y^{\frac{1}{3}}}{x^{\frac{1}{3}} + y^{\frac{1}{3}}}\right)^4\right].$$

Hier haben wir beim Aufstellen des $\mathcal{H}_T$-Operators die Zulässigkeitsbedingung $\mathcal{Z}_T(\eta)$ mit $\eta = 1.0$ benutzt, vgl. Definition 3.8.34. Damit besitzen die dem $\mathcal{H}_T$-Operator zugrundeliegenden Matrizen die Struktur aus Abbildung 3.15. Der Parameter $b_{\min}$ wurde wieder entsprechend Bemerkung 3.6.16 gewählt, wobei sich diese Wahl im Fall des quadratischen Operators als weniger günstig herausgestellt hat, wie wir an Hand der Experimente noch sehen werden.

Für den Emulsionskern mit den Konstanten aus Tabelle 2.2 haben wir das Integrationsintervall $[0, x_{\max}] = [0, 10^{-9}]$ benutzt, was einem maximalen Tropfendurchmesser von ca. $10^{-3}m$ entspricht. Bei größeren Intervallen waren aufgrund der (großen) Konstanten der größte Teil der Operatoreinträge Null und somit wäre keine Vergleichbarkeit von $\mathcal{H}_T$-Operator und $\boldsymbol{Q}_T$ gewährleistet gewesen. Für den Emulsionskern bei dem alle Konstanten 1.0 gesetzt wurden und für den Smoluchowski-Kern haben wir auf dem Intervall $[0, x_{\max}] = [0, 1]$ gerechnet.

Die Intervalle wurden für die Diskretisierung wieder äquidistant zerlegt. Hier sei noch einmal darauf hingewiesen, dass im Fall des quadratischen Operators, im Gegensatz zum linearen und quasilinearen Operator, die von uns vorgeschlagene Approximation durch $\mathcal{H}_T$-Operatoren eine äquidistante Gitterzerlegung voraussetzt.

5.4.2. Speicherbedarf

In Abschnitt 3.8.4 finden wir in Lemma 3.8.36 die Abschätzung für den Speicherbedarf des $\mathcal{H}_T$-Operators. Hier erwarten wir kein rein lineares Verhalten mehr, sondern ein Verhalten von $\mathcal{O}(n \log n)$, wenn n die Dimension des Problems bezeichnet.

Dimension	128		256		512		1024	
$k = 1$	0.367	146.8%	0.781	78.1%	1.590	39.8%	3.465	21.7%
$k = 2$	0.367	146.8%	0.816	81.6%	1.879	47.0%	4.344	27.2%
$k = 3$	0.348	139.2%	0.836	83.6%	2.058	51.4%	5.000	31.2%
$k = 4$	0.402	160.8%	1.016	101.6%	2.566	64.2%	6.324	39.5%
$k = 5$	0.340	136.0%	0.926	92.6%	2.500	62.5%	6.457	40.4%
$k = 6$	0.363	145.2%	1.039	103.9%	2.863	71.6%	7.477	46.7%
$k = 7$	0.390	156.0%	1.148	114.8%	3.223	80.6%	8.492	53.1%
$k = 8$	0.414	165.6%	1.262	126.2%	3.586	89.6%	9.512	59.4%
$k = 9$	0.312	124.8%	1.008	100.8%	3.113	77.8%	8.750	54.7%
$k = 10$	0.316	126.4%	1.058	105.8%	3.336	83.4%	9.473	59.2%
$k = 11$	0.320	128.0%	1.109	110.9%	3.558	89.0%	10.20	63.8%
$k = 12$	0.324	129.6%	1.160	116.0%	3.781	94.5%	10.92	68.2%
vollbesetzt	0.25	100.0%	1	100.0%	4	100.0%	16	100.0%
Dimension	2048		4096		8192		16384	
$k = 1$	7.562	11.8%	16.46	6.4%	35.66	3.5%	76.88	1.9%
$k = 2$	9.934	15.5%	22.45	8.8%	50.13	4.9%	110.0	2.7%
$k = 3$	11.86	18.5%	27.53	10.8%	62.78	6.1%	141.0	3.4%
$k = 4$	15.12	23.6%	35.29	13.8%	80.82	7.9%	182.0	4.4%
$k = 5$	15.97	25.0%	38.21	14.9%	89.11	8.7%	203.0	5.0%
$k = 6$	18.61	29.1%	44.72	17.5%	104.0	10.2%	240.0	5.9%
$k = 7$	21.26	33.2%	51.24	20.0%	120.0	11.7%	276.0	6.7%
$k = 8$	23.90	37.3%	57.79	22.6%	136.0	13.3%	313.0	7.6%
$k = 9$	22.88	35.8%	56.87	22.2%	136.0	13.3%	318.0	7.8%
$k = 10$	24.91	38.9%	62.16	24.3%	149.0	14.6%	349.0	8.5%
$k = 11$	26.95	42.1%	67.44	26.3%	162.0	15.8%	363.0	8.9%
$k = 12$	28.98	45.3%	72.73	28.4%	175.0	17.1%	412.0	10.1%
vollbesetzt	64	100.0%	256	100.0%	1024	100.0%	4096	100.0%

Tabelle 5.3.: Quadratischer Operator: Benötigter Speicher in Megabyte.

In Tabelle 5.3 erhält man eine Übersicht über den Speicherverbrauch. Die zweite Angabe neben dem Speicher bezeichnet wieder den prozentualen Unterschied beim Vergleich von $\mathcal{H}_T$- und vollbesetzten Operator $\boldsymbol{Q}_T$, wobei dieser jeweils mit 100% angesetzt wurde.

Auf den Speicher bezogen „lohnt“ sich der Einsatz erst ab einer Dimensionsgröße von 1024, hier kommt auch der Einfluss von $\log n$ zum Tragen. In Abbildung 5.5 haben wir das Speicherverhalten für den festen Rang $k = 12$ graphisch dargestellt, im linken Bild erkennt man, dass sich der Speicherbedarf für den $\mathcal{H}_T$-Operator asymptotisch eher linear verhält, im Gegensatz zum vollbesetzten.

5.4.3. Zeiten für das Aufstellen

Auch im quadratischen Fall hängt die Zeit für das Aufstellen der entsprechenden Operatoren direkt mit dem Speicherbedarf zusammen, d.h. wir erwarten auch hier ein $\mathcal{O}(n \log n)$ Verhalten. In Abbildung 5.6 wurde dies für festen Rang $k = 12$ den Smoluchowski-Kern veranschaulicht, wobei im linken Bild für den $\mathcal{H}_T$-Operator ein

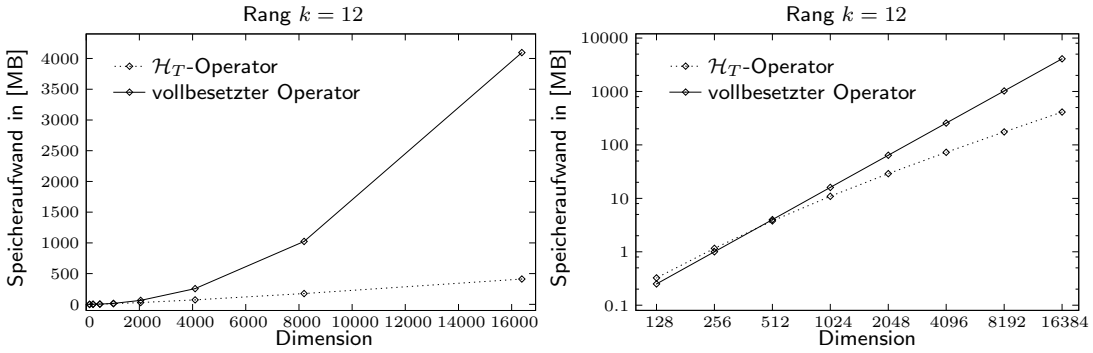


Abbildung 5.5.: Quadratischer Operator: Speicherverhalten.

fast lineares Verhalten erkennbar ist.

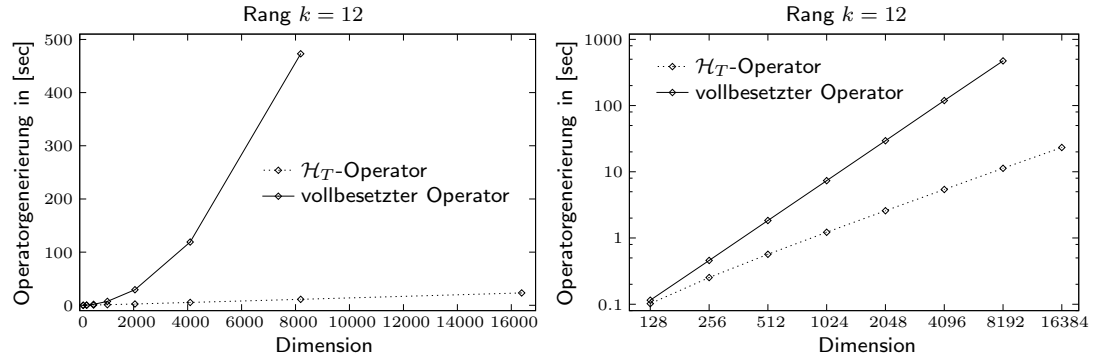


Abbildung 5.6.: Quadratischer Operator: Zeiten für das Aufstellen.

An dieser Stelle sei bemerkt, dass aufgrund des höheren Speicherverbrauchs des quadratischen Operators, der vollbesetzte Operator nur bis zu einer Dimension von 8192 darstellbar war[1)]. Entsprechend ist im linken Bild in Abbildung 5.6 der übermäßig steile Anstieg für den vollbesetzten Operator zu bewerten. Im Anhang D.3 findet man in Tabelle D.9 die genauen Werte.

Im Falle des Emulsionskerns dauert das Aufstellen des Operators aufgrund der komplizierteren Kernfunktion etwas länger, das Verhalten von $\mathcal{H}_T$- und vollbesetzten $\boldsymbol{Q}_T$-Operator entspricht ansonsten dem der Operatoren mit Smoluchowski-Kern.

5.4.4. Zeiten für die Auswertung

Den Aufwand für eine Operation mit dem diskretisierten quadratischen Operator haben wir im Abschnitt 3.8.4 in Lemma 3.8.37 abgeschätzt. Daher erwarten wir einen Aufwand, der ein Verhalten von $\mathcal{O}(n \log^2 n)$ zeigt.

In Tabelle 5.4 sind die entsprechenden Resultate dargestellt. Wir haben wieder hinter jedem Zeiteintrag den Unterschied zum vollbesetzten Operator in Prozent angegeben.

[1)] Für den Speicher konnte der Aufwand für den vollbesetzten Operator berechnet werden und somit haben wir auch ein Ergebnis für die Dimension 16384 vorliegen.

Dabei ist bemerkenswerter Weise für $k = 1, \ldots, 8$ mit steigendem k teilweise eine Abnahme des Aufwandes zu beobachten. Außerdem benötigt der $\mathcal{H}_T$-Operator in diesem Bereich sehr viel mehr Rechenzeit, so haben wir für die Dimension 128 für $k = 1$ einen extremen Mehraufwand von 7020%(!) bei der Auswertung des $\mathcal{H}_T$-Operators. Dieses Verhalten liegt darin begründet, dass die FFT-Routinen erst ab einer gewissen Matrixgröße der vollbesetzten Matrix-Vektor-Multiplikation überlegen ist. Abhilfe würde eine Anpassung des Parameters $b_{\min}$ schaffen, den man für kleine k entsprechend vergrößern müsste.

Dimension	128		256		512		1024	
$k = 1$	1.06e−02	7020%	2.56e−02	1463%	6.18e−02	854.8%	1.50e−01	530.0%
$k = 2$	4.67e−03	3093%	1.29e−02	737.1%	3.27e−02	452.3%	8.13e−02	287.3%
$k = 3$	1.97e−03	1305%	6.59e−03	376.6%	1.87e−02	258.6%	4.81e−02	170.0%
$k = 4$	2.08e−03	1378%	7.32e−03	418.3%	2.03e−02	280.8%	5.22e−02	184.4%
$k = 5$	7.84e−04	519.2%	3.94e−03	225.1%	1.30e−02	179.8%	3.66e−02	129.3%
$k = 6$	8.33e−04	551.7%	4.32e−03	246.7%	1.42e−02	196.4%	4.03e−02	142.4%
$k = 7$	8.92e−04	590.7%	4.70e−03	268.6%	1.53e−02	211.6%	4.34e−02	153.4%
$k = 8$	9.35e−04	619.2%	5.05e−03	288.6%	1.66e−02	229.6%	4.70e−02	166.1%
$k = 9$	2.47e−04	163.6%	2.81e−03	160.6%	1.13e−02	156.3%	3.63e−02	128.3%
$k = 10$	2.55e−04	168.9%	2.99e−03	170.9%	1.21e−02	167.4%	3.87e−02	136.8%
$k = 11$	2.64e−04	174.8%	3.22e−03	184.0%	1.29e−02	178.4%	4.12e−02	145.6%
$k = 12$	2.72e−04	180.1%	3.35e−03	191.4%	1.39e−02	192.2%	4.36e−02	154.1%
vollbesetzt	1.51e−04	100.0%	1.75e−03	100.0%	7.23e−03	100.0%	2.83e−02	100.0%
Dimension	2048		4096		8192		16384	
$k = 1$	3.38e−01	304.5%	7.73e−01	175.3%	1.72e+00	94.51%	3.92e+00	–
$k = 2$	1.94e−01	174.8%	4.53e−01	102.7%	1.02e+00	56.04%	2.31e+00	–
$k = 3$	1.22e−01	109.9%	2.93e−01	66.44%	6.80e−01	37.36%	1.58e+00	–
$k = 4$	1.34e−01	120.7%	3.20e−01	72.56%	7.44e−01	40.88%	1.74e+00	–
$k = 5$	9.79e−02	88.20%	2.45e−01	55.56%	5.86e−01	32.20%	1.40e+00	–
$k = 6$	1.07e−01	96.40%	2.68e−01	60.77%	6.46e−01	35.49%	1.54e+00	–
$k = 7$	1.15e−01	103.6%	2.93e−01	66.44%	7.07e−01	38.85%	1.67e+00	–
$k = 8$	1.25e−01	112.6%	3.17e−01	71.88%	7.63e−01	41.92%	1.83e+00	–
$k = 9$	1.02e−01	91.89%	2.69e−01	61.00%	6.68e−01	36.70%	1.63e+00	–
$k = 10$	1.10e−01	99.10%	2.88e−01	65.31%	7.17e−01	39.40%	1.76e+00	–
$k = 11$	1.17e−01	105.4%	3.07e−01	69.61%	7.65e−01	42.03%	1.88e+00	–
$k = 12$	1.25e−01	112.6%	3.26e−01	73.92%	8.17e−01	44.89%	2.01e+00	–
vollbesetzt	1.11e−01	100.0%	4.41e−01	100.0%	1.82e+00	100.0%	–	

Tabelle 5.4.: Quadratischer Operator: Zeiten für eine Operation in [sec].

Für $k = 9, \ldots, 12$ zeigt sich wieder ein normales Verhalten, so dass, wenn man ausschließlich die Operationszeiten als Kriterium benutzte, ab einer Dimension von 4096 (bei $k = 12$) der Einsatz des $\mathcal{H}_T$-Operators sinnvoll ist.

Wir werden im nächsten Abschnitt sehen, dass Approximationsfehler erst bei ca. $k = 9$ in dem von uns geforderten Bereich liegt, so dass wir auf eine Anpassung von $b_{\min}$ verzichtet haben.

In Abbildung 5.7 sind für $k = 12$ die Zeiten für die Auswertung graphisch dargestellt. Hier erkennt man im linken Bild für den $\mathcal{H}_T$-Operator asymptotisch ein fast lineares

Verhalten.

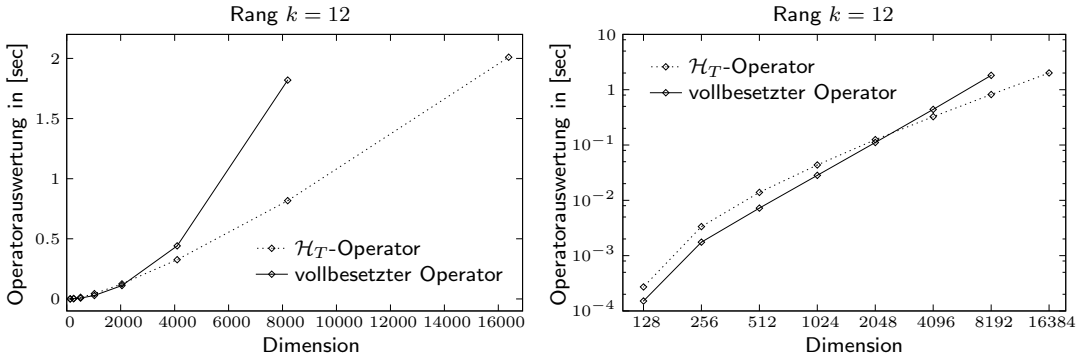


Abbildung 5.7.: Quadratischer Operator: Zeiten für die Auswertung.

5.4.5. Approximationsfehler

Der Approximationfehler beim quadratischen Operator wird durch zwei Faktoren bestimmt. Einmal, wie beim linearen bzw. quasilinearen Operator, durch den Fehler der separablen Interpolation der Kernfunktion und durch den Quadraturfehler der Trapezregel.

Der Interpolationsfehler für den Smoluchowski-Kern bzw. den Emulsionskern wurde in Lemma 3.8.31 bzw. 3.8.32 abgeschätzt. Hier erwarten wir eine Halbierung des Fehlers, wenn wir den Rang k um eins erhöhen. Die Fehlerabschätzung der Quadratur finden wir in den Lemmata 3.8.18 und 3.8.19. Die dortigen Abschätzungen prognostizieren einen Fehler, der sich wie $\mathcal{O}(h^2)$ verhält, d.h. für eine Halbierung der Gitterweite h bzw. eine Verdopplung der Dimension, erwarten wir eine Viertelung des Fehlers.

Betrachten wir zunächst den Einfluss den der Parameter k auf den Fehler hat. In Tabelle 5.5 haben wir ε_{rel} für den Emulsionskern mit den Konstanten aus Tabelle 2.2 dargestellt. Zusätzlich findet man im Anhang D.3.2 für weitere der betrachteten Kernfunktionen genaue Fehlerangaben in den Tabellen D.10, D.11 und D.12. Diese wurden ebenfalls zur Interpretation der Ergebnisse herangezogen. Neben jedem Fehler haben wir zusätzlich die Verbesserung eingetragen, die mit wachsendem k erreicht wird. Auffällig ist hier, dass eine Erhöhung von k nicht in jedem Fall eine Verbesserung mit sich bringt. Dies ist auf die doppelte Abhängigkeit des Fehler von Dimension und Rang zurückzuführen. Wird beim Übergang von k zu $k + 1$ eine Verbesserung erreicht, so entspricht diese im Wesentlichen der vorhersagten.

In Abbildung 5.8 haben wir das Fehlerverhalten mit steigendem Rang und fester Dimension 1024 für den Smoluchowski- und den Emulsionskern skizziert. Dabei wurde wieder mit verschiedenen Konstanten benutzt. Hier ist gut zu erkennen, welcher Fehlereinfluss jeweils dominiert.

Wir betrachten nun den Fehler in Abhängigkeit von der Dimension, wie in Abbildung 5.9 für festen Rang $k = 12$ dargestellt. Zunächst bemerken wir für den Emulsionskern wie erwartet eine Reduktion des Fehlers mit zunehmender Dimensionsgröße. Die Fehlerverbesserung selbst liegt allerdings etwa bei einer Halbierung, anstelle der vorherge-

Dimension	128		256		512		1024	
$k = 1$	3.462e-02	–	3.465e-02	–	3.466e-02	–	3.466e-02	–
$k = 2$	3.081e-03	0.09	3.080e-03	0.09	3.080e-03	0.09	3.080e-03	0.09
$k = 3$	2.725e-04	0.09	2.731e-04	0.09	2.727e-04	0.09	2.726e-04	0.09
$k = 4$	4.858e-05	0.18	4.849e-05	0.18	4.834e-05	0.18	4.829e-05	0.18
$k = 5$	1.003e-05	0.21	7.931e-06	0.16	7.932e-06	0.16	7.898e-06	0.16
$k = 6$	9.944e-06	0.99	5.142e-06	0.65	2.559e-06	0.32	1.601e-06	0.20
$k = 7$	9.865e-06	0.99	5.089e-06	0.99	2.521e-06	0.98	1.278e-06	0.80
$k = 8$	9.859e-06	1.00	5.086e-06	1.00	2.519e-06	1.00	1.276e-06	1.00
$k = 9$	3.548e-06	0.36	2.466e-06	0.48	1.272e-06	0.50	6.301e-07	0.49
$k = 10$	3.548e-06	1.00	2.466e-06	1.00	1.272e-06	1.00	6.301e-07	1.00
$k = 11$	3.548e-06	1.00	2.466e-06	1.00	1.272e-06	1.00	6.301e-07	1.00
$k = 12$	3.548e-06	1.00	2.466e-06	1.00	1.272e-06	1.00	6.301e-07	1.00
Dimension	2048		4096		8192		16384	
$k = 1$	3.466e-02	–	3.466e-02	–	3.466e-02	–	–	
$k = 2$	3.080e-03	0.09	3.080e-03	0.09	3.080e-03	0.09	–	
$k = 3$	2.725e-04	0.09	2.724e-04	0.09	2.725e-04	0.09	–	
$k = 4$	4.828e-05	0.18	4.827e-05	0.18	4.827e-05	0.18	–	
$k = 5$	7.854e-06	0.16	7.884e-06	0.16	7.886e-06	0.16	–	
$k = 6$	1.581e-06	0.20	1.577e-06	0.20	1.576e-06	0.20	–	
$k = 7$	6.742e-07	0.43	3.727e-07	0.24	3.067e-07	0.20	–	
$k = 8$	6.720e-07	1.00	3.667e-07	0.98	2.060e-07	0.67	–	
$k = 9$	3.193e-07	0.48	1.681e-07	0.46	9.175e-08	0.44	–	
$k = 10$	3.193e-07	1.00	1.681e-07	1.00	9.174e-08	1.00	–	
$k = 11$	3.193e-07	1.00	1.681e-07	1.00	9.173e-08	1.00	–	
$k = 12$	3.193e-07	1.00	1.681e-07	1.00	9.173e-08	1.00	–	

Tabelle 5.5.: Quadratischer Operator: Relativer Approximationsfehler ε_{rel} für den Emulsionskern mit $c_{\text{co},3} = 2.41 \cdot 10^{-6}$ und $c_{\text{co},4} = 6.32 \cdot 10^{15}$.

sagten Viertelung. Da der Quadraturfehler auch von der zweiten Ableitung des Kerns abhängt, ist zu vermuten, dass ihr Einfluss in den von uns betrachteten Dimensionen noch zu groß ist. Wir erreichen einfache Genauigkeit ab der Dimension 8192 und für $k = 9$.

Für den Smoluchowski-Kern mit $c_{\text{smol}} = 10^{-8}$ ist keine Fehlerreduzierung zu beobachten, und somit wird auch die geforderte einfache Genauigkeit nicht erreicht. Offenbar überwiegt hier der Einfluss der zweiten Ableitungen. Somit ist die Trapezregel für die Quadratur der Integrale des quadratischen Operators bei der Verwendung des Smoluchowski-Kerns mit kleinen c_{smol} ungeeignet, selbst auf den zulässigen Gebieten.

Der Smoluchowski-Kern mit $c_{\text{smol}} = 1.0$ zeigt das beste Fehlerverhalten und erfüllt als einziger Kern die vorhergesagte Viertelung. Er liegt ab Dimension 2048 im Bereich der einfachen Genauigkeit.

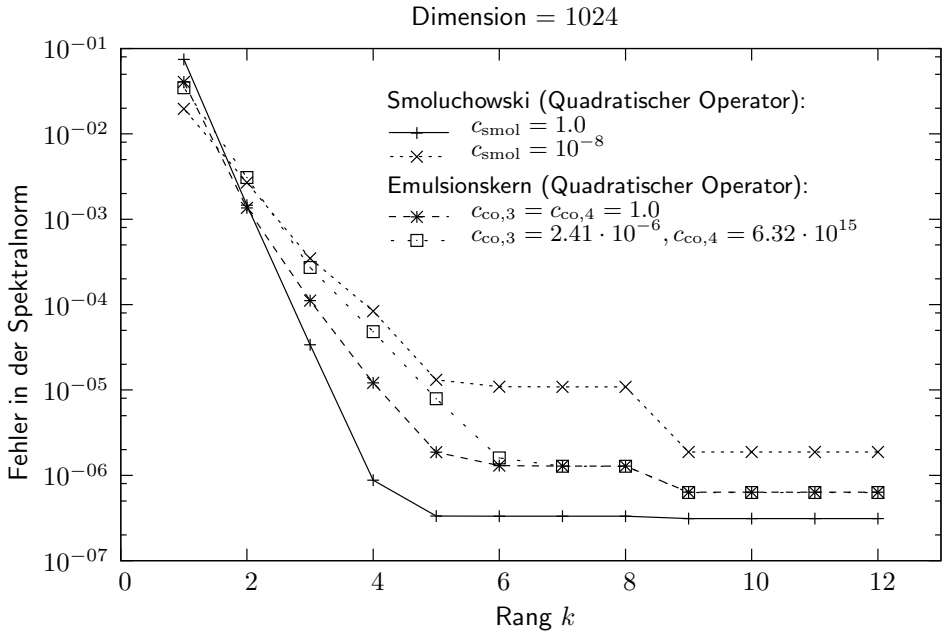


Abbildung 5.8.: Quadratischer Operator: Verhalten des relativen Fehlers $\varepsilon_{\mathrm{rel}}$ für verschiedene Kernfunktionen und feste Dimension (1024).

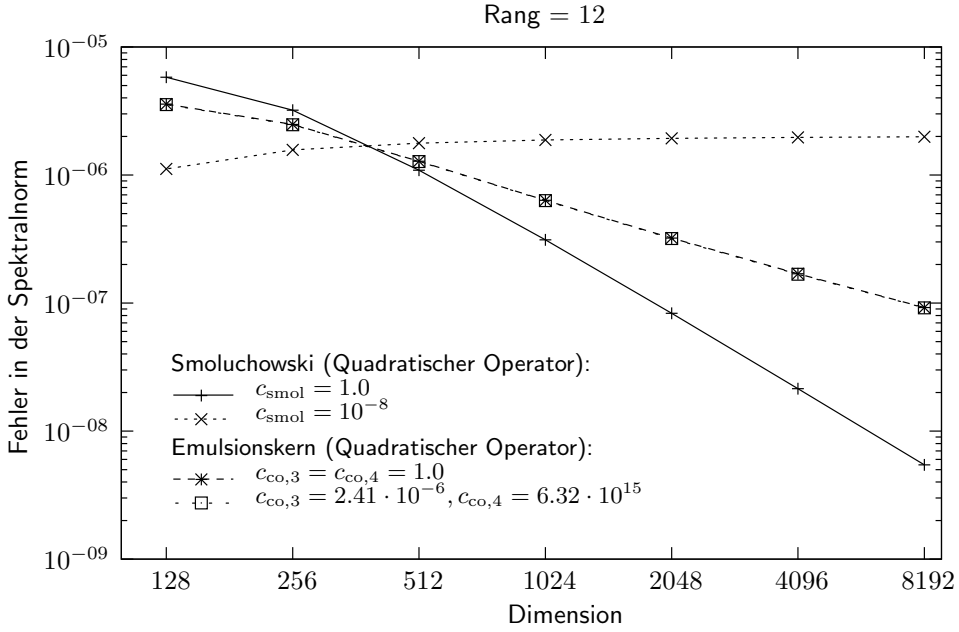


Abbildung 5.9.: Quadratischer Operator: Verhalten des relativen Fehlers $\varepsilon_{\mathrm{rel}}$ für verschiedene Kernfunktionen und festen Rang ($k = 12$).

Fazit

Ausgangspunkt war die numerische Behandlung von Volterra-Integraloperatoren, wie sie typischerweise in populationsdynamischen Modellen auftreten. Durch Galerkin-Diskretisierung erhält man bei naiver Herangehensweise vollbesetzte Systemmatrizen, welchen quadratische Komplexität zugrunde liegt. Das Ziel war es, diese durch Matrizen mit besserer Komplexität zu approximieren.

Der grundlegende Gedanke war dabei die Separation der Kernfunktion κ. Liegt z.B. eine separable Kernfunktion vor, so kann die zum Operator gehörige Systemmatrix in kanonischer Weise als Niedrigrangmatrix dargestellt werden. In dieser Darstellung ist keine Approximation mehr notwendig, in diesem Falle besitzt die Systemmatrix bereits eine Gestalt, die eine effiziente Speicherung und Auswertung erlaubt.

Für nichtseparable Kernfunktionen wurden die Ideen des $\mathcal{H}$-Matrix-Kalküls benutzt, der entsprechend auf variable Integralgrenzen erweitert wurde. An wichtigen Beispielkernen wurden dabei (Zulässigkeits-)Kriterien erforscht und formuliert, mit deren Hilfe man abschätzen kann, ob eine (lokale) Approximation von κ durch ein Interpolationspolynom und somit durch eine separable Kernfunktion sinnvoll ist. Bei entsprechendem Verhalten der Kernfunktion kann man für die Systemmatrix eine blockweise Approximation durch Niedrigrangmatrizen erreichen. Die approximierende Matrix wird dann als $\mathcal{H}$-Matrix bezeichnet.

Für die Beispielkerne des linearen und quasilinearen Operators konnte gezeigt werden, dass diese den Zulässigkeitskriterien genügen und somit optimale Komplexität $\mathcal{O}(n)$ erreicht werden kann. Durch diese lineare Komplexität ist es bereits bei kleinen Problemgrößen n vorteilhaft, die vollbesetze Matrix durch $\mathcal{H}$-Matrizen zu ersetzen. Der Approximationsfehler kann dabei über den Grad des Interpolationspolynoms gesteuert werden und somit beliebig klein werden.

Bei der Approximation der Systemmatrizen des quadratischen Operators wurde außerdem der Faltungscharakter der zugehörigen Kernfunktionen ausgenutzt. Man erhält auf diese Weise eine blockweise Approximation der Systemmatrizen durch Toeplitz-Matrizen. Solche Matrizen können mithilfe der schnellen Fourier-Transformation mit einem Vektor multipliziert werden.

Hier konnten nicht so gute Ergebnisse erzielt werden wie im linearen bzw. quasilinearen Fall. Die Komplexität für die Speicherung beträgt $\mathcal{O}(n \log n)$ und für die Anwendung auf einen Vektor $\mathcal{O}(n \log^2 n)$. Der Einsatz von $\mathcal{H}$-Matrizen, welche die Systemmatrizen des quadratischen Operators approximieren, ist erst bei höheren Dimensionen sinnvoll. Weiterhin ist auch der Approximationsfehler neben dem Grad des Interpolationspolynoms noch von der Dimension abhängig. Damit erreichen wir kleine Approximationfehler erst in hohen Dimensionen.

A. Toeplitz-Matrizen

Wir führen zunächst zwei spezielle Matrizenräume ein.

Definition A.0.1 (Toeplitz-Matrix): Eine Matrix der Form

$$\boldsymbol{T} = \begin{pmatrix} t_0 & t_1 & \dots & t_{n-2} & t_{n-1} \\ t_{-1} & t_0 & \dots & t_{n-3} & t_{n-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ t_{-(m-2)} & t_{-(m-3)} & \dots & t_0 & t_1 \\ t_{-(m-1)} & t_{-(m-2)} & \dots & t_{-1} & t_0 \end{pmatrix} \tag{A.1a}$$

nennen wir *Toeplitz-Matrix*. Die Elemente t_{ij} von $\boldsymbol{T} \in \mathbb{R}^{m \times n}$ hängen also nur von der Differenz $(j - i)$ ab, d.h.

$$t_{ij} = t_{(j-i)}. \tag{A.1b}$$

Eine spezielle Toeplitz-Matrix ist die folgende:

Definition A.0.2 (periodische Toeplitz-Matrix): Eine quadratische Toeplitz-Matrix der Form

$$\boldsymbol{C} = \begin{pmatrix} c_0 & c_{n-1} & c_{n-2} & \dots & c_1 \\ c_1 & c_0 & c_{n-1} & \dots & c_2 \\ c_2 & c_1 & c_0 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & c_{n-1} \\ c_{n-1} & c_{n-2} & \dots & c_1 & c_0 \end{pmatrix} \tag{A.2a}$$

nennt man *periodische Toeplitz-Matrix*, *zirkulante Matrix* oder *Zirkulante*. Die Elemente c_{ij} von $\boldsymbol{C} \in \mathbb{R}^{n \times n}$ hängen also nur von der Differenz $(i - j) \bmod n$ ab, d.h.

$$c_{ij} = c_{(i-j) \bmod n}. \tag{A.2b}$$

Wie man (A.2a) leicht entnehmen kann, ist eine zirkulante Matrix bereits durch eine Zeile bzw. Spalte eindeutig festgelegt. Wir definieren daher mit den Bezeichnungen aus (A.2a)

$$\boldsymbol{C} =: \operatorname{circ}(c_0, c_1, \dots, c_{n-1}). \tag{A.2c}$$

Aufgrund ihrer Struktur besitzen zirkulante Matrizen gewisse positive Eigenschaften.

Definition A.0.3 (DFT-Matrix): Die Matrix $\boldsymbol{F} \in \mathbb{C}^{n\times n}$ mit den Spalten

$$f_k = (\omega_k^0, \dots, \omega_k^{n-1})^\top \quad \text{mit} \quad \omega_k = e^{\mathrm{i}\frac{2\pi}{n}(k-1)} \tag{A.3}$$

heißt *Matrix der diskreten Fouriertransformation (DFT).*

Satz A.0.4 (Spektralsatz für Zirkulanten): *Eine Matrix $\boldsymbol{C}$ ist genau dann zirkulant, wenn es eine Diagonalmatrix $\boldsymbol{\Lambda}$ gibt, so dass $\boldsymbol{C} = \frac{1}{n}\boldsymbol{F^*\Lambda F}$ gilt. Es gilt dann $\boldsymbol{\Lambda} = \operatorname{diag}(\boldsymbol{F}\cdot(c_0, c_1, \dots, c_{n-1})^\top)$.*

Bemerkung A.0.5: Die Produkte $\boldsymbol{Fx}, \boldsymbol{F}^*\boldsymbol{x}$ und $\boldsymbol{F}^{-1}\boldsymbol{x}$ können nach dem Algorithmus der schnellen Fouriertransformation (FFT) von Cooley und Tukey Cooley und Tukey [8] mit $\frac{3}{2}n\log_2 n$ komplexen Operationen für $n = 2^p, p \in \mathbb{N}$ realisiert werden.

Folgerung A.0.6: *Satz A.0.4 in Kombination mit Bemerkung A.0.5 liefert uns eine Methode für eine schnelle Matrix-Vektor-Multiplikation für zirkulante Matrizen. Sei dazu $\boldsymbol{C} \in \mathbb{R}^{n\times n}$, mit $n = 2^p, p \in \mathbb{N}$, eine Zirkulante. Die Berechnung eines Produktes $\boldsymbol{y} = \boldsymbol{Cx}$, $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$, führen wir wie folgt durch:*

$$\begin{aligned}
&\tilde{\boldsymbol{x}} = \boldsymbol{Fx} && \textit{FFT: } 3/2\, n\log_2 n\\
&\boldsymbol{\Lambda} = \frac{1}{n}\operatorname{diag}(\boldsymbol{F}\cdot(c_0, c_1, \dots, c_{n-1})^\top) && \textit{FFT + Multiplikation: } 3/2\, n\log_2 n + n\\
&\tilde{\boldsymbol{y}} = \boldsymbol{\Lambda}\tilde{\boldsymbol{x}} && \textit{Multiplikation Diagonalmatrix: } n\\
&\boldsymbol{y} = \boldsymbol{F}^*\tilde{\boldsymbol{y}} && \textit{FFT: } 3/2\, n\log_2 n
\end{aligned}$$

und haben somit einen Gesamtaufwand von $\frac{9}{2}n\log_2 n + 2n$ Operationen.

Bemerkung A.0.7: Falls für die Zirkulante $\boldsymbol{C} = \operatorname{circ}(c_0, \dots, c_{n-1}) \in \mathbb{R}^{n\times n}$ und den Vektor $\boldsymbol{x} = (x_1, \dots, x_n)^\top$ gilt $n \neq 2^p$, können wir so vorgehen, dass wir $\boldsymbol{C}$ und $\boldsymbol{x}$ einbetten in $\tilde{\boldsymbol{C}} = \operatorname{circ}(c_0, \dots, c_{n-1}, 0, \dots, 0) \in \mathbb{R}^{\tilde{n}\times\tilde{n}}$ und $\tilde{\boldsymbol{x}} = (x_1, \dots, x_n, 0, \dots, 0)^\top \in \mathbb{R}^{\tilde{n}}$, wobei $2^p \leq n < \tilde{n} = 2^{p+1}$. Dann entsprechen die ersten n Einträge des Produkts $\tilde{\boldsymbol{C}}\tilde{\boldsymbol{x}}$ ursprünglichen $\boldsymbol{Cx}$. Wir erhalten auf diese Weise im schlimmsten Fall, gemäß Folgerung A.0.6, einen Aufwand von $9n\log_2 n + 13n$.

Bemerkung A.0.8: Wir können jede nichtperiodische Toeplitz-Matrix $\boldsymbol{T}$ in eine Zirkulante einbetten. Sei dazu $\boldsymbol{T} \in m \times n$ mit den Bezeichnungen aus (A.1a) gegeben und sei $\hat{m} = \max(m, n)$. Gemäß (A.2c), ist die $2\hat{m} \times 2\hat{m}$ Matrix

$$\operatorname{circ}(t_0, t_1, \dots, t_{m-1}, *, \dots, *, t_{-(n-1)}, t_{-(n-2)}, \dots, t_{-1}) \tag{A.4}$$

eine Zirkulante. Für $*$ können beliebige Werte eingefügt werden, vorzugsweise Null, falls $\hat{m}$ in der Form $\hat{m} := 2^p$, $p \in \mathbb{N}$ vorliegt und man diese Form für die zirkulante Matrix beibehalten möchte, was z.B. bei Berechnungen mittels FFT günstig ist.

Folgerung A.0.9: *Mit der Einbettung aus Bemerkung A.0.8 können wir auch nichtperiodische Toeplitz-Matrizen schnell auf einen Vektor $\boldsymbol{x}$ anwenden, indem wir die eingebettete Matrix auf einen Vektor $\tilde{\boldsymbol{x}} := (x_1, x_2, \dots, x_n, 0, \dots, 0)^\top$ anwenden, gemäß Folgerung A.0.6 und wir nur die ersten m Einträge des Ergebnisvektors verwenden, was ebenfalls einen Aufwand $O(\hat{m}\log\hat{m})$ erfordert.*

B. Approximationstheorie

Zum besseren Verständnis wollen wir hier noch einmal eine Übersicht über wichtige Grundlagen aus der Approximationstheorie geben, die auch in der Arbeit Verwendung finden.

Ein Schwerpunkt in der Approximationstheorie ist die *Interpolationsaufgabe*. Zu vorgegebenen paarweise verschiedenen Stützwerten $x_0, \dots, x_k \in D \subset \mathbb{R}$ und einer Funktion f von der zumindest die Funktionswerte an den Stützstellen bekannt sind, suchen wir eine Funktion $P \in V_k$ für welche gilt

$$P(x_i) = f(x_i) \quad \text{für} \quad i = 1, \dots, k, \tag{B.1}$$

wobei V_k ein k-dimensionaler linearer Unterraum von $C(D)$ sei. Wir sprechen hier auch von *(linearer) Interpolation*. Die Aufgabe (B.1) braucht im Allgemeinen keine Lösung zu besitzen oder zumindest keine eindeutige, jedoch sind Lösbarkeit und Eindeutigkeit äquivalent.

Zunächst benötigen wir die folgende Definition.

Definition B.0.1 (Lagrange-Polynome): Das *i-te Lagrange-Polynom L_i^k vom Grade k* bzgl. der paarweise verschiedenen Stützstellen $x_0, \dots, x_k$ ist definiert gemäß

$$L_i^k(x) := \prod_{j=0, i \neq j}^{k} \frac{x - x_j}{x_i - x_j} \quad \text{für} \quad i = 0, \dots, k. \tag{B.2}$$

Insbesondere gilt

$$L_i^k(x_j) = \delta_{ij}. \tag{B.3}$$

Sei nun die eindeutige Lösbarkeit der Interpolationsaufgabe im Folgenden vorausgesetzt, dann ist das *Lagrangesche Interpolationspolynom* gut für die Darstellung von P geeignet.

Definition B.0.2 (Lagrangesches Interpolationspolynom): Es ist

$$P_k(x) := \sum_{i=0}^{k} f(x_i) L_i^k(x) \tag{B.4}$$

das k-te *Lagrangesches Interpolationspolynom* bzgl. der paarweise verschiedenen Stützstellen $x_0, \dots, x_k \in D$ und der Funktion f, dabei ist L_i^k gerade das i-te Lagrange-Polynom vom Grade k.

Besitzt die Funktion f Ableitungen bis zu einer gewissen Ordnung, läßt sich ebenfalls die Güte der Interpolation angeben.

Lemma B.0.3: *Mit den Bezeichnungen aus Definition B.0.2 und für $f \in C^{k+1}(D)$ gilt die Darstellung*

$$f(x) = P_k(x) + \frac{f^{(k+1)}(\xi(x))}{(k+1)!} \cdot (x - x_0) \cdot (x - x_1) \cdot \ldots \cdot (x - x_k) \tag{B.5}$$

für ein $\xi(x)$ zwischen $x_0, x_1, \ldots, x_k$ und x.

Die Wahl der Stützstellen spielt ebenfalls eine große Rolle. In der Arbeit haben wir bereits erwähnt, dass wir gerade die *Tschebyscheff-Knoten* benutzen, welche eine günstige Abschätzung (gemäß Bemerkung B.0.8) der *Lebesgue-Konstante* Λ_k erlauben, welche wir in Lemma 3.7.33 für die dortige Abschätzung benötigen. Wir beschränken uns dazu im Folgenden auf das Referenzintervall $[-1, 1]$.

Definition B.0.4 (Tschebyscheff-Polynome): Das *Tschebyscheff-Polynom T_k vom Grad k* ist definiert gemäß

$$T_k(x) := \cos(k \arccos x), \quad x \in [-1, 1]. \tag{B.6}$$

Bemerkung B.0.5: a) Tschebyscheff-Polynome erfüllen die dreigliedrige Rekursionsformel

$$T_{k+1}(x) = 2\,x\,T_k(x) - T_{k-1}(x), \quad k \geq 1; \quad T_0(x) = 1, T_1(x) = x. \tag{B.7}$$

b) Die Nullstellen x_i, $i = 1, \ldots, k$, des Tschebyscheff-Polynoms T_k auf dem Intervall $[-1, 1]$ die so genannten *Tschebyscheff-Knoten*, sind gegeben durch

$$x_i = \cos\left(\frac{2i-1}{k}\frac{\pi}{2}\right). \tag{B.8}$$

Definition B.0.6 (Lebesgue-Funktion): Die *Lebesgue-Funktion* λ_k bzgl. der paarweise verschiedenen Stützstellen $x_0, \ldots, x_k \in [-1, 1]$ ist definiert gemäß:

$$\lambda_k(x) := \sum_{i=0}^{k} |L_i^k(x)|, \tag{B.9}$$

dabei ist L_i^k gerade das i-te Lagrange-Polynome bzgl. $x_0, \ldots, x_k$.

Definition B.0.7 (Lebesgue-Konstante): Die Zahl

$$\Lambda_k := \max_{x \in [-1,1]} \lambda_k(x) = \|\lambda_k\|_{\infty,[-1,1]} \tag{B.10}$$

heißt *Lebesgue-Konstante*.

Bemerkung B.0.8: Für die Kombination Lagrange-Polynome und Tschebyscheff-Knoten auf dem Intervall $[-1, 1]$, genügt Λ_k gerade der Abschätzung

$$\Lambda_k \leq \frac{2}{\pi} \log k + 1 \leq k + 1. \tag{B.11}$$

Beweis: Siehe z.B. Davis [11]. □

C. Integration

Wir haben noch einmal die wichtigsten Eigenschaften der Trapezmethode zusammengetragen. Seien dazu im Folgenden $a, b, c, d \in \mathbb{R}$ und weiterhin $a < b$ sowie $c < d$.

Lemma C.0.1 (Trapezmethode in 1D): *Sei $f \in C^2([a,b])$, dann gilt*

$$\int_a^b f(x)\,dx = (b-a)\frac{f(a)+f(b)}{2} - (b-a)^3\frac{f''(\xi)}{12} \tag{C.1}$$

für ein $\xi \in (a,b)$. Dabei ist durch den ersten Term der rechten Seite gerade die Trapezmethode in 1D charakterisiert und durch den zweiten Term das Fehlerglied.

Beweis: Mit der Darstellung aus dem Lemma B.0.3 haben wir mit $a = x_0$ und $b = x_1$

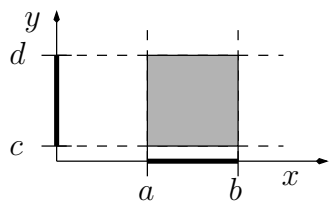
$$\begin{aligned}\int_a^b f(x) &= \int_a^b f(a)\frac{x-b}{a-b} + f(b)\frac{x-a}{b-a} + \frac{f''(\xi)}{2}(x-a)(x-b)dx \\ &= (b-a)\frac{f(a)}{2} + (b-a)\frac{f(b)}{2} - \frac{f''(\xi)}{2}\frac{(b-a)^3}{6} \\ &= (b-a)\frac{f(a)+f(b)}{2} - (b-a)^3\frac{f''(\xi)}{12}.\end{aligned} \tag{C.2}$$

□

Lemma C.0.2 (Trapezmethode in 2D): *Sei $f \in C^2([a,b]\times[c,d])$, dann gilt*

$$\begin{aligned}\int_a^b\int_c^d f(x,y)\,dy\,dx =& (d-c)(b-a)\frac{f(a,c)+f(a,d)+f(b,c)+f(b,d)}{4} \\ &- \frac{(d-c)(b-a)}{12}\left[(b-a)^2\frac{\partial^2}{\partial x^2}f(\xi,\zeta) + (d-c)^2\frac{\partial^2}{\partial y^2}f(\tilde{\xi},\tilde{\zeta})\right]\end{aligned}$$

für $\xi, \tilde{\xi} \in [a,b]$ und $\zeta, \tilde{\zeta} \in [c,d]$. Dabei ist durch den ersten Term der rechten Seite gerade die Trapezmethode in 2D charakterisiert und durch den zweiten Term das Fehlerglied. Siehe hierzu auch nebenstehende Skizze.



Beweis: Wir wenden zunächst Lemma C.0.1 auf das innere Integral an und behandeln die Variable x als Konstante:

$$\int_c^d f(x,y)\,dy = (d-c)\frac{f(x,c)+f(x,d)}{2} - \frac{(d-c)^3}{12}\frac{\partial^2}{\partial y^2}f(x,\hat{\zeta}). \tag{C.3}$$

Auf (C.3) wenden wir nun das äußere Integral gemäß Lemma C.0.1 an und erhalten

$$\begin{aligned}\int_a^b \int_c^d f(x,y)\,dx\,dy =& (d-c)(b-a)\frac{f(a,c)+f(a,d)+f(b,c)+f(b,d)}{4} \\ &- \frac{(d-c)(b-a)^3}{24}\left[\frac{\partial^2}{\partial x^2}f(\hat{\xi},c)+\frac{\partial^2}{\partial x^2}f(\hat{\hat{\xi}},d)\right] \\ &- \frac{(d-c)^3}{12}\int_a^b \frac{\partial^2}{\partial y^2}f(x,\tilde{\zeta})\,dx.\end{aligned} \tag{C.4}$$

Mit der Annahme $f \in C^2([a,b]\times[c,d])$ können wir auf den zweiten Term der rechten Seite von (C.4) den Zwischenwertsatz anwenden und auf den dritten Term den Mittelwertsatz der Integralrechnung. Somit erhalten wir

$$\frac{(d-c)(b-a)^3}{24}\left[\frac{\partial^2}{\partial x^2}f(\hat{\xi},c)+\frac{\partial^2}{\partial x^2}f(\hat{\hat{\xi}},d)\right] = \frac{(d-c)(b-a)^3}{12}\frac{\partial^2}{\partial x^2}f(\xi,\zeta) \tag{C.5}$$

und

$$\frac{(d-c)^3}{12}\int_a^b \frac{\partial^2}{\partial y^2}f(x,\tilde{\zeta})\,dx = \frac{(d-c)^3(b-a)}{12}\frac{\partial^2}{\partial y^2}f(\tilde{\xi},\tilde{\zeta}), \tag{C.6}$$

daraus folgt die Behauptung. □

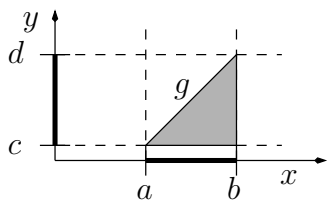
Lemma C.0.3 (Modifizierte Trapezmethode in 2D): *Sei $f \in C^2([a,b]\times[c,d])$ und die Funktion g definiert gemäß:*

$$g(x) = x\frac{d-c}{b-a} - a\frac{d-c}{b-a} + c. \tag{C.7}$$

Dann gilt:

$$\begin{aligned}\int_a^b \int_c^{g(x)} f(x,y)\,dy\,dx =& (d-c)(b-a)\,\frac{3\,f(a,c)+f(a,d)+5\,f(b,c)+3\,f(b,d)}{24} \\ &- \frac{(d-c)(b-a)}{24}\left[(b-a)^2\frac{\partial^2}{\partial x^2}f(\xi,\zeta)+(d-c)^2\frac{\partial^2}{\partial y^2}f(\tilde{\xi},\tilde{\zeta})\right]\end{aligned}$$

für $\xi,\tilde{\xi} \in [a,b]$ und $\zeta,\tilde{\zeta} \in [c,d]$ (wobei der Integrationsbereich gerade dem Dreieck aus nebenstehender Skizze entspricht). Hierbei sei wieder durch den ersten Term der rechten Seite die Methode beschrieben und durch den zweiten das Fehlerglied.



Beweis: Der Beweis wird analog zu dem von Lemma C.0.2 geführt, man ersetze hier nur die Variable d durch die Funktion g. □

Folgerung C.0.4: *Analoge Methoden und Fehler gelten natürlich für andere mögliche Dreiecksgebiete.*

D. Ergänzungen zu den numerischen Experimenten

D.1. Linearer Operator

D.1.1. Zeiten für das Aufstellen

Dimension	128		256		512		1024	
$k = 1$	2.32e−03	4.17%	4.84e−03	2.18%	9.45e−03	1.07%	1.86e−02	0.52%
$k = 2$	3.42e−03	6.15%	6.60e−03	2.98%	1.28e−02	1.45%	2.63e−02	0.74%
$k = 3$	4.20e−03	7.55%	8.32e−03	3.76%	1.70e−02	1.92%	3.39e−02	0.95%
$k = 4$	5.41e−03	9.73%	1.11e−02	5.01%	2.18e−02	2.46%	4.25e−02	1.19%
$k = 5$	6.68e−03	12.01%	1.34e−02	6.05%	2.66e−02	3.00%	5.16e−02	1.45%
$k = 6$	7.95e−03	14.30%	1.57e−02	7.09%	3.22e−02	3.64%	6.20e−02	1.74%
$k = 7$	8.98e−03	16.15%	1.78e−02	8.03%	3.59e−02	4.06%	7.27e−02	2.04%
$k = 8$	1.02e−02	18.34%	2.12e−02	9.57%	4.28e−02	4.84%	8.31e−02	2.33%
$k = 9$	1.26e−02	22.66%	2.50e−02	11.28%	4.88e−02	5.51%	9.50e−02	2.66%
$k = 10$	1.62e−02	29.13%	2.71e−02	12.23%	5.48e−02	6.19%	1.07e−01	3.00%
$k = 11$	1.49e−02	26.80%	3.03e−02	13.68%	6.01e−02	6.79%	1.18e−01	3.31%
$k = 12$	1.62e−02	29.13%	3.29e−02	14.85%	6.67e−02	7.54%	1.32e−01	3.70%
vollbesetzt	5.56e−02	100.0%	2.22e−01	100.0%	8.85e−01	100.0%	3.57e+00	100.0%
Dimension	2048		4096		8192		16384	
$k = 1$	3.59e−02	0.25%	8.12e−02	0.13%	1.64e−01	0.06%	3.45e−01	0.03%
$k = 2$	5.28e−02	0.37%	1.09e−01	0.18%	2.36e−01	0.09%	4.65e−01	0.05%
$k = 3$	6.84e−02	0.48%	1.45e−01	0.24%	2.85e−01	0.11%	5.88e−01	0.06%
$k = 4$	8.69e−02	0.62%	1.79e−01	0.29%	3.72e−01	0.15%	7.35e−01	0.07%
$k = 5$	1.07e−01	0.76%	2.17e−01	0.36%	4.35e−01	0.17%	8.68e−01	0.09%
$k = 6$	1.28e−01	0.96%	2.61e−01	0.43%	5.19e−01	0.21%	1.02e+00	0.11%
$k = 7$	1.50e−01	1.06%	3.04e−01	0.50%	5.99e−01	0.24%	1.18e+00	0.12%
$k = 8$	1.70e−01	1.20%	3.46e−01	0.56%	6.76e−01	0.27%	1.35e+00	0.13%
$k = 9$	1.89e−01	1.34%	3.91e−01	0.64%	7.54e−01	0.30%	1.52e+00	0.15%
$k = 10$	2.14e−01	1.51%	4.36e−01	0.71%	8.72e−01	0.34%	1.70e+00	0.17%
$k = 11$	2.37e−01	1.68%	4.82e−01	0.79%	9.64e−01	0.38%	1.90e+00	0.19%
$k = 12$	2.64e−01	1.87%	5.41e−01	0.88%	1.06e+00	0.42%	2.11e+00	0.21%
vollbesetzt	1.41e+01	100.0%	6.12e+01	100.0%	2.52e+02	100.0%	1.01e+03	100.0%

Tabelle D.1.: Linearer Operator: Zeiten für das Aufstellen in [sec].

D.1.2. Zeiten für die Auswertung

Dimension	128		256		512		1024	
$k = 1$	1.39e−05	15.66%	2.11e−05	1.76%	3.84e−05	0.80%	7.34e−05	0.38%
$k = 2$	1.65e−05	18.53%	3.03e−05	2.54%	5.65e−05	1.17%	1.17e−04	0.61%
$k = 3$	2.07e−05	23.22%	3.79e−05	3.17%	7.36e−05	1.53%	1.44e−04	0.76%
$k = 4$	2.56e−05	28.73%	4.77e−05	3.99%	9.09e−05	1.89%	2.44e−04	1.28%
$k = 5$	2.82e−05	31.62%	5.50e−05	4.60%	1.08e−04	2.23%	3.52e−04	1.84%
$k = 6$	3.26e−05	36.65%	6.33e−05	5.29%	1.33e−04	2.75%	4.54e−04	2.38%
$k = 7$	3.65e−05	41.05%	7.20e−05	6.02%	1.45e−04	3.01%	6.55e−04	3.43%
$k = 8$	4.07e−05	45.74%	8.00e−05	6.69%	1.63e−04	3.38%	8.34e−04	4.37%
$k = 9$	4.18e−05	46.92%	8.65e−05	7.23%	1.93e−04	4.00%	1.00e−03	5.26%
$k = 10$	4.46e−05	50.11%	9.43e−05	7.88%	2.82e−04	5.85%	1.14e−03	5.95%
$k = 11$	4.86e−05	54.54%	1.02e−04	8.49%	3.51e−04	7.29%	1.35e−03	7.07%
$k = 12$	5.15e−05	57.82%	1.09e−04	9.14%	3.49e−04	7.26%	1.53e−03	8.03%
vollbesetzt	8.90e−05	100.0%	1.20e−03	100.0%	4.81e−03	100.0%	1.91e−02	100.0%
Dimension	2048		4096		8192		16384	
$k = 1$	1.54e−04	0.19%	5.99e−04	0.17%	1.52e−03	0.10%	3.51e−03	0.05%
$k = 2$	3.68e−04	0.46%	1.35e−03	0.38%	2.46e−03	0.17%	5.46e−03	0.08%
$k = 3$	6.84e−04	0.85%	1.89e−03	0.54%	3.22e−03	0.22%	7.31e−03	0.11%
$k = 4$	1.01e−03	1.26%	2.38e−03	0.68%	4.06e−03	0.28%	9.17e−03	0.13%
$k = 5$	1.32e−03	1.64%	2.95e−03	0.84%	4.85e−03	0.34%	1.12e−02	0.16%
$k = 6$	1.68e−03	2.09%	3.44e−03	0.98%	5.68e−03	0.40%	1.42e−02	0.21%
$k = 7$	1.95e−03	2.43%	3.70e−03	1.06%	6.14e−03	0.43%	1.58e−02	0.23%
$k = 8$	2.20e−03	2.74%	4.18e−03	1.19%	7.02e−03	0.49%	1.68e−02	0.24%
$k = 9$	2.46e−03	3.06%	4.73e−03	1.35%	7.89e−03	0.55%	1.87e−02	0.27%
$k = 10$	2.72e−03	3.39%	5.06e−03	1.44%	8.82e−03	0.62%	2.01e−02	0.29%
$k = 11$	2.95e−03	3.68%	5.60e−03	1.60%	9.26e−03	0.65%	2.20e−02	0.32%
$k = 12$	3.20e−03	3.99%	6.02e−03	1.72%	1.02e−02	0.71%	2.40e−02	0.35%
vollbesetzt	8.02e−02	100.0%	3.51e−01	100.0%	1.43e+00	100.0%	6.87e+00	100.0%

Tabelle D.2.: Linearer Operator: Zeiten für die Auswertung in [sec].

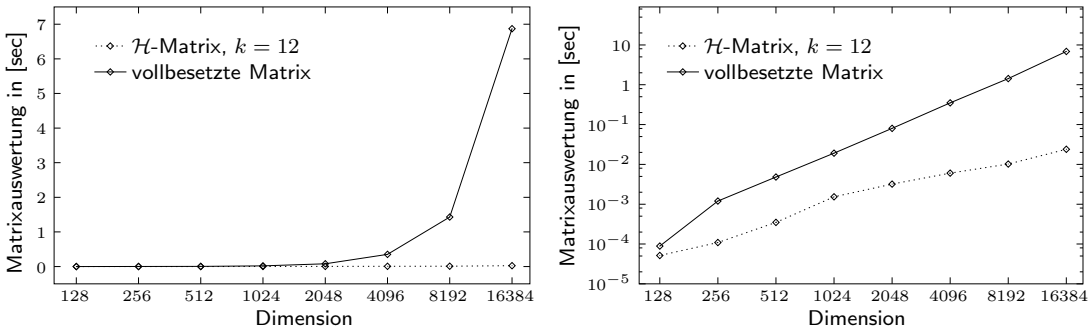


Abbildung D.1.: Linearer Operator: Verhalten der Zeiten für die Auswertung.

D.1.3. Approximationsfehler

Dimension	128		256		512		1024	
$k = 1$	1.943e−01	–	1.950e−01	–	1.954e−01	–	1.955e−01	–
$k = 2$	4.765e−02	0.24	4.770e−02	0.24	4.772e−02	0.24	4.773e−02	0.24
$k = 3$	1.033e−02	0.22	1.039e−02	0.22	1.041e−02	0.22	1.042e−02	0.22
$k = 4$	2.717e−03	0.26	2.730e−03	0.26	2.736e−03	0.26	2.739e−03	0.26
$k = 5$	6.303e−04	0.23	6.344e−04	0.23	6.360e−04	0.23	6.368e−04	0.23
$k = 6$	1.537e−04	0.24	1.549e−04	0.24	1.553e−04	0.24	1.556e−04	0.24
$k = 7$	3.749e−05	0.24	3.784e−05	0.24	3.797e−05	0.24	3.803e−05	0.24
$k = 8$	9.118e−06	0.24	9.223e−06	0.24	9.262e−06	0.24	9.277e−06	0.24
$k = 9$	2.222e−06	0.24	2.251e−06	0.24	2.262e−06	0.24	2.266e−06	0.24
$k = 10$	5.373e−07	0.24	5.461e−07	0.24	5.491e−07	0.24	5.503e−07	0.24
$k = 11$	1.305e−07	0.24	1.329e−07	0.24	1.338e−07	0.24	1.341e−07	0.24
$k = 12$	3.157e−08	0.24	3.223e−08	0.24	3.246e−08	0.24	3.254e−08	0.24
Dimension	2048		4096		8192		16384	
$k = 1$	1.956e−01	–	1.957e−01	–	1.957e−01	–	1.957e−01	–
$k = 2$	4.774e−02	0.24	4.774e−02	0.24	4.774e−02	0.24	4.773e−02	0.24
$k = 3$	1.043e−02	0.22	1.043e−02	0.22	1.043e−02	0.22	1.043e−02	0.22
$k = 4$	2.740e−03	0.26	2.741e−03	0.26	2.741e−03	0.26	2.741e−03	0.26
$k = 5$	6.371e−04	0.23	6.373e−04	0.23	6.373e−04	0.23	6.371e−04	0.23
$k = 6$	1.556e−04	0.24	1.557e−04	0.24	1.557e−04	0.24	1.556e−04	0.24
$k = 7$	3.805e−05	0.24	3.806e−05	0.24	3.807e−05	0.24	3.805e−05	0.24
$k = 8$	9.284e−06	0.24	9.287e−06	0.24	9.289e−06	0.24	9.284e−06	0.24
$k = 9$	2.268e−06	0.24	2.269e−06	0.24	2.269e−06	0.24	2.268e−06	0.24
$k = 10$	5.508e−07	0.24	5.510e−07	0.24	5.511e−07	0.24	5.507e−07	0.24
$k = 11$	1.342e−07	0.24	1.343e−07	0.24	1.343e−07	0.24	1.342e−07	0.24
$k = 12$	3.257e−08	0.24	3.258e−08	0.24	3.259e−08	0.24	3.256e−08	0.24

Tabelle D.3.: Linearer Operator: Approximationsfehler ε_{rel} für den Emulsionskern (Dispersion) mit $c_{\varphi_1} = c_{\varphi_2} = c_{\text{br},3} = c_{\text{br},4} = 1.0$.

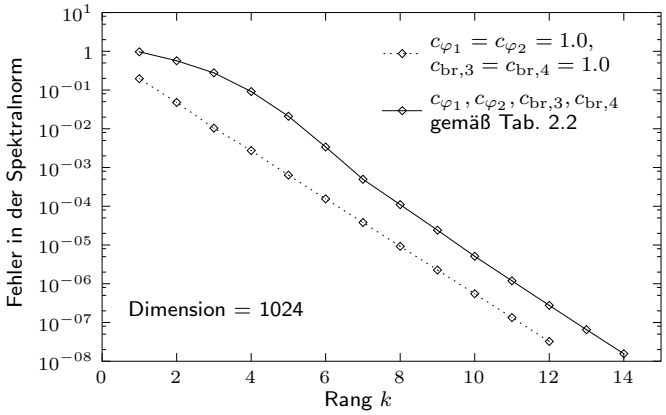


Abbildung D.2.: Linearer Operator: Fehlerverhalten.

D.2. Quasilinearer Operator

D.2.1. Speicherbedarf

Dimension	128		256		512		1024	
$k = 1$	0.176	140.8%	0.199	39.80%	0.250	12.50%	0.344	4.30%
$k = 2$	0.176	140.8%	0.199	39.80%	0.250	12.50%	0.344	4.30%
$k = 3$	0.180	144.0%	0.203	40.60%	0.258	12.90%	0.363	4.54%
$k = 4$	0.184	147.2%	0.215	43.00%	0.277	13.85%	0.402	5.02%
$k = 5$	0.184	147.2%	0.219	43.80%	0.289	14.45%	0.430	5.38%
$k = 6$	0.188	150.4%	0.230	46.00%	0.309	15.45%	0.469	5.86%
$k = 7$	0.195	156.0%	0.238	47.60%	0.328	16.40%	0.504	6.30%
$k = 8$	0.199	159.2%	0.246	49.20%	0.348	17.40%	0.543	6.79%
$k = 9$	0.199	159.2%	0.254	50.80%	0.359	17.95%	0.574	7.18%
$k = 10$	0.203	162.4%	0.262	52.40%	0.379	18.95%	0.613	7.66%
$k = 11$	0.207	165.6%	0.270	54.00%	0.398	19.90%	0.652	8.15%
$k = 12$	0.211	168.8%	0.277	55.40%	0.414	20.70%	0.688	8.60%
vollbesetzt	0.125	100.0%	0.5	100.0%	2	100.0%	8	100.0%
Dimension	2048		4096		8192		16384	
$k = 1$	0.535	1.67%	0.922	0.72%	1.684	0.33%	3.238	0.16%
$k = 2$	0.535	1.67%	0.922	0.72%	1.684	0.33%	3.234	0.16%
$k = 3$	0.574	1.79%	0.996	0.78%	1.855	0.36%	3.570	0.17%
$k = 4$	0.652	2.04%	1.152	0.90%	2.168	0.42%	4.195	0.20%
$k = 5$	0.707	2.21%	1.270	0.99%	2.398	0.47%	4.664	0.23%
$k = 6$	0.785	2.45%	1.426	1.11%	2.711	0.53%	5.289	0.26%
$k = 7$	0.863	2.70%	1.598	1.25%	3.047	0.60%	5.934	0.29%
$k = 8$	0.941	2.94%	1.750	1.37%	3.359	0.66%	6.559	0.32%
$k = 9$	1.008	3.15%	1.887	1.47%	3.629	0.71%	7.106	0.35%
$k = 10$	1.082	3.38%	2.039	1.59%	3.941	0.77%	7.727	0.38%
$k = 11$	1.160	3.62%	2.195	1.72%	4.250	0.83%	8.352	0.41%
$k = 12$	1.234	3.86%	2.352	1.84%	4.562	0.89%	8.977	0.44%
vollbesetzt	32	100.0%	128	100.0%	512	100.0%	2048	100.0%

Tabelle D.4.: Quasilinearer Operator: Benötigter Speicher in Megabyte.

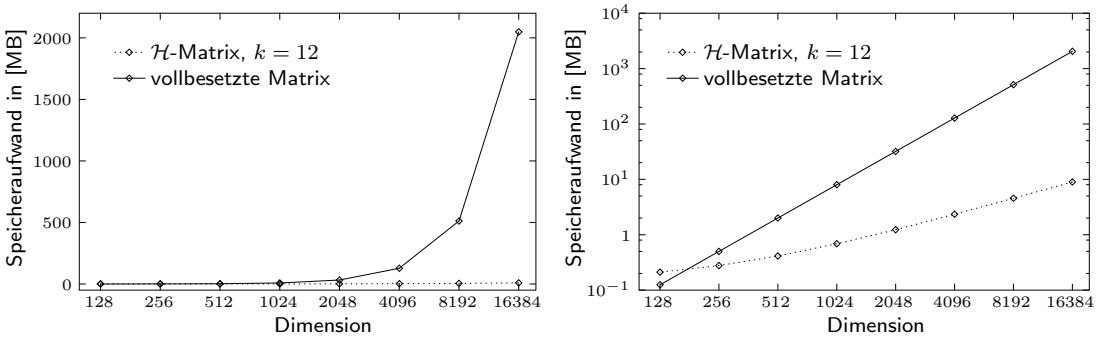


Abbildung D.3.: Quasilinearer Operator: Speicherbedarf.

D.2.2. Zeiten für das Aufstellen

Dimension	128		256		512		1024	
$k = 1$	1.00e−02	2.13%	4.00e−02	2.17%	7.00e−02	0.95%	1.40e−01	0.48%
$k = 2$	2.00e−02	4.25%	4.00e−02	2.17%	9.00e−02	1.22%	1.70e−01	0.58%
$k = 3$	3.00e−02	6.38%	5.00e−02	2.72%	1.00e−01	1.36%	2.00e−01	0.68%
$k = 4$	4.00e−02	8.51%	6.00e−02	3.26%	1.20e−01	1.6%	2.30e−01	0.78%
$k = 5$	4.00e−02	8.51%	8.00e−02	4.35%	1.40e−01	1.90%	2.60e−01	0.89%
$k = 6$	4.00e−02	8.51%	9.00e−02	4.89%	1.50e−01	2.04%	2.90e−01	0.99%
$k = 7$	5.00e−02	10.64%	8.00e−02	4.35%	1.60e−01	2.18%	3.20e−01	1.09%
$k = 8$	6.00e−02	12.77%	9.00e−02	4.89%	1.80e−01	2.45%	3.80e−01	1.30%
$k = 9$	9.00e−02	19.15%	1.40e−01	7.61%	2.40e−01	3.26%	4.20e−01	1.43%
$k = 10$	9.00e−02	19.15%	1.50e−01	8.15%	2.50e−01	3.40%	4.40e−01	1.50%
$k = 11$	1.00e−01	21.28%	1.50e−01	8.15%	2.60e−01	3.54%	4.80e−01	1.64%
$k = 12$	1.10e−01	23.40%	1.60e−01	8.70%	2.70e−01	3.67%	6.60e−01	2.25%
vollbesetzt	4.70e−01	100.0%	1.84e+00	100.0%	7.35e+00	100.0%	2.93e+01	100.0%
Dimension	2048		4096		8192		16384	
$k = 1$	3.00e−01	0.26%	5.80e−01	0.12%	1.16e+00	0.06%	2.41e+00	0.03%
$k = 2$	3.40e−01	0.29%	6.90e−01	0.14%	1.37e+00	0.07%	2.93e+00	0.04%
$k = 3$	4.00e−01	0.34%	8.00e−01	0.17%	1.61e+00	0.09%	3.46e+00	0.05%
$k = 4$	4.60e−01	0.39%	9.10e−01	0.19%	1.82e+00	0.10%	4.01e+00	0.05%
$k = 5$	5.20e−01	0.44%	1.03e+00	0.22%	2.06e+00	0.11%	4.57e+00	0.06%
$k = 6$	5.80e−01	0.50%	1.14e+00	0.24%	2.29e+00	0.12%	5.10e+00	0.07%
$k = 7$	6.40e−01	0.55%	1.27e+00	0.26%	2.52e+00	0.14%	5.66e+00	0.08%
$k = 8$	6.90e−01	0.59%	1.38e+00	0.29%	2.75e+00	0.15%	6.24e+00	0.08%
$k = 9$	7.90e−01	0.68%	1.54e+00	0.32%	3.03e+00	0.16%	6.84e+00	0.09%
$k = 10$	8.50e−01	0.73%	1.66e+00	0.35%	3.26e+00	0.17%	7.40e+00	0.10%
$k = 11$	9.80e−01	0.84%	1.78e+00	0.37%	3.52e+00	0.19%	8.01e+00	0.11%
$k = 12$	1.00e+00	0.86%	1.90e+00	0.40%	3.75e+00	0.20%	8.77e+00	0.12%
vollbesetzt	1.17e+02	100.0%	4.79e+02	100.0%	1.87e+03	100.0%	7.48e+03	100.0%

Tabelle D.5.: Quasilinearer Operator(Emulsionskern): Zeiten für das Aufstellen in [sec].

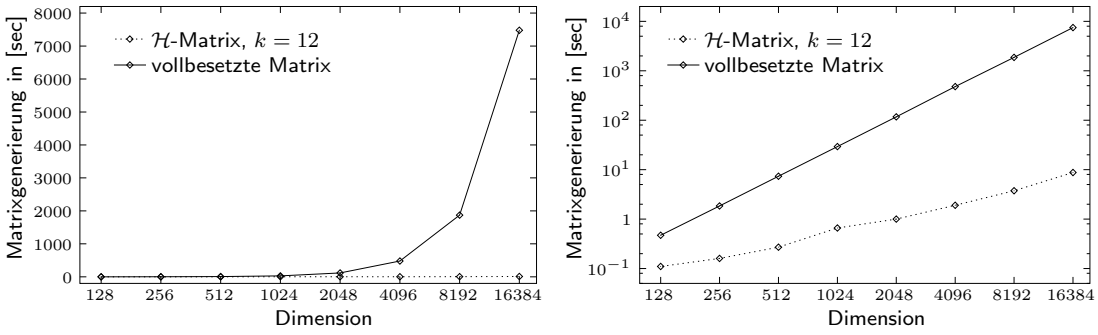


Abbildung D.4.: Quasilinearer Operator: Verhalten der Zeiten für das Aufstellen.

D.2.3. Zeiten für die Auswertung

Dimension	128		256		512		1024	
$k = 1$	1.39e−05	17.41%	2.21e−05	1.87%	3.84e−05	0.82%	7.22e−05	0.39%
$k = 2$	1.78e−05	23.96%	3.14e−05	2.66%	5.61e−05	1.20%	1.09e−04	0.58%
$k = 3$	2.21e−05	29.72%	4.01e−05	3.40%	7.33e−05	1.57%	1.70e−04	0.91%
$k = 4$	2.73e−05	36.70%	5.04e−05	4.27%	9.15e−05	1.96%	2.96e−04	1.58%
$k = 5$	3.08e−05	41.37%	5.79e−05	4.91%	1.09e−04	2.34%	5.20e−04	2.78%
$k = 6$	3.60e−05	48.31%	6.71e−05	5.69%	1.63e−04	3.49%	6.24e−04	3.34%
$k = 7$	3.99e−05	53.61%	7.61e−05	6.45%	1.88e−04	4.03%	7.91e−04	4.24%
$k = 8$	4.43e−05	59.43%	8.63e−05	7.32%	2.57e−04	5.50%	1.18e−03	6.30%
$k = 9$	4.62e−05	61.98%	9.23e−05	7.82%	2.93e−04	6.27%	1.21e−03	6.50%
$k = 10$	5.23e−05	70.21%	1.02e−04	8.65%	2.96e−04	6.35%	1.47e−03	7.88%
$k = 11$	5.42e−05	72.77%	1.07e−04	9.07%	4.83e−04	10.34%	1.40e−03	7.52%
$k = 12$	5.74e−05	77.05%	1.16e−04	9.79%	5.60e−04	12.00%	1.80e−03	9.65%
vollbesetzt	8.90e−05	100.0%	1.20e−03	100.0%	4.81e−03	100.0%	1.91e−02	100.0%
Dimension	2048		4096		8192		16384	
$k = 1$	1.54e−04	0.21%	6.78e−04	0.22%	1.96e−03	0.17%	4.60e−03	0.08%
$k = 2$	4.42e−04	0.60%	1.45e−03	0.48%	3.35e−03	0.28%	7.24e−03	0.13%
$k = 3$	7.74e−04	1.04%	1.92e−03	0.64%	4.37e−03	0.37%	1.01e−02	0.18%
$k = 4$	1.22e−03	1.65%	2.70e−03	0.90%	5.37e−03	0.45%	1.33e−02	0.24%
$k = 5$	1.40e−03	1.89%	2.94e−03	0.98%	6.89e−03	0.58%	1.58e−02	0.28%
$k = 6$	1.87e−03	2.52%	3.90e−03	1.29%	7.68e−03	0.65%	1.95e−02	0.35%
$k = 7$	1.93e−03	2.61%	4.31e−03	1.43%	8.83e−03	0.74%	2.13e−02	0.38%
$k = 8$	2.42e−03	3.26%	4.83e−03	1.60%	9.86e−03	0.83%	2.42e−02	0.43%
$k = 9$	2.45e−03	3.31%	5.45e−03	1.81%	1.12e−02	0.95%	2.73e−02	0.49%
$k = 10$	2.96e−03	4.00%	6.06e−03	2.01%	1.21e−02	1.02%	2.98e−02	0.53%
$k = 11$	2.88e−03	3.89%	6.51e−03	2.16%	1.33e−02	1.12%	3.24e−02	0.58%
$k = 12$	3.55e−03	4.79%	7.24e−03	2.40%	1.42e−02	1.20%	3.45e−02	0.61%
vollbesetzt	8.02e−02	100.0%	3.51e−01	100.0%	1.43e+00	100.0%	6.87e+00	100.0%

Tabelle D.6.: Quasilinearer Operator: Zeiten für die Auswertung in [sec].

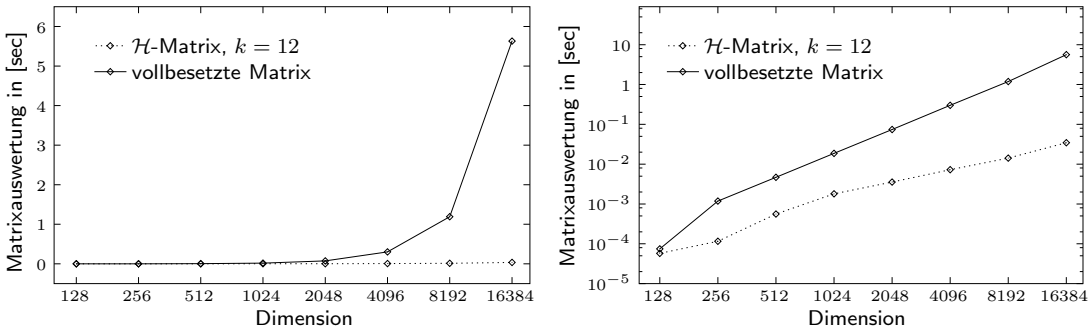


Abbildung D.5.: Quasilinearer Operator: Verhalten der Zeiten für die Auswertung.

D.2.4. Approximationsfehler

Dimension	128		256		512		1024	
$k = 1$	1.822e−01	–	1.844e−01	–	1.862e−01	–	1.876e−01	–
$k = 2$	5.026e−04	3e-3	3.568e−04	2e-3	2.533e−04	1e-3	1.799e−04	1e-3
$k = 3$	8.606e−05	0.17	6.128e−05	0.17	4.353e−05	0.17	3.093e−05	0.17
$k = 4$	1.480e−05	0.17	1.058e−05	0.17	7.528e−06	0.17	5.350e−06	0.17
$k = 5$	2.533e−06	0.17	1.822e−06	0.17	1.297e−06	0.17	9.223e−07	0.17
$k = 6$	4.319e−07	0.17	3.125e−07	0.17	2.229e−07	0.17	1.586e−07	0.17
$k = 7$	7.353e−08	0.17	5.353e−08	0.17	3.826e−08	0.17	2.723e−08	0.17
$k = 8$	1.251e−08	0.17	9.165e−09	0.17	6.566e−09	0.17	4.676e−09	0.17
$k = 9$	2.095e−09	0.17	1.567e−09	0.17	1.126e−09	0.17	8.025e−10	0.17
$k = 10$	3.572e−10	0.17	2.680e−10	0.17	1.931e−10	0.17	1.377e−10	0.17
$k = 11$	6.100e−11	0.17	4.583e−11	0.17	3.309e−11	0.17	2.363e−11	0.17
$k = 12$	1.044e−11	0.17	7.839e−12	0.17	5.672e−12	0.17	4.053e−12	0.17
Dimension	2048		4096		8192		16384	
$k = 1$	1.889e−01	–	1.885e−01	–	1.887e−01	–	1.890e−01	–
$k = 2$	1.279e−04	1e-3	9.017e−05	5e-4	6.384e−05	3e-4	4.524e−05	2e-4
$k = 3$	2.199e−05	0.17	1.550e−05	0.17	1.098e−05	0.17	7.778e−06	0.17
$k = 4$	3.803e−06	0.17	2.682e−06	0.17	1.899e−06	0.17	1.346e−06	0.17
$k = 5$	6.557e−07	0.17	4.624e−07	0.17	3.274e−07	0.17	2.320e−07	0.17
$k = 6$	1.127e−07	0.17	7.950e−08	0.17	5.628e−08	0.17	3.988e−08	0.17
$k = 7$	1.937e−08	0.17	1.366e−08	0.17	9.669e−09	0.17	6.852e−09	0.17
$k = 8$	3.326e−09	0.17	2.345e−09	0.17	1.661e−09	0.17	1.177e−09	0.17
$k = 9$	5.709e−10	0.17	4.026e−10	0.17	2.850e−10	0.17	2.020e−10	0.17
$k = 10$	9.799e−11	0.17	6.911e−11	0.17	4.893e−11	0.17	3.467e−11	0.17
$k = 11$	1.682e−11	0.17	1.186e−11	0.17	8.397e−12	0.17	–	
$k = 12$	2.886e−12	0.17	2.035e−12	0.17	1.441e−12	0.17	–	

Tabelle D.7.: Quasilinearer Operator: Approximationsfehler ε_{rel} für den Smoluchowski-Kern mit $c_{\text{smol}} = 0.00000001$.

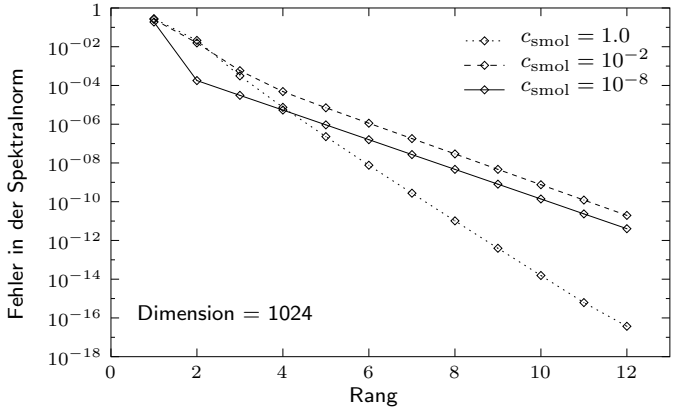


Abbildung D.6.: Quasilinearer Operator: Fehlerverhalten für den Smoluchowski-Kern.

Dimension	128		256		512		1024	
$k = 1$	9.335e-02	–	9.346e-02	–	9.351e-02	–	9.354e-02	–
$k = 2$	2.655e-03	0.03	2.665e-03	0.03	2.670e-03	0.03	2.672e-03	0.03
$k = 3$	2.110e-04	0.08	2.121e-04	0.08	2.125e-04	0.08	2.127e-04	0.08
$k = 4$	2.092e-05	0.10	2.109e-05	0.10	2.115e-05	0.10	2.117e-05	0.10
$k = 5$	2.458e-06	0.12	2.484e-06	0.12	2.493e-06	0.12	2.496e-06	0.12
$k = 6$	3.029e-07	0.12	3.130e-07	0.13	3.144e-07	0.13	3.149e-07	0.13
$k = 7$	4.091e-08	0.14	4.167e-08	0.13	4.191e-08	0.13	4.198e-08	0.13
$k = 8$	5.624e-09	0.14	5.754e-09	0.14	5.795e-09	0.14	5.808e-09	0.14
$k = 9$	7.877e-10	0.14	8.152e-10	0.14	8.225e-10	0.14	8.246e-10	0.14
$k = 10$	1.135e-10	0.14	1.182e-10	0.14	1.194e-10	0.14	1.198e-10	0.14
$k = 11$	1.670e-11	0.15	1.740e-11	0.15	1.761e-11	0.15	1.768e-11	0.15
$k = 12$	2.486e-12	0.15	2.600e-12	0.15	2.636e-12	0.15	2.647e-12	0.15
Dimension	2048		4096		8192		16384	
$k = 1$	9.355e-02	–	9.356e-02	–	9.356e-02	–	9.356e-02	–
$k = 2$	2.673e-03	0.03	2.674e-03	0.03	2.674e-03	0.03	2.674e-03	0.03
$k = 3$	2.128e-04	0.08	2.128e-04	0.08	2.129e-04	0.08	2.128e-04	0.08
$k = 4$	2.118e-05	0.10	2.119e-05	0.10	2.119e-05	0.10	2.118e-05	0.10
$k = 5$	2.496e-06	0.12	2.497e-06	0.12	2.498e-06	0.12	2.497e-06	0.12
$k = 6$	3.151e-07	0.13	3.152e-07	0.13	3.068e-07	0.12	3.151e-07	0.13
$k = 7$	4.201e-08	0.13	4.202e-08	0.13	4.203e-08	0.14	4.201e-08	0.13
$k = 8$	5.813e-09	0.14	5.815e-09	0.14	5.811e-09	0.14	5.813e-09	0.14
$k = 9$	8.255e-10	0.14	8.257e-10	0.14	8.258e-10	0.14	8.255e-10	0.14
$k = 10$	1.199e-10	0.14	1.200e-10	0.14	1.200e-10	0.14	1.199e-10	0.14
$k = 11$	1.770e-11	0.15	1.771e-11	0.15	1.771e-11	0.15	1.767e-11	0.15
$k = 12$	2.651e-12	0.15	2.652e-12	0.15	2.652e-12	0.15	2.651e-12	0.15

Tabelle D.8.: Quasilinearer Operator: Approximationsfehler ε_{rel} für den Emulsionskern (Aggregation) mit $c_{\text{co},3} = 2.41 \cdot 10^{-6}$ und $c_{\text{co},4} = 6.32 \cdot 10^{15}$.

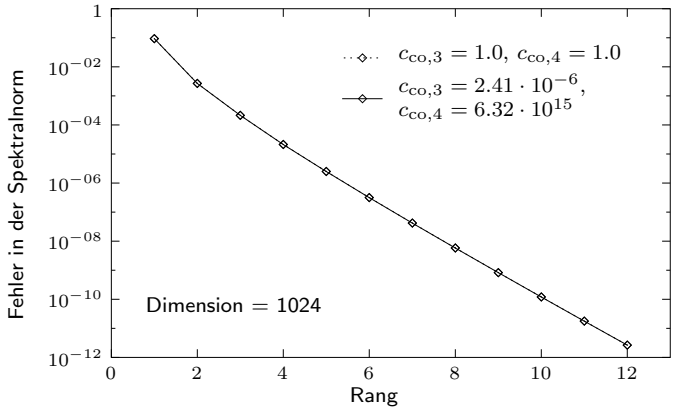


Abbildung D.7.: Quasilinearer Operator: Fehlerverhalten für den Emulsionskern.

D.3. Quadratischer Operator

D.3.1. Zeiten für das Aufstellen

Dimension	128		256		512		1024	
$k = 1$	1.20e−02	10.44%	2.45e−02	5.34%	5.10e−02	2.77%	1.06e−01	1.44%
$k = 2$	1.99e−02	17.30%	4.26e−02	9.28%	8.93e−02	4.85%	1.87e−01	2.54%
$k = 3$	3.54e−02	30.78%	7.64e−02	16.64%	1.60e−01	8.70%	3.34e−01	4.54%
$k = 4$	3.61e−02	31.39%	7.86e−02	17.12%	1.66e−01	9.02%	3.49e−01	4.75%
$k = 5$	6.28e−02	54.61%	1.40e−01	30.50%	2.99e−01	16.25%	6.26e−01	8.52%
$k = 6$	6.32e−02	54.96%	1.42e−01	30.94%	3.04e−01	16.52%	6.39e−01	8.69%
$k = 7$	6.35e−02	55.22%	1.43e−01	31.16%	3.08e−01	16.74%	6.52e−01	8.87%
$k = 8$	6.39e−02	55.56%	1.45e−01	31.59%	3.13e−01	17.01%	6.67e−01	9.08%
$k = 9$	1.01e−01	87.83%	2.51e−01	54.68%	5.58e−01	30.33%	1.19e+00	16.19%
$k = 10$	1.01e−01	87.83%	2.52e−01	54.90%	5.61e−01	30.49%	1.20e+00	16.33%
$k = 11$	1.01e−01	87.83%	2.52e−01	54.90%	5.64e−01	30.65%	1.21e+00	16.46%
$k = 12$	1.02e−01	88.70%	2.53e−01	55.12%	5.68e−01	30.87%	1.22e+00	16.60%
vollbesetzt	1.15e−01	100.0%	4.59e−01	100.0%	1.84e+00	100.0%	7.35e+00	100.0%
Dimension	2048		4096		8192		16384	
$k = 1$	2.20e−01	0.75%	4.54e−01	0.38%	9.41e−01	0.20%	1.95e+00	–
$k = 2$	3.87e−01	1.32%	8.03e−01	0.68%	1.68e+00	0.36%	3.48e+00	–
$k = 3$	6.91e−01	2.35%	1.43e+00	1.20%	2.95e+00	0.62%	6.08e+00	–
$k = 4$	7.31e−01	2.49%	1.53e+00	1.29%	3.19e+00	0.67%	6.64e+00	–
$k = 5$	1.30e+00	4.42%	2.68e+00	2.25%	5.52e+00	1.17%	1.14e+01	–
$k = 6$	1.33e+00	4.52%	2.77e+00	2.33%	5.74e+00	1.21%	1.18e+01	–
$k = 7$	1.36e+00	4.63%	2.84e+00	2.39%	5.92e+00	1.25%	1.23e+01	–
$k = 8$	1.41e+00	4.80%	2.96e+00	2.49%	6.18e+00	1.31%	1.29e+01	–
$k = 9$	2.49e+00	8.47%	5.15e+00	4.33%	1.06e+01	2.24%	2.18e+01	–
$k = 10$	2.51e+00	8.54%	5.22e+00	4.39%	1.08e+01	2.28%	2.23e+01	–
$k = 11$	2.54e+00	8.64%	5.30e+00	4.45%	1.10e+01	2.33%	2.27e+01	–
$k = 12$	2.58e+00	8.78%	5.40e+00	4.54%	1.13e+01	2.39%	2.33e+01	–
vollbesetzt	2.94e+01	100.0%	1.19e+02	100.0%	4.73e+02	100.0%	–	

Tabelle D.9.: Quadratischer Operator: Zeiten für das Aufstellen der Operatoren, Smoluchowski-Kern.

D.3.2. Approximationsfehler

Dimension	128		256		512		1024	
$k = 1$	7.436e−02	–	7.445e−02	–	7.447e−02	–	7.447e−02	–
$k = 2$	1.457e−03	0.02	1.458e−03	0.02	1.458e−03	0.02	1.458e−03	0.02
$k = 3$	3.513e−05	0.02	3.376e−05	0.02	3.375e−05	0.02	3.377e−05	0.02
$k = 4$	1.745e−05	0.50	5.028e−06	0.15	1.437e−06	0.04	8.784e−07	0.03
$k = 5$	1.285e−05	0.74	4.355e−06	0.87	1.247e−06	0.87	3.329e−07	0.38
$k = 6$	1.285e−05	1.00	4.354e−06	1.00	1.247e−06	1.00	3.325e−07	1.00
$k = 7$	1.285e−05	1.00	4.354e−06	1.00	1.247e−06	1.00	3.325e−07	1.00
$k = 8$	1.285e−05	1.00	4.354e−06	1.00	1.247e−06	1.00	3.325e−07	1.00
$k = 9$	5.812e−06	0.45	3.212e−06	0.74	1.089e−06	0.87	3.118e−07	1.00
$k = 10$	5.812e−06	1.00	3.212e−06	1.00	1.089e−06	1.00	3.118e−07	1.00
$k = 11$	5.812e−06	1.00	3.212e−06	1.00	1.089e−06	1.00	3.118e−07	1.00
$k = 12$	5.812e−06	1.00	3.212e−06	1.00	1.089e−06	1.00	3.118e−07	1.00
Dimension	2048		4096		8192		16384	
$k = 1$	7.447e−02	–	7.448e−02	–	7.448e−02	–	–	
$k = 2$	1.458e−03	0.02	1.458e−03	0.02	1.458e−03	0.02	–	
$k = 3$	3.377e−05	0.02	3.377e−05	0.02	3.377e−05	0.02	–	
$k = 4$	8.536e−07	0.02	8.511e−07	0.02	8.507e−07	0.02	–	
$k = 5$	8.658e−08	0.10	2.699e−08	0.03	2.274e−08	0.03	–	
$k = 6$	8.578e−08	0.99	2.179e−08	0.81	5.499e−09	0.24	–	
$k = 7$	8.578e−08	1.00	2.178e−08	1.00	5.487e−09	1.00	–	
$k = 8$	8.578e−08	1.00	2.178e−08	1.00	5.486e−09	1.00	–	
$k = 9$	8.313e−08	0.97	2.144e−08	0.98	5.445e−09	0.99	–	
$k = 10$	8.313e−08	1.00	2.144e−08	1.00	5.445e−09	1.00	–	
$k = 11$	8.313e−08	1.00	2.144e−08	1.00	5.445e−09	1.00	–	
$k = 12$	8.313e−08	1.00	2.144e−08	1.00	5.445e−09	1.00	–	

Tabelle D.10.: Quadratischer Operator: Relativer Approximationsfehler ε_{rel} für den Smoluchowski-Kern mit $c_{\text{smol}} = 1.0$.

Dimension	128		256		512		1024	
$k = 1$	1.743e−02	–	1.854e−02	–	1.920e−02	–	1.960e−02	–
$k = 2$	2.253e−03	0.13	2.474e−03	0.13	2.612e−03	0.14	2.696e−03	0.14
$k = 3$	2.628e−04	0.12	3.049e−04	0.12	3.315e−04	0.13	3.480e−04	0.13
$k = 4$	6.949e−05	0.26	7.678e−05	0.25	8.120e−05	0.24	8.386e−05	0.24
$k = 5$	9.924e−06	0.14	1.161e−05	0.15	1.259e−05	0.16	1.316e−05	0.16
$k = 6$	8.826e−06	0.89	9.995e−06	0.86	1.061e−05	0.84	1.094e−05	0.83
$k = 7$	8.800e−06	1.00	9.955e−06	1.00	1.056e−05	1.00	1.088e−05	0.99
$k = 8$	8.804e−06	1.00	9.960e−06	1.00	1.057e−05	1.00	1.088e−05	1.00
$k = 9$	1.114e−06	0.13	1.573e−06	0.16	1.773e−06	0.17	1.878e−06	0.17
$k = 10$	1.114e−06	1.00	1.573e−06	1.00	1.773e−06	1.00	1.878e−06	1.00
$k = 11$	1.114e−06	1.00	1.573e−06	1.00	1.773e−06	1.00	1.878e−06	1.00
$k = 12$	1.114e−06	1.00	1.573e−06	1.00	1.773e−06	1.00	1.878e−06	1.00
Dimension	2048		4096		8192		16384	
$k = 1$	1.985e−02	–	2.004e−02	–	2.023e−02	–	–	
$k = 2$	2.748e−03	0.14	2.785e−03	0.14	2.817e−03	0.14	–	
$k = 3$	3.582e−04	0.13	3.649e−04	0.13	3.704e−04	0.13	–	
$k = 4$	8.550e−05	0.24	8.663e−05	0.24	8.764e−05	0.24	–	
$k = 5$	1.349e−05	0.16	1.371e−05	0.16	1.390e−05	0.16	–	
$k = 6$	1.112e−05	0.82	1.124e−05	0.82	1.134e−05	0.82	–	
$k = 7$	1.105e−05	0.99	1.117e−05	0.99	1.127e−05	0.99	–	
$k = 8$	1.106e−05	1.00	1.117e−05	1.00	1.128e−05	1.00	–	
$k = 9$	1.933e−06	0.18	1.966e−06	0.18	1.991e−06	0.18	–	
$k = 10$	1.933e−06	1.00	1.966e−06	1.00	1.991e−06	1.00	–	
$k = 11$	1.933e−06	1.00	1.966e−06	1.00	1.991e−06	1.00	–	
$k = 12$	1.933e−06	1.00	1.966e−06	1.00	1.991e−06	1.00	–	

Tabelle D.11.: Quadratischer Operator: Relativer Approximationsfehler ε_{rel} für den Smoluchowski-Kern mit $c_{\text{smol}} = 0.00000001$.

Dimension	128		256		512		1024	
$k = 1$	4.043e-02	–	4.052e-02	–	4.054e-02	–	4.055e-02	–
$k = 2$	1.339e-03	0.03	1.350e-03	0.03	1.353e-03	0.03	1.354e-03	0.03
$k = 3$	1.097e-04	0.08	1.107e-04	0.08	1.111e-04	0.08	1.113e-04	0.08
$k = 4$	2.310e-05	0.21	1.510e-05	0.14	1.273e-05	0.11	1.213e-05	0.11
$k = 5$	1.000e-05	0.43	5.284e-06	0.35	2.850e-06	0.22	1.864e-06	0.15
$k = 6$	9.931e-06	0.99	5.125e-06	0.97	2.543e-06	0.89	1.297e-06	0.70
$k = 7$	9.930e-06	1.00	5.121e-06	1.00	2.536e-06	1.00	1.285e-06	0.99
$k = 8$	9.930e-06	1.00	5.121e-06	1.00	2.536e-06	1.00	1.285e-06	1.00
$k = 9$	3.575e-06	0.36	2.483e-06	0.48	1.280e-06	0.50	6.341e-07	0.49
$k = 10$	3.575e-06	1.00	2.483e-06	1.00	1.280e-06	1.00	6.341e-07	1.00
$k = 11$	3.575e-06	1.00	2.483e-06	1.00	1.280e-06	1.00	6.341e-07	1.00
$k = 12$	3.575e-06	1.00	2.483e-06	1.00	1.280e-06	1.00	6.341e-07	1.00
Dimension	2048		4096		8192		16384	
$k = 1$	4.055e-02	–	4.055e-02	–	4.055e-02	–	–	
$k = 2$	1.355e-03	0.03	1.355e-03	0.03	1.355e-03	0.03	–	
$k = 3$	1.114e-04	0.08	1.114e-04	0.08	1.114e-04	0.08	–	
$k = 4$	1.197e-05	0.11	1.192e-05	0.11	1.191e-05	0.11	–	
$k = 5$	1.559e-06	0.13	1.474e-06	0.12	1.449e-06	0.12	–	
$k = 6$	6.974e-07	0.45	4.060e-07	0.28	2.705e-07	0.19	–	
$k = 7$	6.768e-07	0.97	3.697e-07	0.91	2.082e-07	0.77	–	
$k = 8$	6.765e-07	1.00	3.690e-07	1.00	2.070e-07	0.99	–	
$k = 9$	3.213e-07	0.47	1.692e-07	0.46	9.227e-08	0.45	–	
$k = 10$	3.213e-07	1.00	1.692e-07	1.00	9.227e-08	1.00	–	
$k = 11$	3.213e-07	1.00	1.692e-07	1.00	9.227e-08	1.00	–	
$k = 12$	3.213e-07	1.00	1.692e-07	1.00	9.227e-08	1.00	–	

Tabelle D.12.: Quadratischer Operator: Relativer Approximationsfehler ε_{rel} für den Emulsionskern mit $c_{\text{co},3} = 1.0$ und $c_{\text{co},4} = 1.0$.

Index

Literaturverzeichnis

[1] Bastian, P., K. Birken, K. Johannsen, S. Lang, N. Neuss, H. Rentz-Reichert und C. Wieners: *UG - A flexible software toolbox for solving partial differenital equations.* Computing and Visualizing in Science, 1:27–40, 1 1997.

[2] Bebendorf, M.: *Effiziente numerische Lösung von Randintegralgleichungen unter Verwendung von Niedrigrang-Matrizen.* Doktorarbeit, Universität Saarbrücken, 2000.

[3] Bebendorf, M. und S. Rjasanow: *Adaptive Low-Rang Approximation of Collocation Matrices.* Technischer Bericht 39, Universität Saarbrücken, 2001.

[4] Börm, Steffen: *$\mathcal{H}^2$-matrices - Multilevel methods for the approximation of integral operators.* Technischer Bericht 7, Max-Planck-Institut für Mathematik in den Wissenschaften, 2003.

[5] Börm, Steffen und Lars Grasedyck: *Low-rank approximation of integral operators by interpolation.* Technischer Bericht 72, Max-Planck-Institut für Mathematik in den Wissenschaften, 2002. erscheint bei *Computing.*

[6] Börm, Steffen und Wolfgang Hackbusch: *Hierarchical Quadrature of Singular Integrals.* Computing, 74:75–100, 2005. 19th Dundee Biennial Conference on Numerical Analysis (2001).

[7] Busam, Rolf und Eberhard Freitag: *Funktionentheorie.* Springer-Verlag Berlin, 3. Auflage, 2000.

[8] Cooley, James W. und John W. Tukey: *An algorithm for the machine calculation of complex Fourier series.* Math. Comp., 19:297–301, 1965.

[9] Coulaloglou, C. A. und L. L. Tavlarides: *Description of interaction processes in agitated liquid-liquid dispersions.* Chemical Engineering Science, 32:1289–1297, 1977.

[10] Dahmen, Wolfgang und Reinhold Schneider: *Wavelets on manifolds. I. Construction and domain decomposition.* SIAM J. Math. Anal., 31(1):184–230 (electronic), 1999.

[11] Davis, Philip J.: *Interpolation and approximation.* Blaisdell Publishing Co. Ginn and Co. New York-Toronto-London, 1963.

[12] Eckel, Bruce: *Thinking in C++*. Prentice Hall, 2. Auflage, 2000. Volume One: Introduction to C++.

[13] Emmen, Ad: *Japanese 'Computenik' Earth Simulator shatters US supercomputer hegemony*. http://www.hoise.com/primeur/02/articles/weekly/AE-PR-05-02-59.html, 2002.

[14] Fischer, T., E.-D. Gilles, D. Logashenko, S. Motz und G. Wittum: *Simulation of crystal growth and attrition in a stirred tank*. Technischer Bericht, IWR Heidelberg, 2004. eingereicht.

[15] Fischer, T., M. Kirkilionis, D. Logashenko und G. Wittum: *Fast numerical integration for simulation of disperse systems*. Technischer Bericht, IWR Heidelberg, 2003. eingereicht.

[16] Frigo, Matteo und Steven G. Johnson: *FFTW*. http://www.fftw.org/.

[17] Gahn, C. und A. Mersmann: *Brittle fracture in crystallization processes*. Chemical Engineering Science, 54:1273–1292, 1999. Part A. Attrition and abrasion of brittle solids; Part B. Growth of fragments and scale-up of suspension crystallizers.

[18] Gerstlauer, A.: *Herleitung und Reduktion populationsdyamischer Modelle am Beispiel der Flüssig-Flüssig-Extraktion*. Fortschritt-Berichte VDI. VDI Verlag, 1999.

[19] Gerstlauer, A., E.-D. Gilles, A. Mitrović und S. Motz: *Development, analysis and validation of population models for continouous and batchcrystallizers*. Technischer Bericht, ISR, Universität Stuttgart, 2002. Preprint submitted to Elsevier Preprint.

[20] Ginter, D. M. und S. K. Loyalka: *Apparent size-dependent growth in aggregation crystallizers*. Chemical Engineering Science, 51(14):3685–3695, 1996.

[21] Golub, Gene H. und Charles F. Van Loan: *Matrix computations*. Johns Hopkins University Press, Baltimore, MD, 3. Auflage, 1996.

[22] Grasedyck, Lars: *Theorie und Anwendung Hierachischer Matrizen*. Doktorarbeit, CAU Kiel, 2001.

[23] Grasedyck, Lars und Wolfgang Hackbusch: *Construction and Arithmetics of $\mathcal{H}$-Matrices*. Computing, 70:295–334, 2003.

[24] Greengard, Leslie und Vladimir Rokhlin: *The rapid evaluation of potential fields in three dimensions*. In: *Vortex methods (Los Angeles, CA, 1987)*, Band 1360 der Reihe *Lecture Notes in Math.*, Seiten 121–141. Springer, Berlin, 1988.

[25] Greengard, Leslie und Vladimir Rokhlin: *A new version of the fast multipole method for the Laplace equation in three dimensions*. In: *Acta numerica, 1997*, Band 6 der Reihe *Acta Numer.*, Seiten 229–269. Cambridge Univ. Press, Cambridge, 1997.

[26] Hackbusch, W.: *A sparse matrix arithmetic based on $\mathcal{H}$-matrices: Part I: Introduction to $\mathcal{H}$-matrices.* Computing, 62:89–108, 1999.

[27] Hackbusch, W., B. Khoromskij und S. A. Sauter: *On $\mathcal{H}^2$-Matrices.* In: Bungartz, H., R. Hoppe und C. Zenger (Herausgeber): *Lectures on Applied Mathematics*, Seiten 9–29. Springer-Verlag, Berlin, 2000.

[28] Hackbusch, W. und Z. P. Nowak: *On the fast multiplication in the boundary element method by panel clustering.* Numerische Mathematik, 54:463–491, 1989.

[29] Hackbusch, Wolfgang: *Integralgleichungen.* B. G. Teubner, Stuttgart, 2. Auflage, 1997. Theorie und Numerik. [Theory and numerical treatment], Teubner Studienbücher Mathematik. [Teubner Mathematical Textbooks].

[30] Hackbusch, Wolfgang und Steffen Börm: *$\mathcal{H}^2$-matrix approximation of integral operators by interpolation.* Appl. Numer. Math., 43(1-2):129–143, 2002. 19th Dundee Biennial Conference on Numerical Analysis (2001).

[31] Heuser, Harro: *Lehrbuch der Analysis 2.* B. G. Teubner, 1991.

[32] Lage, C.: *Softwareentwicklung zur Randelementmethode: Analyse und Entwurf effizienter Techniken.* Doktorarbeit, CAU Kiel, 1995.

[33] Larman, Craig: *Applying UML and patterns.* Prentice Hall PTR, 2. Auflage, 2001.

[34] Levich, V. G.: *Physicochemical Hydrodynamics.* Prentice Hall, Englewood Cliffs, New Jersey, 1962.

[35] Löhndorf, Maike: *Effiziente Behandlung von Integraloperatoren mit $\mathcal{H}^2$-Matrizen variabler Ordnung.* Doktorarbeit, Universität Leipzig, 2003.

[36] Motz, S.: *Populationsdynamische Modellierung von Kristallistationsprozessen unter Berücksichtigung der plastischen Deformationsenergie einzelner Kristalle.* Technischer Bericht, ISR, Universität Stuttgart, 1998. Studienarbeit.

[37] Motz, S.: *Modellierung und Simulation von Agglomerationsprozessen in dispersen Fest-Flüssig-Systemen.* Diplomarbeit, Rheinisch-Westfälische Technische Hochschule Aachen, 2000.

[38] Ramkrishna, D.: *The Status of Population Balances.* Reviews in Chemical Engineering, 3:49–95, 1984.

[39] Ramkrishna, D.: *Population Balances.* Academic Press, San Diego, 2000. Theory an Applications to Particulate Systems in Engineering.

[40] Ritter, J.: *Dispergierung und Phasentrennung in gerührten Flüssig/flüssig Systemen.* Doktorarbeit, TU Berlin, 2002.

[41] Rokhlin, Vladimir: *Rapid solution of integral equations of classical potential theory.* J. Comput. Phys., 60(2):187–207, 1985.

[42] Rumbaugh, J., M. Blaha und W. Premerlani: *Object-oriented modeling and design*. Prentice Hall, 1991.

[43] Schneider, Reinhold: *Multiskalen- und Wavelet-Matrixkompression*. Advances in Numerical Mathematics. B. G. Teubner, Stuttgart, 1998. Analysisbasierte Methoden zur effizienten Lösung großer vollbesetzter Gleichungssysteme. [Analysis-based methods for the efficient solution of large nonsparse systems of equations].

[44] Smoluchowski, M. V.: *Versuch einer mathematischen Theorie der Koagulationskinetik kolloider Lösungen*. Zeitschrift für physikalische Chemie, 92:129–168, 1917.

[45] Stoer, Josef: *Numerische Mathematik 1*. Springer, Berlin u.a., 8. Auflage, 1999.

[46] Stroustrup, Bjarne: *Die C++ Programmiersprache*. Addison-Wesley-Longman, 1998.

[47] Wolfram, Stephen: *Das Mathematica Buch*. Addison-Wesley-Longman, Berlin; Paris u.a., 3. Auflage, 1997. Mathematica Version 3.

[48] Ziff, Robert M.: *New solutions to the fragmentation equation*. J. Phys. A, 24(12):2821–2828, 1991.

Selbständigkeitserklärung

Hiermit erkläre ich, die vorliegende Dissertation selbständig und ohne unzulässige fremde Hilfe angefertigt zu haben. Ich habe keine anderen als die angeführten Quellen und Hilfsmittel benutzt und sämtliche Textstellen, die wörtlich oder sinngemäß aus veröffentlichten oder unveröffentlichten Schriften entnommen wurden, und alle Angaben, die auf mündlichen Auskünften beruhen, als solche kenntlich gemacht. Ebenfalls sind alle von anderen Personen bereitgestellten Materialien oder erbrachten Dienstleistungen als solche gekennzeichnet.

Magdeburg, 1. Juli 2005