Skalierbare Löser für ein Meereis-Modell

René Schmieding

Geboren am 26. Juli 1997 in Bad Neuenahr-Ahrweiler

23. September 2020

Bachelorarbeit Mathematik

Betreuer: Prof. Dr. Joscha Gedicke

Zweitgutachter: Dr. Martin Kühn

Institut für Numerische Simulation

Mathematisch-Naturwissenschaftliche Fakultät der

Rheinischen Friedrich-Wilhelms-Universität Bonn

Inhaltsverzeichnis

Zusammenfassung

In dieser Bachelorarbeit beschäftigen wir uns mit dem von Danilov et al. [1] vorgestellten Modell namens *FESIM* zur Simulation von Meereis. Es wird als Bestandteil des Ozeanmodells *FESOM* eingesetzt, und ist diejenige Komponente, welche die Skalierbarkeit der Simulation am stärksten beeinträchtigt. Zuerst wollen wir das Modell in seiner aktuellen Implementierung nachvollziehen. Dabei gehen wir genauer auf dessen Bestandteile Impulsbilanz, Transportgleichung sowie die Verzerrungsrate und den Spannungsterm ein, und betrachten die für die Umsetzung dieser physikalischen Gleichungen verwendeten Methoden. Anschließend untersuchen wir, ob eine Implementierung mit einem modernen Vorkonditionierer aus dem Bereich der Gebietszerlegung, konkret dem *GDSW* Vorkonditionierer aus dem Trilinos Projekt, die Skalierbarkeit von *FESIM* verbessern kann. Wir erreichen dabei bessere Gesamtlaufzeiten im Vergleich zu einem weiteren Modell aus der *FESIM* Implementierung, ohne eine gute Skalierbarkeit zu erzielen. Mit einer optimierten Implementierung könnten wir das Modell jedoch weiter beschleunigen.

Kapitel 1

Einleitung

Das *Finite-Element Sea Ice Model* [1], kurz FESIM, wird zur Simulation von Meereisbewegungen verwendet. Es ist Bestandteil eines Ozeanmodells Namens *Finite-Element Sea Ice Ocean circulation Model* (FESOM), welches die Ergebnisse des Meereismodells verwendet, um beispielsweise Gebiete des Arktischen Ozeans zu simulieren.

Dieses Meereismodell basiert auf einem Modell von Hibler [2], welches auch als *VP-Modell* bezeichnet wird. Nach [1, 3] skaliert es schlecht mit der Gitterauflösung und ist für eine parallele Berechnung nicht gut geeignet. Deshalb gibt es verschiedene Varianten des Modells, zum Beispiel das *EVP-Modell* von Hunke und Dukowicz [3]. Dieses modelliert das Meereis unter bestimmten Vorraussetzungen, bei denen Probleme in der numerischen Behandlung des VP-Modells auftreten würden, als elastisches Material.

In der Implementierung von FESIM [1] werden drei verschiedene Modelle umgesetzt, nämlich das VP- und EVP-Modell, sowie das mEVP-Modell, welches eine Modifikation des EVP-Modells ist.

In dieser Arbeit beschränken wir uns auf das VP-Modell, da es im Gegensatz zu den beiden anderen die Meereisgeschwindigkeiten mit Hilfe eines allgemeinen iterativen Lösers bestimmt, wie zum Beispiel GMRES. An dieser Stelle von FESIM wollen wir anknüpfen, und durch Ankopplung eines modernen Gebietszerlegungsverfahren versuchen, die Leistung und Skalierbarkeit des Modells zu verbessern.

Als Gebietszerlegungsverfahren verwenden wir dabei den GDSW Vorkonditionierer [4] aus dem Trilinos Projekt [5]. Die eigentliche Zerlegung wird dabei durch die Software Metis [6] realisiert.

Kapitel 2

Das Meereismodell

Die Modellierung des Meereises, wie sie von Danilov et al. [1] beschrieben wird, hat ihre Ursprünge in dem VP-Modell von Hibler [2]. Es lässt sich grob in vier Hauptbestandteile unterteilen. Der erste ist die Impulsbilanz

$$m * (\partial_t \boldsymbol{u} + f \times \boldsymbol{u}) = a\tau - aC_d\rho_o(\boldsymbol{u} - \boldsymbol{u}_o)\,|\boldsymbol{u} - \boldsymbol{u}_o| + F - mg\nabla H, \qquad (2.1)$$

welche die Bewegungsgeschwindigkeit des Meereises unter dem Einfluss verschiedener Kräfte beschreibt. Dies ist die zentrale Gleichung, die wir für dieses Modell lösen.

Der zweite Teil besteht aus Transportgleichungen, die den Massetransport des Eises im Verhältnis zu dessen Geschwindigkeit bestimmen. Den letzten Bestandteil bilden das Materialgesetz, oder in diesem Zusammenhang auch Rheologie genannt, in Verbindung mit einem Term für die Stärke der Eisinteraktion.

Bevor wir uns diese Gleichungen näher anschauen, gehen wir auf ihre Bestandteile ein.

2.1 Externe Größen und Konstanten

Diese Größen können als gegebene Parameter angesehen werden. In der Praxis werden sie vor jedem Aufruf von FESIM von dem Ozeanmodell bereitgestellt.

f	Corioliskraft
τ	Schubspannung durch Wind
C_d	Strömungswiderstandskoeffizient
$\boldsymbol{u}_o$	Ozeangeschwindigkeit
ρ_o	Wasserdichte
ρ_s, ρ_{ice}	Schnee- und Eisdichte
g	Gravitationskonstante
H	Höhe der Wasseroberfläche

2.2 Modellgrößen

Dies sind die Werte, welche bei der Simulation regelmäßig aktualisiert werden müssen.

$\boldsymbol{u}$	Geschwindigkeit des Eises
m	Gesamtmasse von Schnee und Eis
h_s, h_{ice}	Schnee- und Eishöhe
a	Eiskonzentration
P	Eisstärke
ζ, η	(Volumen- und Scher-) Viskositäten
$\dot{\epsilon}$	Verformungsrate des Eises
σ	Spannungen im Eis
F	Kraft verursacht durch die Spannungen im Eis

Die Größen a, h_{ice} und h_s werden über die *Transportgleichung* (2.2) bestimmt. Die Geschwindigkeit $\boldsymbol{u}$ unterliegt der *Impulsbilanz* (2.1).

Sowohl bei σ als auch bei $\dot{\epsilon}$ handelt es sich um symmetrische Tensoren zweiter Ordnung. Man kann sie in unserem Fall, da wir ein zweidimensionales Gebiet betrachten, auch als 2×2-Matrizen verstehen, wobei die einzigen beiden nicht-Diagonaleinträge identisch sind. Im weiteren Verlauf müssen wir daher für jeden Tensor nur drei Werte bestimmen.

Alle weiteren Variablen sind von diesen Größen abhängig, wie genau besprechen wir im folgenden Abschnitt.

2.3 Das Modell

Wir wollen nun auf die vier oben genannten Bestandteile des Modells von Hibler [2] eingehen. Es wird in [1] auch als VP-Modell bezeichnet, was für viskos-plastisch steht. Diese Begriffe beschreiben die zentralen Annahmen des Modells, wie das Eis mit sich interagiert oder auf externe Kräfte reagiert.

Grundsätzlich ist das Meereis ein Festkörper und verhält sich als plastisches Material, das heißt Verformungen des Materials bleiben auch ohne weitere Krafteinwirkung erhalten [7, Kapitel 6.2]. Bei sehr kleinen Verformungen über einen längeren Zeitraum hinweg, und in einem nicht zu kleinen Rahmen, wird das Eis als (nichtlinear) viskose, kompressible Flüssigkeit modelliert, es wird also bei zunehmendem Druck zähflüssiger [7, Kapitel 8.2].

Diese Einschränkung bezieht sich darauf, dass das Modell üblicherweise Eisbewegungen auf einem Meer oder Ozean beschreiben soll, und das in einem Zeitraum von Monaten oder Jahren. Dem entsprechend wird nicht betrachtet, wie einzelne Eisbrocken exakt interagieren, sondern wie sich größere Eisflächen über einen ganzen Tag hinweg verhalten, sodass das Eis insgesamt wie eine viskose Flüssigkeit erscheint (siehe auch [8]).

Die Hauptbestandteile des VP-Modells [2, Abschnitte 2.a und 2.b] werden auch in FESIM aufgegriffen. Diese sind

a) **die Impulsbilanz:**

Diese Gleichung beschreibt das Produkt der Masse und Beschleunigung des Eises $m * \partial_t \boldsymbol{u}$ durch eine Summe von verschiedenen Kräften, die auf das Eis einwirken. Man kann sie also grundsätzlich auf das 2. Newtonsche Gesetz $\mathcal{F} = m * a$ zurückführen, wobei $\mathcal{F}$ in diesem Fall als Summe aller auf das Eis einwirkenden Kräfte zu verstehen ist.

Tatsächlich *alle* möglichen Kräfte zu berücksichtigen, wäre jedoch nicht praktikabel, weshalb man sich auf einige dominierende beschränkt. Diese sind nach [9]:

- die Corioliskraft $m * f \times \boldsymbol{u}$,
- die durch Wind und Wasser verursachten Schubspannungen, hier bezeichnet mit τ beziehungsweise dem Term $C_d \rho_o (\boldsymbol{u} - \boldsymbol{u}_o) \, |\boldsymbol{u} - \boldsymbol{u}_o|$ (von Hibler [2] mit τ_a bzw. τ_w bezeichent),
- die durch Spannungen im Eisinneren ausgeübte Kraft F, definiert über den Spannungstensor σ via $F = \nabla \cdot \sigma$,
- und Kräfte durch Höhenunterschiede in der Ozeanoberfläche, $-m * g \nabla H$.

Diese Prozesse gehören dem Teilgebiet der Dynamik an. Die thermodynamischen Vorgänge treten in diesem Modell in der Form von Quell- und Senktermen als rechte Seite der Transportgleichungen auf.

Besonderes Augenmerk ist hier auf F und die Wasserschubspannungen zu legen, da diese nichtlinear von $\boldsymbol{u}$ abhängig sind. Denn im Rahmen der numerischen Behandlung wollen wir $\boldsymbol{u}$ durch ein lineares Gleichungssystem bestimmen, was nur möglich ist, wenn wir diese Kräfte durch lineare Näherungen ersetzen.

Ergänzend zu Hiblers Modell [2] wird neben der Höhe des Eises h_{ice} auch die Höhe h_s einer prognostische Schneeschicht bestimmt, wie sie in [9] verwendet wird. Wir zählen deren Masse zu der des Eises hinzu, das heißt $m = h_{ice} * \rho_{ice} + h_s * \rho_s$.

b) **die Verteilung des Meereises:**

Meereis kann in verschiedenen Formen auftreten, zum Beispiel dünne, frisch gefrorene Schichten an der Wasseroberfläche oder dicke Eisschollen.

Um eine variable *Eishöhe* h_{ice} zu modellieren, verwenden wir nach dem Modell von Hibler [2, Abschnitt 3] eine Unterteilung in dicke und dünne Eisschichten, mit einer vorgegebenen Höhe h_0, an welcher wir den Übergang von dünnen zu dickem Eis festmachen. Den Anteil, welchen das dicke Eis auf einem bestimmten Gebiet ausmacht, bezeichnen wir mit der *Eiskonzentration* a. Wir gehen davon aus, dass der restliche Anteil (also $1-a$) vollständig von dünnem Eis eingenommen wird, also unterscheiden wir nicht explizit zwischen dünnem Eis und offenem Wasser.

In der numerischen Umsetzung unterteilen wir das Simulationsgebiet in Finite Elemente. Wir verwenden dabei Dreieckselemente, auf denen wir jeweils die Eiskonzentration und -höhe bestimmen können.

Die Verteilung von a und h_{ice}, sowie der *Schneehöhe* h_s, werden durch die *Transportgleichungen*

$$\partial_t a + \nabla\cdot(\boldsymbol{u}a) = S_a, \quad \partial_t h_{ice} + \nabla\cdot(\boldsymbol{u}h_{ice}) = S_{ice}, \quad \partial_t h_s + \nabla\cdot(\boldsymbol{u}h_s) = S_s \tag{2.2}$$

beschrieben. S_a und S_{ice} sind dabei Quell- und Senkterme, welche, im Bezug auf das Frieren und Schmelzen von Eis, das Zu- oder Abnehmen der jeweiligen Größe beschreiben sollen. S_s ist der entsprechende Term für den Niederschlag und die Schmelze von Schnee.

c) **die Eisstärke:**

Die *Eisstärke* ist definiert als

$$P_0 = p_0 h_{ice} \exp\left(-C(1-a)\right) \tag{2.3}$$

mit empirischen Konstanten $C = 20$ und $p_0 = 27500\,\mathrm{Nm}^{-2}$, mit Werten aus [1]. Dadurch, dass die Eiskonzentration im Exponent vorkommt, nimmt P_0 rapide ab, wenn die Eisschicht nicht nur aus dickem Eis besteht.

Die Eisstärke P_0 wird ausschließlich zur Berechnung der Viskositäten (2.6) und (2.7) verwendet, an anderer Stelle verwenden Danilov et al. [1] die modifizierte Eisstärke

$$P = \frac{\Delta}{\Delta + \Delta_{min}} P_0 \tag{2.4}$$

mit dem Hilfsterm

$$\Delta = \sqrt{(\dot\epsilon_{11}^2 + \dot\epsilon_{22}^2)(1+\frac{1}{e^2}) + \frac{4}{e^2}\dot\epsilon_{12}^2 + 2\dot\epsilon_{11}\dot\epsilon_{22}(1-\frac{1}{e^2})} \tag{2.5}$$

und dem *Elliptizitätsparameter* $e = 2$. Des weiteren haben wir die *Volumenviskosität*

$$\zeta = \frac{P_0}{2(\Delta + \Delta_{min})}. \tag{2.6}$$

Der Parameter $\Delta_{min} = 2 \times 10^{-9}$ s^{-1} dient in [1] als viskose Regularisierung des plastischen Fließverhaltens, wenn Δ sehr klein wird. Zuletzt ist die *Scherviskosität*

$$\eta = \frac{P_0}{2e^2(\Delta + \Delta_{min})} = \frac{\zeta}{e^2}. \tag{2.7}$$

Für die Formulierung der Viskositäten ζ und η müsste ohne Δ_{min} angenommen werden, dass $\Delta \neq 0$ gilt. Da Δ aber beliebig klein werden und auch den Wert 0 annehmen kann, müssen die Viskositäten künstlich beschränkt werden. Hibler [2] sieht dafür obere Schranken vor, welche abhängig vom Problem so gewählt werden sollen, dass sie möglichst selten erreicht werden.

d) **die Rheologie:**

Die (viskos-plastische) Rheologie, wie sie von Hibler [2] aufgestellt wurde, beschreibt das Verhältnis der Spannungen im Eis zur Verzerrungsrate $\dot{\epsilon}_{ij} = \frac{1}{2}(\frac{\partial \boldsymbol{u}_i}{\partial x_j} + \frac{\partial \boldsymbol{u}_j}{\partial x_i})$, mit Hilfe der Eisstärke P sowie den Volumen- und Scherviskositäten ζ und η.

$$\sigma_{ij} = 2\eta \left(\dot{\epsilon}_{ij} - \frac{1}{2}\delta_{ij}\text{Spur}(\dot{\epsilon}) \right) + \zeta\delta_{ij}\text{Spur}(\dot{\epsilon}) - \delta_{ij}\frac{P}{2} \tag{2.8}$$

Die Gleichungen (2.4)-(2.8) formulieren das nichtlineare *Materialgesetz*, welches die Spannung im Eisinneren mit der Verzerrungsrate verknüpft. Es ermöglicht die Modellierung des Meereises als plastisches Material, wobei die Wahl der Viskositäten einen Übergang zu einer viskosen Flüssigkeit ermöglichen.

Kapitel 3

Numerische Umsetzung

3.1 Die Transportgleichungen

In unserem Modell bleibt das Eis nicht immer am selben Ort stehen, sondern bewegt sich stetig weiter. Das führt dazu, dass einige der Modellgrößen *transportiert* werden, das heißt sie verändern sich abhängig von den Eisbewegungen. Im Folgenden betrachten wir exemplarisch nur die Transportgleichung für die Eiskonzentration a, die Gleichungen für Schnee- und Eishöhe (2.2) werden auf gleiche Weise behandelt:

$$\partial_t a + \nabla \cdot (\boldsymbol{u} \cdot a) = S_a \qquad \text{in } \Omega \tag{3.1}$$

$$\nabla a \cdot \mathbf{n} = 0 \qquad \text{auf } \Gamma \tag{3.2}$$

Wobei $\Gamma \subset \partial\Omega$ den Teil des Randes bezeichnet, auf dem Neumann Randbedingungen gelten. Anschaulich ist es der Rand, welcher an Festland grenzt. Der Vektor $\mathbf{n}$ bezeichnet die auswärtsgerichtete Normale auf dem Rand Γ.

Danilov et al. [1, Abschnitt 3.1] schlagen vor, zuerst die rein homogene Differentialgleichung zu betrachten, das heißt wir setzen für die Berechnung $S_a = 0$. Danach erst sollen die Auswirkungen des Quell- und Senkterms in jedem Knoten einzeln angerechnet werden. Weiterhin nehmen wir an, dass während dieser Berechnung die Geschwindigkeit $\boldsymbol{u}$ des Transportmediums in der Zeit konstant bleibt.

Wir verwenden im folgenden eine Taylor-Galerkin Methode, das heißt wir behandeln die Zeit- und Ortsableitungen getrennt voneinander. Zunächst verwenden wir für die Zeit eine Taylorentwicklung, danach erst greifen wir auf die Finite Elemente Methode für die Ortsableitungen zurück.

3.1.1 Zeitschritte

Wir berechnen alle Gleichungen in diskreten Zeitschritten. Die Variablen, welche sich mit der Zeit verändern, versehen wir mit einem hochgestellten n.

Sei t^n der n-te Zeitpunkt in der Berechnung. Wir definieren die Schrittweite $h := t^{n+1} - t^n$. In der Umsetzung von [1] ist h für die gesamte Berechnung konstant. Entsprechend verwenden wir $a^n(x, y) = a(x, y, t^n)$. Dann ist die Taylorentwicklung in der Zeit

$$a^{n+1} = a^n + h \cdot \partial_t a^n + \frac{h^2}{2} \partial_{tt} a^n + \mathcal{O}(h^3), \tag{3.3}$$

unter der Annahme, dass a ausreichend differenzierbar ist.

Nun ersetzen wir die partiellen Zeitableitungen durch die Transportgleichung (3.1) und deren partielle Ableitung nach t, also $\partial_t a^n = -\nabla \cdot (\boldsymbol{u}^n \cdot a^n)$ und $\partial_{tt} a^n = \nabla \cdot (\boldsymbol{u}^n \nabla \cdot (\boldsymbol{u}^n \cdot a^n))$. Bei der zweiten Ableitung haben wir verwendet, dass $\boldsymbol{u}$ zeitlich konstant ist, also $\partial_t \boldsymbol{u}^n = 0$. Wir erhalten

$$\begin{aligned} a^{n+1} &= a^n - h\nabla \cdot (\boldsymbol{u}^n \cdot a^n) + \frac{h^2}{2} \nabla \cdot (\boldsymbol{u}^n \nabla \cdot (\boldsymbol{u}^n \cdot a^n))) \\ &= a^n - h\nabla \cdot G^n. \end{aligned} \tag{3.4}$$

Der hier definierte Vektor $G^n := (\boldsymbol{u}^n \cdot a^n) - \frac{h}{2} \nabla \cdot (\boldsymbol{u}^n \nabla \cdot (\boldsymbol{u}^n \cdot a^n))$ beschreibt den (diffusen) Materialfluss, also die Verteilungsrate der Eiskonzentration.

Ausgehend von diesem Ergebnis erhalten wir die *schwache Formulierung* der Transportgleichung

$$\int_\Omega (a^{n+1} - a^n + h\nabla \cdot G^n)\varphi \, \mathrm{d}x = 0 \tag{3.5}$$

für geeignete Testfunktionen φ.

3.1.2 Ortsableitungen

Wir wenden nun das Galerkin-Verfahren [10, Kapitel II.4] an. Wir nehmen an, dass Ω ein beschränktes und offenes Gebiet ist, und dass eine *Triangulierung* nach [10, Definition II.5.1] gegeben ist. Ω ist also in endlich viele, zusammenhängende Dreieckselemente $T = (k_{i_1}, k_{i_2}, k_{i_3})$ zerlegt (im folgenden auch nur als *Elemente* bezeichnet). Die Eckpunkte k_i eines Elements T bezeichnen wir auch als *Knoten*. Wir verwenden in den meisten Fällen die Knoten ohne Indizes, und schreiben $k \in T$ oder einfach nur k für einen beliebigen Knoten.

Wir verwenden P_1 Elemente, das heißt wir bestimmen eine auf den Elementen stückweise lineare Näherung α^n der Funktion a^n so, dass ihre Einschränkung α^n_T auf ein Element T durch ihre Werte an den Knoten $k \in T$ vollständig bestimmt ist. Die stückweise linearen Funktionen $\varphi_k \in C^0_c(\Omega)$, die an genau einem Knoten k einen von Null verschiedenen Wert annehmen, bilden eine nodale Basis.

Damit können wir die Approximation $\alpha^n = \sum_k a_k \varphi_k$ als Summe der Basisfunktionen mit Werten $a_k \in \mathbb{R}$ auf den Knoten schreiben. Um die

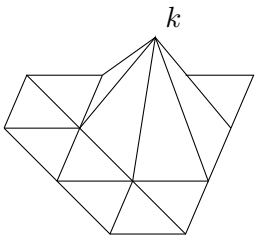

Abbildung 3.1: Basisfunktion φ_k

diskretisierte Lösung der Transportgleichung auf Ω zu finden, genügt es also, die Werte a_k zu bestimmen.

Dafür setzen wir die Basisfunktionen φ_k in die schwache Formulierung (3.5) für φ ein.

$$0 = \int_\Omega (a^{n+1} - a^n + h\nabla \cdot G^n)\varphi_k \, \mathrm{d}x \tag{3.6}$$

$$= \int_\Omega (a^{n+1} - a^n)\varphi_k \, \mathrm{d}x + h \int_\Omega (\nabla \cdot G^n)\varphi_k \, \mathrm{d}x \tag{3.7}$$

$$\tag{3.8}$$

Durch partielle Integration erhalten wir

$$= \int_\Omega (a^{n+1} - a^n)\varphi_k \, \mathrm{d}x - h \left(\int_\Omega G^n \nabla \varphi_k \, \mathrm{d}x - \int_{\partial\Omega} (G^n \cdot \mathbf{n})\varphi_k \, \mathrm{d}s \right). \tag{3.9}$$

Das Randintegral $\int_{\partial\Omega}(G^n \cdot \mathbf{n})\varphi_k \, \mathrm{d}s$ entfällt auf dem gesamten Rand $\partial\Omega$. Auf Γ gilt nach den Randbedingungen $G^n \cdot n = 0$, und für den restlichen Teil des Randes nehmen Danilov et al. [1, Abscnitt 3.1] an, dass dieser weit genug von den eisbedeckten Flächen entfernt liegt, sodass dort $a = 0$ gilt. In beiden Fällen ist das Integral also 0.

Wir definieren nun die *Massematrix*

$$M_{ij} := \int_\Omega \varphi_i \varphi_j \, \mathrm{d}x \tag{3.10}$$

und den diskretisierten linearen Operator

$$A_{ij} := -h \int_\Omega \nabla \varphi_i (\boldsymbol{u}^n \varphi_j - \frac{h}{2} \boldsymbol{u}^n \nabla \cdot (\boldsymbol{u}^n \varphi_j) \, \mathrm{d}x, \tag{3.11}$$

mit welchen wir die Gleichung (3.8) zu dem System

$$M(a^{n+1} - a^n) + Aa^n = 0 \tag{3.12}$$

umformulieren. Dieses lineare System direkt zu lösen wäre jedoch nach [1] zu teuer, und das Schema müsste noch für h_{ice} und h_s wiederholt werden. Stattdessen wird von [1] die folgende Methode vorgeschlagen.

3.1.3 Der Flux Corrected Transport

Da das Invertieren der Massematrix, beziehungsweise das Lösen des linearen Gleichungssystems (kurz *LGS*), sehr teuer werden kann, wollen wir eine alternative Methode verwenden, um a^{n+1} entsprechend (3.12) zu bestimmen. Dazu betrachten wir in diesem Abschnitt die *Flux Corrected Transport* Methode für Finite Elemente von Löhner et al. [11], welches zwei verschiedene Methoden (3.14) und (3.16) zur Lösung von (3.12) kombiniert. Wir bezeichnen sie auch kurz als *FCT* Methode.

Dafür verwenden wir die konzentrierte Massematrix (engl. *lumped mass matrix*)

$$M_{ij}^L := \begin{cases} \sum_t M_{it} & \text{falls } i = j \\ 0 & \text{sonst.} \end{cases} \tag{3.13}$$

M^L ist diagonal, weshalb sich ein LGS mit dieser Matrix sehr leicht lösen lässt. Allerdings erhalten wir keine korrekte Lösung mehr, wenn wir M einfach durch M^L ersetzen. Stattdessen lösen wir das LGS der Form $Mx = b$ iterativ über ein Gesamtschrittverfahren

$$M^L x^{p+1} = (M^L - M)x^p + b \tag{3.14}$$

mit dem Startwert $x^0 = 0$. In [1] werden stets drei Iterationen dieses Verfahrens durchgeführt. Das Endergebnis bezeichnen wir mit $\tilde{x}$.

Um $\tilde{a}^{n+1}$ zu erhalten, führen wir (3.14) mit der rechten Seite $b = Ma^n - Aa^n$ durch. Im Rahmen der FCT Methode [11] wird dieses Ergebnis als *grobe Lösung* bezeichnet.

Für die Berechnung der *feinen Lösung* $\bar{a}^{n+1}$ definieren wir den *Diffusionsterm*

$$\gamma(M - M^L)a^n \tag{3.15}$$

nach [11, (18)]. Dieser Term wird verwendet, um für nicht physikalische Bestandteile in der Transportgleichung (3.12) zu kompensieren. Ein Beispiel dafür ist die zweite Ableitung von a in der Zeit (3.4), welche erst durch die numerische Behandlung im Modell aufgetaucht ist. Aus diesem Grund wird diese Art Term als *numerische Diffusion* [12, Example 5.1] bezeichnet.

Wir setzen nun (3.15) als rechte Seite in (3.12) ein, und ersetzen dort die M durch M^L, wodurch wir

$$M^L(\bar{a}^{n+1} - a^n) + Aa^n = \gamma(M - M^L)a^n \tag{3.16}$$

erhalten. Mit dieser Gleichung bestimmen wir die feine Lösung. Wird der Parameter γ groß genug gewählt (etwa 1), so liefert (3.16) laut Danilov et al. [1, Abschnit 3.1] eine monotone Lösung.

Mit den folgenden Schritten kombinieren wir beide Lösungen. Wir formulieren sie auch als Pseudocode. Zuerst stellen wir die Differenz $\hat{a} := \bar{a}^{n+1} - \tilde{a}^{n+1}$ zwischen den beiden Lösungen auf, welche die Gleichung

$$M^L(\tilde{a}^{n+1} - \bar{a}^{n+1}) = (M^L - M)((\gamma - 1)a^n + \tilde{a}^{n+1}) \tag{3.17}$$

erfüllt, welche wir aus der Differenz der Gleichungen (3.16) und (3.14) erhalten.

$\hat{a}$ wird auch als *antidiffusive flux* bezeichnet [1]. Wir verwenden $\hat{a}$, um ausgehend von der feinen Lösung das Endergebnis a^{n+1} zu erhalten, indem wir die Werte von $\hat{a}$ auf Elementen mitteln und mit einem Faktor $c_T \in [0, 1]$ multiplizieren. Das heißt wir bestimmen das Ergebnis für den nächsten Zeitschritt über

$$a^{n+1} = \tilde{a}^{n+1} + \sum_T c_T \hat{a}_T. \tag{3.18}$$

Den Faktor c_T erhalten wir entsprechend des Abschnitts [11, „The limiting procedure“]. Er wird hier im Rahmen des Pseudocodes behandelt.

Im Folgenden nehmen wir an, dass $\tilde{a}^{n+1}$ und $\bar{a}^{n+1}$ schon bestimmt wurden, und dass wir eine Ordnung für unsere Knoten $1, \ldots, N$ haben. Für einen beliebigen Knoten k bezeichnet (x_k, y_k) dessen Koordinaten.

Um den Pseudocodes etwas übersichtlicher zu machen, definieren wir zuerst einige Kürzel. Für k bezeichne $a_k^n := (a^n(x_k, y_k))$ den Funktionswert von a^n auf diesem Knoten.

Darauf aufbauend definieren wir für ein Element $T = (k_1, k_2, k_3)$ die Vektoren $a_T^n := \begin{pmatrix} a_{k_1}^n & a_{k_2}^n & a_{k_3}^n \end{pmatrix}$ und $\mathcal{M}_j$ als j-te Spalte der Matrix

$$\mathcal{M} := \begin{pmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{pmatrix}.$$

Wir erhalten diese Matrix, wenn wir den Diffusionsterm (3.15) auf einem Element T betrachten [11, (19)], das heißt

$$\gamma(M - M^L)_T = -\frac{\gamma\,|T|}{12}\mathcal{M}_j, \tag{3.19}$$

wobei $|T|$ die Fläche von T bezeichnet.

Algorithm 1 Bestimme a^{n+1} nach Gleichung (3.17)

1: $\hat{a} := \begin{pmatrix} 0 & \dots & 0 \end{pmatrix}$
2: $c := \begin{pmatrix} 0 & \dots & 0 \end{pmatrix}$
3: $P_{\pm} := \begin{pmatrix} 0 & \dots & 0 \end{pmatrix}$
4: $Q_{\pm} := \begin{pmatrix} 0 & \dots & 0 \end{pmatrix}$
5: $R_{\pm} := \begin{pmatrix} 0 & \dots & 0 \end{pmatrix}$
6: **for all** Elemente $T = (k_1, k_2, k_3)$ **do**
7: **for all** Knoten k_j in T **do**
8: $\hat{a}_{k_j} = (M^L_{k_j,k_j})^{-1} \frac{|T|}{12} \mathcal{M}_j \sum_{l=1}^{3} \tilde{a}^{n+1}_{k_l} - a^n_T$
9: **for all** Knoten k **do**
10: $Q_{+,k} = \max\{a_{k'} | k' = k \text{ oder } k' \text{ ist Nachbar von k}\} - \bar{a}^{n+1}_k$
11: $Q_{-,k} = \min\{a_{k'} | k' = k \text{ oder } k' \text{ ist Nachbar von k}\} - \bar{a}^{n+1}_k$
12: **for all** Elemente T **do**
13: **for all** Knoten k in T **do**
14: **if** $\hat{a}_k > 0$ **then**
15: $P_{+,k} = P_{+,k} + \hat{a}_k$
16: **else**
17: $P_{-,k} = P_{-,k} + \hat{a}_k$
18: **for all** Knoten k **do**
19: $R_{+,k} = \begin{cases} \min\{1, \frac{Q_{+,k}}{P_{+,k}}\} & \text{if } P_{+,k} > 0 \\ 0 & \text{else} \end{cases}$
20: $R_{-,k} = \begin{cases} \min\{1, \frac{Q_{-,k}}{P_{-,k}}\} & \text{if } P_{-,k} > 0 \\ 0 & \text{else} \end{cases}$
21: **for all** Elemente T **do**
22: $c_{R_+} := \min\{R_{+,k} \mid \hat{a}_k >= 0, k \in T\}$
23: $c_{R_-} := \min\{R_{-,k} \mid \hat{a}_k < 0, k \in T\}$
24: $c_T = \min\{1, c_{R_+}, c_{R_-}\}$
25: **for all** Knoten k **do**
26: $a^{n+1}_k = \tilde{a}^{n+1}_k$
27: **for all** Elemente T **do**
28: **for all** Knoten k in T **do**
29: $a^{n+1}_k = a^{n+1}_k + c_T \hat{a}_k$

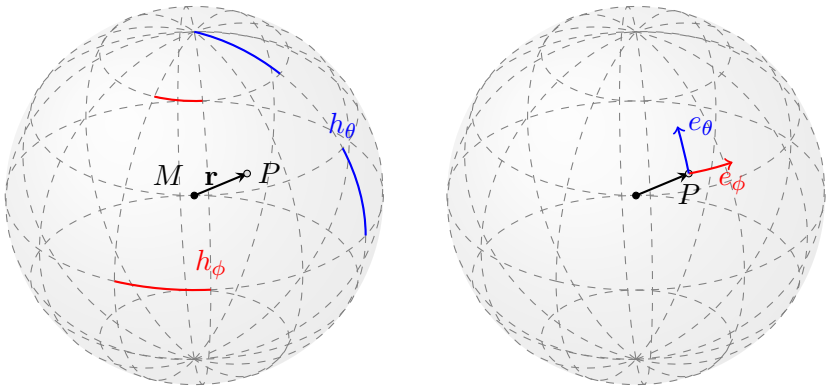

Abbildung 3.2: Darstellung des sphärischen Koordinatensystems mit Maßstabsfaktoren und Basisvektoren.

3.2 Das Materialgesetz

3.2.1 Verzerrungsrate

Die Verzerrungsrate ist in kartesischen Koordinaten definiert als

$$\dot{\epsilon}_{ij} = \frac{1}{2}\left(\frac{\partial \boldsymbol{u}_i}{\partial x_j} + \frac{\partial \boldsymbol{u}_j}{\partial x_i}\right). \tag{3.20}$$

Wir nehmen an, dass wir bereits eine Näherung an die Eisgeschwindigkeit $\boldsymbol{u} = \sum_k \boldsymbol{u}_k \varphi_k$ bestimmt haben. Da diese Näherung auf ein einzelnes Element eingeschränkt linear ist, ist die Verzerrungsrate darauf konstant.

Das Modell soll jedoch auch die Krümmung der Erdoberfläche mit einbeziehen. Entsprechend werden wir die Verzerrungsrate in krummlinigen Koordinaten formulieren. Insbesondere wollen wir später sphärische Koordinaten verwenden, beziehungsweise Kugelkoordinaten mit festem (Erd)radius R.

Wir starten einen kurzen Exkurs in die Tensoralgebra/-analysis.

Den Definitionen von Bärwolf [13, Kapitel 13.2] folgend, erhalten wir die *ζ_i-Koordinatenlinie*, indem wir in einen Punkt P, mit Ortsvektor $\mathbf{r}$ und Koordinaten (ζ_1, ζ_2), die Koordinate ζ_i variieren und die andere festhalten. Mit den Tangentialvektoren

$$\tilde{e}_i = \frac{\partial \mathbf{r}(P)}{\partial \zeta_i} \tag{3.21}$$

an P erhalten wir eine *natürliche Basis* für das krummlinige Koordinatensystem. Im Allgemeinen sind diese Basisvektoren aber nicht normiert, weshalb wir positive *Maßstabsfakoren* h_i so definieren, dass wir durch $e_i = h_i \tilde{e}_i$ normierte Basisvektoren erhalten. Auch ist hier hervorzuheben, dass die Basis

von P abhängig ist, die Basisvektoren müssen also nicht überall gleich sein.

Die Ableitung eines Vektorfeldes in einem solchen System ist dadurch komplizierter als in kartesischen Koordinaten. Sei zum Beispiel $A = \sum_k a_k e_k$ ein Vektorfeld in einem gekrümmten Raum. Die Ableitung in Richtung m ist dann nach Produktregel

$$\frac{\partial A}{\partial m} = \sum_k \left(\frac{\partial a_k}{\partial m} e_k + a_k \frac{\partial e_k}{\partial m} \right). \tag{3.22}$$

In Koordinatensystemen mit ortsunabhängiger Basis, wie dem kartesischen, gilt $\frac{\partial e_k}{\partial m} = 0$. Ansonsten muss diese Ableitung bestimmt werden.

Um an ein sphärisches System zu gelangen, benennen wir die Koordinaten von (ζ_1, ζ_2) in (ϕ, θ) um, da diese nun für die Winkel stehen sollen. Als Maßstabsfaktoren setzen wir $h_\phi = R\cos(\phi)$ und $h_\theta = R$ ein, mit der Basis (e_ϕ, e_θ).

Da wir dieses System hauptsächlich zur akkuraten Darstellung der Verzerrungsrate verwenden wollen, werden wir darauf zurückgreifen, dass sie schon häufiger gebraucht wurde und es entsprechende Formeln bereits gibt. Wir benutzen die Formeln von Batchelor [14](Appendix 2) für sphärische Polarkoordinaten.

Diese können wir jedoch nicht direkt verwenden, da im von Danilov et al.[1] gewählten System zum einen die Koordinate für den Radius (in Batchelors Formeln $\boldsymbol{u}_r$) entfällt, und zum anderen die Winkel ϕ und θ im Vergleich zu [14] im umgekehrten Sinne verwendet werdet werden. Dies führt dazu, dass in den folgenden Formeln cos mit sin vertauscht sind.

$$\dot{\epsilon}_{11} = \frac{1}{R\cos\theta} \frac{\partial \boldsymbol{u}_1}{\partial \phi} + \boldsymbol{u}_2 \frac{\sin\theta}{R\cos\theta} \tag{3.23}$$

$$\dot{\epsilon}_{12} = \frac{cos\theta}{2R} \frac{\partial}{\partial \theta} \left(\frac{\boldsymbol{u}_1}{\cos\theta} \right) + \frac{1}{2R\cos\theta} \frac{\partial \boldsymbol{u}_2}{\partial \phi} \tag{3.24}$$

$$\dot{\epsilon}_{22} = \frac{1}{R} \frac{\partial \boldsymbol{u}_2}{\partial \theta} \tag{3.25}$$

Diese Formeln sind identisch zu denen von Danilov et al.[1](Kapitel 3.2). Um dies zu sehen, ersetzen wir in (3.23) den Sinus durch $\frac{\partial \cos\theta}{\partial \theta}$, und im Term (3.24) wenden wir die Quotientenregel auf $\frac{\partial}{\partial \theta} \left(\frac{\boldsymbol{u}_1}{cos\theta} \right)$ an. Durch Ausklammern und Kürzen erhalten wir schließlich

$$\dot{\epsilon}_{11} = \frac{1}{R\cos\theta}\left(\frac{\partial \boldsymbol{u}_1}{\partial\phi} + \boldsymbol{u}_2\frac{\partial\cos\theta}{\partial\theta}\right) \tag{3.26}$$

$$\dot{\epsilon}_{12} = \frac{1}{2R}\frac{\partial \boldsymbol{u}_1}{\partial\theta} + \frac{1}{2R\cos\theta}\left(-\boldsymbol{u}_1\frac{\partial\cos\theta}{\partial\theta} + \frac{\partial \boldsymbol{u}_2}{\partial\phi}\right) \tag{3.27}$$

$$\dot{\epsilon}_{22} = \frac{1}{R}\frac{\partial \boldsymbol{u}_2}{\partial\theta} \tag{3.28}$$

Zu bemerken ist jetzt, dass sich in diesen Formeln an den Polen Singularitäten befinden. In unserem Modell können wir diese allerdings leicht umgehen, indem wir sie auf Landmassen verschieben, und sie damit nicht im betrachteten Gebiet liegen.

Die Berechnung dieser Terme ist im Rahmen der Simulation aufwendig, da wir sonst mit kartesischen Koordinaten rechnen möchten. Also wechseln Danilov et al.[1] wieder in ein kartesisches System, indem zunächst jedes einzelne Element als flach angenähert wird. Dafür wird $\cos\theta$ im Term $\frac{1}{R\cos\theta}\frac{\partial}{\partial\phi}$ durch eine Abschätzung für das gesamte Element ersetzt.

Um lokal ein kartesisches Referenzsystem zu erhalten, legen wir die x_1- und x_2-Achsen entlang der Richtungen e_ϕ und e_θ. Diese Basisvektoren sind, unter der Annahme, dass das Element flach ist (die Koordinatenlinien bilden also Geraden), auf diesem Element konstant. Unter Berücksichtigung der Maßstabsfakoren schreiben wir dann ∂_{x_1} statt $\frac{1}{R\cos\theta}\frac{\partial}{\partial\phi}$ und ∂_{x_2} anstelle von $\frac{1}{R}\frac{\partial}{\partial\theta}$. Indem wir nun noch eine Abschätzung m_f für $\frac{\sin\theta}{R\cos\theta}$ auf jedem Element einführen, gelangen wir an die Gleichungen

$$\dot{\epsilon}_{11} = \partial_{x_1}\boldsymbol{u}_1 - \boldsymbol{u}_2 m_f \tag{3.29}$$

$$\dot{\epsilon}_{12} = \frac{1}{2}\left(\partial_{x_2}\boldsymbol{u}_1 + \partial_{x_1}\boldsymbol{u}_2 + \boldsymbol{u}_1 m_f\right) \tag{3.30}$$

$$\dot{\epsilon}_{22} = \partial_{x_2}\boldsymbol{u}_2. \tag{3.31}$$

Wir bezeichnen m_f auch als *metrischen Faktor*.

Zuletzt setzen wir die Näherung $\boldsymbol{u} = \sum_k \boldsymbol{u}_k\varphi_k$ ein. Auf einem Element $T = (k_1, k_2, k_3)$ berechnen wir

$$\dot{\epsilon}_{11,T} = \sum_{k\in T}\boldsymbol{u}_{k,1}\partial_{x_1}\varphi_k - \boldsymbol{u}_{k,2}\frac{m_f}{3} \tag{3.32}$$

$$\dot{\epsilon}_{12,T} = \frac{1}{2}\sum_{k\in T}\left(\boldsymbol{u}_{k,1}\partial_{x_2}\varphi_k + \boldsymbol{u}_{k,2}\partial_{x_1}\varphi_k + \boldsymbol{u}_{k,1}\frac{m_f}{3}\right) \tag{3.33}$$

$$\dot{\epsilon}_{22,T} = \sum_{k\in T}\boldsymbol{u}_{k,2}\partial_{x_2}\varphi_k. \tag{3.34}$$

3.2.2 Spannung und Eisstärke

Mit den Ergebnissen aus dem vorherigen Abschnitt können wir den Spannungsterm leicht auf Elementen berechnen. Wenn wir die Eisstärke P für einzelne Elemente bestimmen können, müssen wir nur noch die entsprechenden Werte $\dot{\epsilon}_T$ in Δ und schließlich σ einsetzen. Die Werte, welche wir dadurch erhalten, bezeichnen wir jeweils mit Δ_T und σ_T.

Um nun P_T zu bestimmen, ermitteln Danilov et al. [1] den durchschnittlichen Wert von a und h_{ice} auf jedem Element T über

$$a_T := \sum_{k \in T} \frac{a_k}{3} \tag{3.35}$$

$$h_{ice,T} := \sum_{k \in T} \frac{h_{ice,k}}{3} \tag{3.36}$$

und wir erhalten

$$P_T = \frac{\Delta_T}{\Delta_T + \Delta_{min}} p_0 h_{ice,T} \exp\left(-C(1 - a_T)\right) \tag{3.37}$$

Mit P_T und Δ_T können wir auch die Viskositäten $\zeta_T = \frac{P_T}{2(\Delta_T + \Delta_{min})}$ und $\eta_T = \frac{\zeta_T}{e^2}$ auf Elementen bestimmen. Indem wir diese Terme in (2.8) einsetzen, erhalten wir σ_T:

$$\sigma_{ij,T} = 2\eta_T \left(\dot{\epsilon}_{ij,T} - \frac{1}{2}\delta_{ij}\text{Spur}(\dot{\epsilon}_T) \right) + \zeta_T \delta_{ij} \text{Spur}(\dot{\epsilon}_T) - \delta_{ij}\frac{P_T}{2} \tag{3.38}$$

3.3 Die Impulsbilanz

Kommen wir nun zu dem aufwendigsten Teil in der Simulation, der Impulsgleichung

$$m * (\partial_t \boldsymbol{u} + f \times \boldsymbol{u}) = a\tau - aC_d\rho_o(\boldsymbol{u} - \boldsymbol{u}_o)\,|\boldsymbol{u} - \boldsymbol{u}_o| + F - mg\nabla H. \tag{3.39}$$

In diesem Abschnitt halte ich mich an den [1, Abschnitt 3.3 und 3.4] vorgestellten Lösungsansatz, mit einigen Ergänzungen.

Wir wollen im Folgenden die Komponenten von $\boldsymbol{u} = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}$ separat auf den Knoten bestimmen. Diese Werte erhalten wir durch das Lösen von linearen Gleichungssystemen der Form $Au_i = b_i$ mit $i = 1, 2$, welche wir mit dem Finite Elemente Ansatz aufstellen. Dabei wollen wir die beiden Komponenten von $\boldsymbol{u}$ getrennt voneinander berechnen, aber mit der selben Matrix.

Dieser Schritt hat den Vorteil, dass wir für N Knoten nicht eine $2N \times 2N$ Matrix aufstellen und lösen müssen, da wir in jedem Knoten jeweils einen Freiheitsgrad für u_1 und u_2 haben. Stattdessen wird nur eine $N \times N$ Matrix für beide Komponenten benötigt, was nach [1] die Laufzeit bei Verwendung von iterativen Lösern beschleunigt. Dazu müssen Terme, die in den jeweiligen Komponenten unterschiedlich sind, auf die Rechte Seite des Gleichungssystems verschoben werden. Dies wird im Abschnitt zur Linearisierung weiter besprochen.

Zum Trennen der Komponenten wählen wir skalarwertige Basisfunktionen für jeweils die x- und y-Richtung. Da wir die Komponenten von $\boldsymbol{u}$ *nicht* per se voneinander unabhängig berechnen können, beispielsweise wegen der Corioliskraft f, ersetzen wir die Basisfunktionen φ_k entweder durch $\hat{\varphi}_k := \begin{pmatrix} \varphi_k \\ 0 \end{pmatrix}$ für die Berechnung von u_1 oder $\hat{\varphi}_k := \begin{pmatrix} 0 \\ \varphi_k \end{pmatrix}$ für u_2. Eine ähnliche Notation mit $\hat{}$ verwenden wir auch für die (konzentrierte) Massematrix.

3.3.1 Räumliche Diskretisierung

Zunächst vernachlässigen wir die Zeitschritte oder -ableitungen, und beschränken den Finite Elemente Ansatz auf die Ortsableitungen in der Impulsbilanz. Ableitungen in der Zeit und die Linearisierung einiger Komponenten besprechen wir im folgenden Abschnitt. Ausgehend von (3.39) betrachten wir die schwache Formulierung

$$\int_\Omega m(\partial_t u + f \times \boldsymbol{u})\hat{\varphi}_j \, \mathrm{d}x = \int_\Omega (a\tau - aC_d\rho_o(\boldsymbol{u} - \boldsymbol{u}_o)\,|\boldsymbol{u} - \boldsymbol{u}_o|)\hat{\varphi}_j \, \mathrm{d}x \quad (3.40)$$

$$+ \int_\Omega F\hat{\varphi}_j \, \mathrm{d}x \quad (3.41)$$

$$- \int_\Omega (mg\nabla H)\hat{\varphi}_j \, \mathrm{d}x. \quad (3.42)$$

Wir beginnen damit, die Approximation $\boldsymbol{u} = \sum_k \boldsymbol{u}_k\varphi_k$ der Geschwindigkeit mit Werten $\boldsymbol{u}_k$ auf den Knoten k einzusetzen. Auf der linken Seite der Gleichung (3.40) erhalten wir dadurch einen Term, der die Massematrix $M_{ij} = \int_\Omega \varphi_i\varphi_j \, \mathrm{d}x$ beinhaltet. Wie schon im Kapitel zur Transportgleichung (3.1) möchten wir das Lösen dieser Matrix vermeiden, weshalb wir sie durch die konzentrierte Massematrix M^L ersetzen werden.

Indem wir die Linearität von $f\times$ und ∂_t in $\boldsymbol{u}$ benutzen, können wir folgende Umformungen machen:

$$\int_\Omega m(\partial_t u + f \times \boldsymbol{u})\hat{\varphi}_j \, \mathrm{d}x = \sum_k \int_\Omega m(\partial_t \boldsymbol{u}_k + f \times \boldsymbol{u}_k)\varphi_k \hat{\varphi}_j \, \mathrm{d}x \tag{3.43}$$

Indem wir die Masse m durch einen konstanten Wert m_k für jeden Knoten k ersetzen, und die Definition der Massematrix einsetzen, vereinfacht sich der Ausdruck zu

$$\sum_k m_k(\partial_t \boldsymbol{u}_k + f \times \boldsymbol{u}_k)\hat{M}_{kj} \tag{3.44}$$

Durch das Ersetzen der Massematrix erhalten wir die Näherung

$$\hat{M}^L_{j,j} m_j(\partial_t \boldsymbol{u}_j + f \times \boldsymbol{u}_j). \tag{3.45}$$

Da M^L eine diagonale Matrix ist, bleibt in der Summe über die Knoten nur der Diagonaleintrag, wodurch wir eine leicht berechenbare linke Seite A erhalten. An dieser Stelle sei jedoch angemerkt, dass bei der Behandlung von ∂_t und bei der Linearisierung der Terme auf der rechten Seite (3.40)-(3.42), noch Werte außerhalb der Diagonalen zu A hinzukommen.

Kommen wir nun zu den Termen der rechten Seite. Für das erste Integral (3.40) gehen wir ähnlich vor wie für die linke Seite, indem wir alle Größen auf Knoten diskretisieren, und erhalten als Näherung

$$\begin{aligned}&\int_\Omega (a\tau - aC_d\rho_o(\boldsymbol{u} - \boldsymbol{u}_o)\,|\boldsymbol{u} - \boldsymbol{u}_o|)\hat{\varphi}_j \, \mathrm{d}x \\ &\quad = \hat{M}^L_{j,j}(a_j\tau_j - a_j C_d \rho_o(\boldsymbol{u} - \boldsymbol{u}_o)_j \,|\boldsymbol{u} - \boldsymbol{u}_o|_j)\end{aligned} \tag{3.46}$$

Die Eiskonzentration auf den Knoten a_j haben wir im Abschnitt zur Transportgleichung bestimmt, die übrigen Größen stammen aus dem Ozeanmodell.

Für den nächsten Term setzen zunächst die Definition der Kräfte durch Spannungen im Eis in (3.41) ein, das heißt wir verwenden
$F = \nabla \cdot \sigma = \begin{pmatrix} \partial_1\sigma_{11} + \partial_2\sigma_{12} \\ \partial_1\sigma_{21} + \partial_2\sigma_{22} \end{pmatrix}$, wobei immer noch $\sigma_{12} = \sigma_{21}$ gilt. Das Ergebnis integrieren wir dann partiell, wobei $\mathbf{n}$ die auswärtsgerichtete Normale auf dem Rand $\partial\Omega$ bezeichnet.

$$\int_\Omega F\hat{\varphi}_j \, \mathrm{d}x = \int_{\partial\Omega} \sigma\mathbf{n} \cdot \hat{\varphi}_j \, \mathrm{d}s \tag{3.47}$$

$$- \int_\Omega \sigma \nabla \hat{\varphi}_j \, \mathrm{d}x \tag{3.48}$$

An dieser Stelle müssen wir beachten, dass wir σ nur auf Elementen bestimmen. Entsprechend summieren wir die Beiträge aller Elemente, zu denen der Knoten j gehört, da $\hat{\varphi}_j$ außerhalb dieser Elemente verschwindet. Zudem kommt hier wieder das krummlinige Koordinatensystem zu tragen, über welches wir σ bestimmt haben. Das bedeutet insbesondere, dass der Term

$$\nabla\hat{\varphi}_j = \begin{pmatrix} \frac{1}{R}\frac{\partial\varphi_j}{\partial_\theta} \\ \frac{1}{R\cos(\theta)}\frac{\partial\varphi_j}{\partial_\phi} \end{pmatrix} \tag{3.49}$$

in (3.48) Einfluss auf beide Komponenten hat. Die Formel beziehen wir wieder aus [14]. Mit den Konventionen aus Abschnitt 3.2 für ∂_{x_1} und ∂_{x_2}, sowie dem metrischen Faktor m_f und der Fläche $|T|$ eines Elements schreiben wir den Ergebnisvektor wie folgt:

$$\int_\Omega \sigma\nabla\hat{\varphi}_j \,\mathrm{d}x = \begin{pmatrix} \sum\limits_{\substack{\text{Element } T \\ j\in T}} |T| \left[\sigma_{11,T}\partial_{x_1}\varphi_j + \sigma_{12,T}\partial_{x_2}\varphi_j + \sigma_{12,T}(\frac{m_f}{3})_T\right] \\ \sum\limits_{\substack{\text{Element } T \\ j\in T}} |T| \left[\sigma_{12,T}\partial_{x_1}\varphi_j + \sigma_{22,T}\partial_{x_2}\varphi_j - \sigma_{11,T}(\frac{m_f}{3})_T\right] \end{pmatrix} \tag{3.50}$$

Das Randintegral $\int_{\partial\Omega}\sigma\mathbf{n}\cdot\hat{\varphi}_j \,\mathrm{d}s$ ist bei den hier verwendeten *no-slip* Randbedingungen null, da diese auf dem Rand $\boldsymbol{u} = 0$ erzwingen. Für *free-slip* Randbedingungen müsste dieser Term jedoch bestimmt werden, wobei der dabei erhaltene tangentiale Anteil der Geschwindigkeit wieder auf null gesetzt werden muss.

Es bleibt noch der letzte Summand (3.42). Wir ersetzen wieder die Masse durch Werte auf Knoten m_k. Die Höhe der Wasseroberfläche H nähern wir als linear auf den Elementen an, das heißt wir setzen $H = \sum_k H_k\varphi_k$. Damit erhalten wir

$$\begin{aligned} \int_\Omega (mg\nabla H)\hat{\varphi}_j \,\mathrm{d}x &= \sum_k \int_\Omega (m_j g\nabla(H_k\varphi_k))\hat{\varphi}_j \,\mathrm{d}x && (3.51) \\ &= \sum_k m_j g H_k \int_\Omega \nabla\varphi_k\hat{\varphi}_j \,\mathrm{d}x && (3.52) \\ &:= -m_j g \sum_k H_k \begin{pmatrix} G_{jk}^{x_1} \\ G_{jk}^{x_2} \end{pmatrix} \,\mathrm{d}x. && (3.53) \end{aligned}$$

Die Integrale $G_{jk}^{x_1} := \int_\Omega \varphi_j\partial_{x_1}\varphi_k \,\mathrm{d}x$ und $G_{jk}^{x_2} := \int_\Omega \varphi_j\partial_{x_2}\varphi_k \,\mathrm{d}x$ können wir aufgrund der Wahl der Basisfunktionen leicht als

$$\sum_k H_k G_{jk}^{x_1} = \sum_{\substack{\text{Element } T \\ j \in T}} \frac{|T|}{3} \sum_{k \in T} H_k \partial_{x_1} \varphi_k \tag{3.54}$$

und

$$\sum_k H_k G_{jk}^{x_2} = \sum_{\substack{\text{Element } T \\ j \in T}} \frac{|T|}{3} \sum_{k \in T} H_k \partial_{x_2} \varphi_k \tag{3.55}$$

bestimmen.

3.3.2 Linearisierung

An dieser Stelle müssen wir beachten, dass manche Terme auf der Rechten Seite von (3.40))-(3.42) $\boldsymbol{u}$ enthalten, und zwar auf nichtlineare Weise. Um ein lineares Gleichungssystem $Au_i = b_i$ lösen zu können, müssen wir diese Terme, nämlich die Schubspannungen des Wassers $C_d \rho_o (\boldsymbol{u} - \boldsymbol{u}_o) |\boldsymbol{u} - \boldsymbol{u}_o|$ und die Kraft $F = \nabla \cdot \sigma$, linearisieren. Dies geschieht hauptsächlich innerhalb des Zeitschrittverfahrens, welches im nächsten Abschnitt vorgestellt wird. Um die Linearisierung etwas mehr hervorzuheben, möchte ich die entsprechenden Terme hier isoliert betrachten.

Wir beginnen mit den Schubspannungen. Im Zeitschrittverfahren gehen wir von bekannten Werten $\boldsymbol{u}^n$ aus, und möchten $\boldsymbol{u}^{n+1}$ bestimmen. Dazu berechnen wir in einem Iterationsverfahren Zwischenergebnisse $\boldsymbol{u}^p$, jeweils unter Verwendung des vorherigen Ergebnis $\boldsymbol{u}^{p-1}$, um eine Näherung von $\boldsymbol{u}^{n+1}$ zu erhalten.

Sei also $\boldsymbol{u}^{p-1}$ bestimmt. Um den nächsten Wert $\boldsymbol{u}_p$ zu erhalten, verwenden wir

$$C_d \rho_o (\boldsymbol{u}^p - \boldsymbol{u}_o) \left| \boldsymbol{u}^{p-1} - \boldsymbol{u}_o \right|, \tag{3.56}$$

sodass im Betrag die Näherung aus der vorherigen Iteration verwendet wird. Der Term ist damit linear in $\boldsymbol{u}^p$.

Zum Schluss der Iterationen in p wird ein Korrekturschritt mit der Differenz

$$\begin{aligned} &C_d \rho_o (\boldsymbol{u}^{n+1} - \boldsymbol{u}_o) \left| \boldsymbol{u}^{p-1} - \boldsymbol{u}_o \right| - C_d \rho_o (\boldsymbol{u}^p - \boldsymbol{u}_o) \left| \boldsymbol{u}^{p-1} - \boldsymbol{u}_o \right| \\ &= C_d \rho_o (\boldsymbol{u}^{n+1} - \boldsymbol{u}^p) \left| \boldsymbol{u}^{p-1} - \boldsymbol{u}_o \right| \end{aligned} \tag{3.57}$$

gemacht, mit den bereits bestimmten Werten von $\boldsymbol{u}^p$ und $\boldsymbol{u}^{p-1}$. Durch diesen erhalten wir eine Lösung für $\boldsymbol{u}^{n+1}$, und kompensieren für Fehler aus

der iterativen Näherung.

Für die Linearisierung von F richten wir uns nach [15, Abschnitt 4.4], das heißt wir Teilen σ (2.8) wie folgt auf:

$$\sigma_{ij} = \underbrace{(\eta+\zeta)(\partial_i u_j)}_{\bar{\sigma}_{ij}} + \underbrace{\eta(\partial_j u_i) - \zeta(\partial_i u_j) + (\zeta-\eta)\delta_{ij}\mathrm{Spur}(\dot{\epsilon}) - \delta_{ij}P/2}_{\tilde{\sigma}_{ij}}, \tag{3.58}$$

mit $i, j = 1, 2$. Wie bei den Schubspannungen erfolgt die Linearisierung im Rahmen des Iterationsverfahrens, wobei $\tilde{F} := \nabla \cdot \tilde{\sigma}$ ausschließlich über die bekannten Werte von $\boldsymbol{u}^{p-1}$ bestimmt wird. Die partielle Ableitung von $\boldsymbol{u}^{n+1}$ in $\bar{F} := \nabla \cdot \bar{\sigma}$ bleibt erhalten, die Viskositäten hingegen werden wie für $\tilde{F}$ über $\boldsymbol{u}^{p-1}$ bestimmt.

In dieser Formulierung fehlen noch die für u_1 und u_2 unterschiedlichen Terme, welche wir durch die Berechnung von σ auf krummlinigen Koordinaten erhalten. Sie ist jedoch so gewählt, dass diese Terme nur abhängig von $\tilde{\sigma}$ auftreten [1, Abschnitt 3.4]. Das erlaubt es uns, da wir $\tilde{F}$ unabhängig von $\boldsymbol{u}$ bestimmen, diese Terme auf die rechte Seite b_i zu verschieben.

Betrachten wir genauer, wie diese Terme aussehen. Wir haben

$$\tilde{\sigma}_{11} = (\zeta-\eta)\partial_2 u_2 - P/2 \tag{3.59}$$
$$\tilde{\sigma}_{12} = \eta\partial_2 u_1 - \zeta\partial_1 u_2 \tag{3.60}$$
$$\tilde{\sigma}_{21} = \eta\partial_1 u_2 - \zeta\partial_2 u_1 \tag{3.61}$$
$$\tilde{\sigma}_{22} = (\zeta-\eta)\partial_1 u_1 - P/2. \tag{3.62}$$

Wir können diese Werte für σ in (3.50) einsetzen, und die Komponenten für die jeweilige rechte Seite verwenden.

3.3.3 Zeitschrittverfahren

Zum behandeln der Zeitableitung verwenden eine Methode von Zhang und Hibler [16] und der darauf aufbauenden Arbeit von Hutchings et al. [15], welche in drei Schritten vorgeht. Zunächst wird ein modifiziertes Euler Verfahren in zwei Schritten angewandt, danach wird ein Korrekturschritt gemacht, um eine stabile Lösung für den Coriolisterm und die Wasser-Schubspannungen zu erhalten.

Grundsätzlich wollen wir die Gleichung

$$m^n\left(\frac{\boldsymbol{u}^{n+1}}{h} + f \times \boldsymbol{u}^{n+1}\right) - m^n\frac{\boldsymbol{u}^n}{h} = a^n\tau - a^n C_d\rho_o(\boldsymbol{u}^{n+1} - \boldsymbol{u}_o^n)\,|\boldsymbol{u}^n - \boldsymbol{u}_o^n| + \nabla\cdot\sigma^{n+1} - m^n g\nabla H^n \tag{3.63}$$

nach $\boldsymbol{u}^{n+1}$ lösen. $\nabla \cdot \sigma^{n+1}$ ersetzen wir dabei durch $\bar{F} + \tilde{F}$ und verschieben $\bar{F}$ auf die linke Seite der Gleichung.

Die ersten beiden Schritte, wie sie in [1, Abschnitt 3.4] formuliert werden, sind von der Form

$$\begin{aligned} m^n \frac{\boldsymbol{u}^p}{h} - \sum_{i=1,2} \partial_i (\eta + \zeta)^* \partial_i \boldsymbol{u}^p = m^n (f \times \boldsymbol{u}^* + \frac{\boldsymbol{u}^n}{h}) + a^n \tau \\ - a^n C_d \rho_o (\boldsymbol{u}^p - \boldsymbol{u}_o^n) \left| \boldsymbol{u}^* - \boldsymbol{u}_o^n \right| \\ + \tilde{F}^* - m^n g \nabla H^n, \end{aligned} \tag{3.64}$$

wobei das hochgestellte n die Bezeichnung für den aktuellen Zeitschritt ist, zum Zeitpunkt t^n und mit der Schrittweite $h := t^{n+1} - t^n$. Der Index p wird hier eingeführt, um den zweiten Schritt wiederholt anzuwenden. Die Anzahl der Iterationen des Schemas für den ersten und zweiten Schritt bezeichnen wir mit N_p. Für das in [16] beschriebene Verfahren wäre also $N_p = 2$. Für jede dieser Iterationen muss jeweils ein Gleichungssystem für u_1 und u_2 gelöst werden muss, weshalb N_p nicht zu groß gewählt werden sollte. Mit einem höheren N_p wird jedoch eine bessere Konvergenz erreicht [1, Abschnitt 4].

Eine Alternative zum festen Wert N_p wäre es, das Residuum der Gleichung zu bestimmen, um bei Bedarf mehr oder weniger Iterationen durchzuführen. Dadurch kann möglicher Weise die Berechnung beschleunigt werden, ohne an Genauigkeit zu verlieren.

Im ersten Schritt mit $p = 1$ ist $\boldsymbol{u}^* := \boldsymbol{u}^n$. Die Viskositäten auf der linken Seite von (3.64) müssen in jedem Schritt von $\boldsymbol{u}^*$ abhängig bestimmt werden, das selbe gilt für $\tilde{F}^* := \tilde{F}(\boldsymbol{u}^*)$. Dadurch erhalten wir eine erste Näherung für $u^n + 1$, welche wir mit weiteren Iterationen nach dem selben Schema verfeinern.

Für $p = 2$ starten wir die Iterationen mit den Werten $\boldsymbol{u}^* = \frac{\boldsymbol{u}^{p-1} + u^n}{2}$, für alle weiteren Iterationen gilt $\boldsymbol{u}* := \boldsymbol{u}^{p-1}$.

Der dritte und letzte Schritt hat die Form

$$m^n \frac{\boldsymbol{u}^{n+1} - \boldsymbol{u}^{N_p}}{h} + m^n f \times (\boldsymbol{u}^{n+1} - \boldsymbol{u}^{N_p - 1}) = -a^n C_d \rho_o (\boldsymbol{u}^{n+1} - \boldsymbol{u}^{N_p}) \left| \boldsymbol{u}^{N_p - 1} - \boldsymbol{u}_o^n \right|. \tag{3.65}$$

Die linearen Gleichungssysteme erhalten wir aus diesen Gleichungen, indem wir die einzelnen Terme wie im Abschnitt zur räumlichen Diskretisierung besprochen behandeln.

Kapitel 4

Implementierung

In diesem Abschnitt beschreibe ich einige Bestandteile des Programms, welches ich für diese Arbeit geschrieben habe. Die Grundlage bildet dabei die Implementierung von FESIM durch Danilov et al. [1], an welche ich einen iterativen Löser mit dem GDSW Vorkonditionierer aus dem Trilinos Projekt [5] kopple. Ich bezeichne diese Kombination im folgenden als *VP-GDSW*. Für die Gebietszerlegung verwende ich vor jeder Ausführung des Programms Metis [6] über ein Python Interface.

Der gesamte Programmablauf ist in Abbildung 4.1 zu sehen, mit einigen Verweisen auf die in anderen Kapiteln besprochenen Bestandteile.

In den grau markierten Schritten wird der VP-GDSW jeweils zweimal aufgerufen, für beide Komponenten der Geschwindigkeit $\boldsymbol{u}$ einmal. Insgesamt sind das $2(2 + N_p)$ Aufrufe in jedem Zeitschritt.

Ich möchte nun etwas genauer auf den Aufbau des VP-GDSW eingehen. Grundsätzlich erhält der er ein LGS $Ax = b$, sowie einige Parameter für einen iterativen Löser, wie die maximale Anzahl der Iterationen oder die Abbruchtoleranz. Nach der Berechnung wird der für x bestimmte Wert zurückgegeben.

Für eine genauere Beschreibung betrachten wir einige Objekte aus dem Trilinos Projekt.

Zunächst verwende ich das Paket *Teuchos* für grundlegende Funktionen. Es verwaltet zum Beispiel den MPI-Kommunikator, der von FESIM erstellt wird, und ließt einige Parameter für den VP-GDSW aus einer Datei ein. Außerdem liefert das Paket einige nützliche Klassen wie *RCP* oder *ArrayView* zum erstellen der im weiteren gebrauchten Objekte.

Tpetra stellt einige Objekte aus der linearen Algebra zur Verfügung. Die vom VP-GDSW verwendeten Vektoren und die Systemmatrix im CRS-Format stammen aus diesem Paket. Es liefert ebenfalls eine Klasse *Map*,

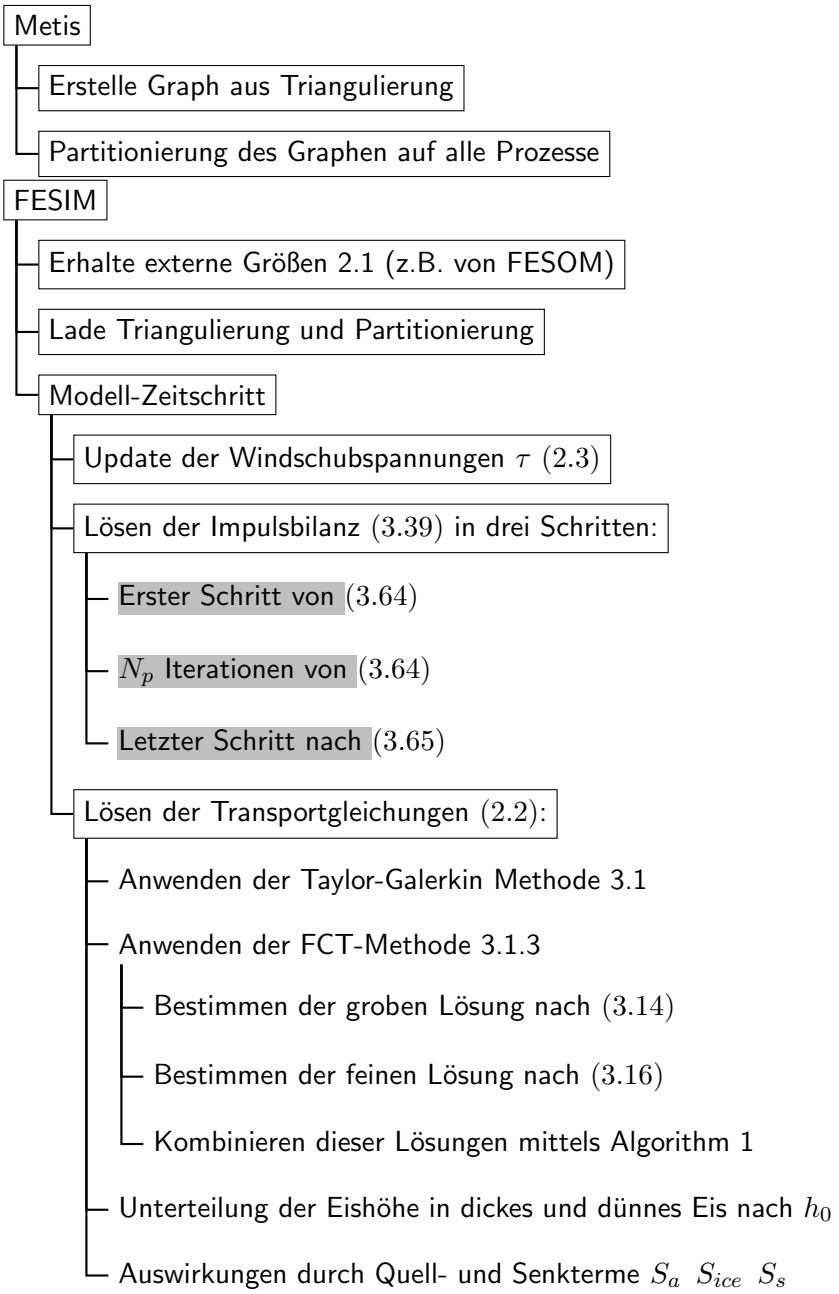

Abbildung 4.1: Der Programmablauf von FESIM als hierarchisches Diagramm. In den hervorgehobenen Schritten wird jeweils der hier beschrieben VP-GDSW verwendet.

welche als Abbildung zwischen den lokalen Knotenindizes k_j auf einem Prozess und den globalen Indizes verstanden werden kann. Dazu verwendet sie die von Metis erstellte Partition.

Die Map, Vektoren und insbesondere die CRS-Matrix werden bei jedem Aufruf des VP-GDSW neu aus Variablen des Fortran-Codes erstellt. Dies kostet zwar Zeit, ist in der aktuellen Implementierung aber notwendig, da die in FESIM benutzten Typen nicht direkt mit Trilinos kompatibel sind.

Für den iterativen Löser des linearen Gleichungssystems verwende ich eine Implementierung von GMRES aus dem Paket *Belos*. Er wird zusammen mit dem GDSW Vorkonditionierer aus dem Paket *FROSch* erstellt.

Beim Programmieren des VP-GDSW sind einige Schwierigkeiten aufgetreten. Zunächst ist FESIM in Fortran geschrieben, Trilinos jedoch in C++. Das bedeutet unter Anderem, dass sich bei der Kommunikation zwischen den beiden Programmen die Indexbasis von 1 auf 0 ändert, was sehr leicht zu gravierenden Fehlern in der Implementierung führen kann, die sich nicht unbedingt direkt bemerkbar machen.

Um solche Fehler zu entdecken, verwende ich Tests unter Einsatz der Bibliothek *Boost.Test*. Damit überprüfe ich einzelne Programmteile mit vorgefertigten Eingaben so, dass ich deren Ergebnisse validieren kann.

Für das compilieren des Programms verwende ich *CMake* zusammen mit *GNU Make*. Diese vereinfachen das Laden der verschiedenen Bibliotheken, wie *Boost.Test*, Trilinos oder OpenMPI unabhängig vom verwendeten System. Sie helfen auch beim Verknüpfen des Fortran- und C++-Codes.

Eine weitere Hürde war, dass sich zum Zeitpunkt der Implementierung einige der für den VP-GDSW verwendeten Pakete von Trilinos noch in aktiver Entwicklung befinden, was teilweise für unerwartetes Verhalten oder fehlende Dokumentation gesorgt hat.

Auch das Kombinieren des GMRES mit dem Vorkonditionierer ist aktuell etwas umständlich, da es über ein *Stratimikos* Interface realisiert wird. Dazu werden zwei zusätzliche Pakete *Xpetra* und *Thyra* benötigt, die ähnlich wie *Tpetra* für lineare Algebra verwendet werden.

Im Kapitel 6 untersuchen wir die Leistung dieses Lösers, und vergleichen ihn mit der Implementierung des mEVP-Modells von [1].

Kapitel 5

Der Vorkonditionierer

In diesem Kapitel betrachten wir den Vorkonditionierer genauer, welchen wir für die Implementierung verwendet haben. Es handelt sich dabei um den von Heinlein et al. [4] vorgestellten GDSW Vorkonditionierer, welcher auf der Zwei-Level Additiven-Schwarz-Methode basiert.

Wir beginnen damit, die ursprüngliche Additive-Schwarz-Methode zu betrachten, wie sie in [17] beschrieben wird. Danach formulieren wir den Vorkonditionierer mit zwei Leveln auf einer Triangulierung, und gehen schließlich auf die Besonderheiten des GDSW Vorkonditionierers ein.

5.1 Die Additive-Schwarz-Methode

Wir betrachten eine lineare, elliptische *partielle Differentialgleichung* (kurz PDG)

$$\begin{aligned} Lu &= f \qquad \text{in } \Omega_i, \\ u &= g \qquad \text{auf } \partial\Omega, \end{aligned} \tag{5.1}$$

mit Dirichlet-Randbedingungen. Die *Alternierende-Schwarz-Methode* wurde von Schwarz formuliert, um das Poisson Problem mit einem iterativen Algorithmus auf komplizierteren Gebieten zu lösen. Das heißt L ist $-\Delta$ und $g = 0$ in (5.1). Dazu betrachtete er Gebiete, die als Vereinigung von einfacheren Geometrien dargestellt werden können [17, Abschnitt 1.1]. Ein Beispiel dafür ist Abbildung 5.1.

Auf einer solchen Zerlegung $\Omega = \Omega_1 \cup \Omega_2$ werden dann abwechselnd Lösungen auf den Teilgebieten Ω_i für $i = 1, 2$ bestimmt, sodass die Werte auf dem Teil des Randes, welcher das andere Gebiet Ω_{3-i} schneidet, mit der dort im vorherigen Schritt bestimmten Lösung übereinstimmen. Mit dieser Methode konnte Schwarz die Existenz einer Lösung des Poisson Problems zeigen [17].

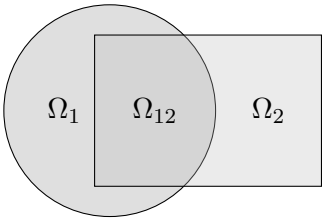

Abbildung 5.1: Das von Schwarz betrachtete Gebiet als Vereinigung von zwei einfachen Geometrien

Anstelle dieses alternierenden Algorithmus [17, (1.2)] betrachten wir die parallele Variante (5.2), da diese in verallgemeinerter Form die Grundlage für die Vorkonditionierer bilden, die wir in den folgenden Abschnitten betrachten werden.

Sei also $\Omega_1 \cup \Omega_2 = \Omega$ eine Zerlegung von Ω in offene Teilgebiete Ω_i, sodass diese einen nichtleeren Schnitt $\Omega_{12} = \Omega_1 \cap \Omega_2$ haben. Wir wollen eine Lösung u bestimmen, indem wir (5.1) eingeschränkt auf die Teilgebiete lösen. Sei $u_i := u_{|\Omega_i}$. Wir betrachten also für $i = 1, 2$ das System

$$\begin{aligned} -\Delta u_i^{n+1} &= f \qquad \text{in } \Omega_i \\ u_i^{n+1} &= g \qquad \text{auf } \partial\Omega_i \cap \partial\Omega \\ u_i^{n+1} &= u_{3-i}^n \qquad \text{auf } \partial\Omega_i \cap \overline{\Omega}_{3-i} \end{aligned} \tag{5.2}$$

mit zusätzlichen Dirichlet-Randbedingungen an den Rändern der Teilgebiete. Dort verwenden wir Funktionswerte aus der vorherigen Iteration.

Um von den Teillösungen wieder auf eine Lösung des gesamten Gebietes zu kommen, definieren wir einen Operator

$$E_i(w_i) : \Omega \mapsto \mathbb{R}, \tag{5.3}$$

sodass $E_i(w_i)$ die *Erweiterung* von einer Funktion $w_i : \Omega_i \mapsto \mathbb{R}$ auf ganz Ω durch den Funktionswert 0 ist. Zusätzlich verwenden wir eine *Zerlegung der Eins* χ_1, χ_2 auf $\Omega = \bigcup_{i=1}^2 \Omega_i$, das heißt

$$\chi_i : \Omega_i \mapsto \mathbb{R}, \quad \chi_i \geq 0 \ \text{ und } \ \chi_i(x) = 0 \ \forall x \in \partial\Omega_i \setminus \partial\Omega. \tag{5.4}$$

Damit schreiben wir die Iterierte

$$u^{n+1} = \sum_{i=1}^{2} E_i(\chi_i u_i) \tag{5.5}$$

Es kann gezeigt werden, dass die Methode (5.2) welche das paar (u_1^n, u_2^n) iteriert, und (5.5), welche u^n iteriert, äquivalent sind [17, Lemma 1.1.1].

5.2 Der Zwei-Level Additive-Schwarz-Vorkonditionierer

In diesem Abschnitt formulieren die Additive-Schwarz-Methode für eine Triangulierung, und führen das zweite Level der Schwarz-Methode ein. Wir halten uns dabei weitestgehend an die Notationen und Definitionen von [18, Kapitel 7.4].

Sei $\mathcal{T}_h$ eine quasi-uniforme Triangulierung des Gebietes Ω mit Gitterweite h, und V_h der endlich dimensionale Finite Elemente-Raum zu $\mathcal{T}_h$. Wir betrachten den diskretisierten Operator

$$A_h u_h = f_h \tag{5.6}$$

zu einer elliptischen, linearen PDG mit dem *symmetrisch positiv definiten* (kurz SPD) Operator $A_h : V_h \to V_h$. Wir wählen zunächst eine Triangulierung $\mathcal{T}_H$ mit Gitterweite $H \geq h$, sodass $\mathcal{T}_h$ eine Verfeinerung von $\mathcal{T}_H$ ist. Entsprechend erhalten wir den Unterraum $V_H \subset V_h$ und den SPD Operator $A_H : V_H \to V_H$ als Einschränkung von A_h auf V_H.

$\mathcal{T}_H$, V_H und A_H beschreiben dabei eine gröbere Version des Problems, weshalb sie eine Art zweites Level beim Lösen des Problems bilden.

Wir betrachten noch weitere Teilräume V_j von V_h, die jeweils zu offenen Teilgebieten $\Omega_1, \ldots, \Omega_N \subset \Omega$ gehören. Wir gehen davon aus, dass die Grenzen dieser Teilgebiete an der Triangulierung $\mathcal{T}_h$ ausgerichtet sind, und dass eine Zerlegung der Eins $\theta_1, \ldots, \theta_N$ mit den folgenden Eigenschaften existiert:

(a) $\theta_j = 0$ auf $\Omega \setminus \Omega_j$

(b) $\sum_{j=1}^{N} \theta_j = 1$ auf $\bar{\Omega}$

(c) Es gilt $\|\nabla \theta_j\|_{L^\infty(\mathbb{R}^2)} \leq C/\delta$ für eine Konstante $\delta \leq H$, sowie einer von h, H, N und δ unabhängigen Konstante C.

(d) Es gibt eine endliche Zahl N_C, sodass jeder Punkt in Ω zu höchstens N_C verschiedenen Teilgebieten gehört.

Durch die Eigenschaften (a) und (b) folgt bereits, dass die sich benachbarte Teilgebiete aus der Zerlegung $\Omega_1, \ldots, \Omega_N$ in einer offenen Menge überschneiden [18, Remark 7.4.7]. Die Konstante δ gibt dabei an, groß diese Überschneidung ist.

Wir definieren nun die Unterräume

$$V_j = \{v \in V_h | v = 0 \text{ auf } \Omega \setminus \Omega_j\} \tag{5.7}$$

der Funktionen, die außerhalb von Ω_j verschwinden. Über die Einschränkung von A_h erhalten wir je einen SPD Operator $A_j : V_j \to V_j'$ für $j = 1, \ldots, N$. V_j' steht dabei für den Dualraum von V_j.

Weiterhin definieren wir $R_H^T : V_H \to V_h$ und $R_j^T : V_j \to V_h$ als die natürlichen Einbettungen der Unterräume in V_h. Die Funktionen R_H und R_j liefern entsprechende Restriktionen auf die Unterräume. Der Vorkonditionierer schreibt sich damit als

$$M^{-1} = R_H^T A_H^{-1} R_H + \sum_{j=1}^{N} R_j^T A_j^{-1} R_j. \tag{5.8}$$

Mit diesem Vorkonditionierer erhalten wir die folgende Abschätzung für die Konditionszahl [18, Theorem 7.4.28]

$$\kappa(M^{-1} A_h) \leq C(1 + \frac{H}{\delta})^2 \tag{5.9}$$

mit einer von h, H, N und δ unabhängigen Konstante C.

5.3 Der GDSW Vorkonditionierer

Nun betrachten wir den für die Implementierung verwendeten Vorkonditionierer, wie er in [4, Abschnitt 2] vorgestellt wird. Dazu benötigen wir zwei verschiedene Zerlegungen des Gebietes Ω. Wir beschränken uns dabei auf ein zweidimensionales Gebiet, da wir ein solches für das Eismodell verwenden. Der dreidimensionale Fall wird ebenfalls in [4] behandelt.

Zum einen betrachten wir die Gebietszerlegung $\Omega_1, \ldots, \Omega_N$ mit offenen Teilgebieten $\Omega_i \subset \Omega$, wobei sich zwei benachbarte Gebiete in einer nichtleeren, offenen Menge überschneiden. Diese Zerlegung entspricht der, die im vorherigen Kapitel für den allgemeinen Zwei-Level Additive-Schwarz-Vorkonditionierer verwendet haben.

Zum anderen teilen wir Ω in disjunkte, offene Mengen $\widetilde{\Omega}_1, \ldots, \widetilde{\Omega}_N \subset \Omega$ auf, sodass die Vereinigung der Abschlüsse $\overline{\widetilde{\Omega}_i}$ von $\widetilde{\Omega}_i$ in Ω wieder das gesamte Gebiet ergibt. Das heißt

$$\Omega = \bigcup_{i=1}^{N} \overline{\widetilde{\Omega}_i}. \tag{5.10}$$

Des weiteren verlangen wir, dass $\widetilde{\Omega}_i \subset \Omega_i$ für alle $i = 1, \ldots, N$ gilt. In der Literatur wird diese Art von Zerlegung auch als *nichtüberlappende Gebietszerlegung* bezeichnet.

Die Mengen $\overline{\widetilde{\Omega}_i} \cap \overline{\widetilde{\Omega}_j}, i \neq j$ bezeichnen wir als *Interface* zwischen den Teilgebieten.

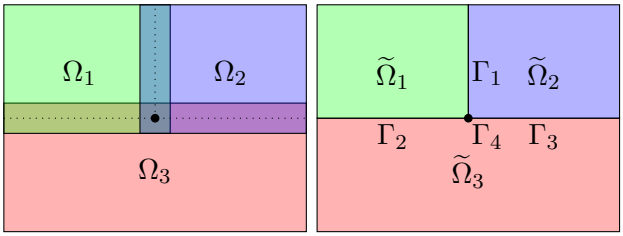

Abbildung 5.2: Eingefärbte Gebietszerlegung in 2D mit überlappenden Teilgebieten Ω_i, und nichtüberlappenden Teilgebieten $\widetilde{\Omega}_i$ einem Knoten Γ_4 und drei Kanten.

Im Bezug auf eine Triangulierung $\mathcal{T}_h$ werden die Gebiete $\widetilde{\Omega}_i$ und Ω_i jeweils so gewählt, dass ihr Rand vollständig auf Kanten der Elemente liegt. Das heißt jedes Element von $\mathcal{T}_h$ liegt in genau einem Gebiet $\widetilde{\Omega}_i$.

In der Implementierung des GDSW Vorkonditionierers [4] erhalten wir Ω_i aus $\widetilde{\Omega}_i$, indem wir $\widetilde{\Omega}_i$ um n Elemente in alle benachbarten Gebiete ausweiten, für eine natürliche Zahl n. Die Konstante δ aus 5.2(c) nimmt dadurch den Wert $\delta = nh$ an.

Bevor wir den Vorkonditionierer aufstellen, teilen wir die Knoten der Triangulierung $\mathcal{T}_h$ wie folgt auf:

Knoten, die zu genau einem Gebiet $\overline{\widetilde{\Omega}_i}$ gehören, fassen wir in der Menge $I := \dot{\bigcup}_{i=1}^{N} I_i$ zusammen. Wir bezeichnen sie als *innere Knoten*. Dazu gehören insbesondere auch Knoten auf dem Rand $\partial\Omega$ des gesamten Gebiets. Das *Interface* Γ sei die Menge aller Knoten, die zu mehr als einem der Gebiete $\overline{\widetilde{\Omega}_i}$ gehören.

Wir können Γ weiter in M zusammenhängende Komponenten Γ_j unterteilen, indem wir alle Knoten zusammenfassen, die zu der selben Teilmenge der Gebiete $\overline{\widetilde{\Omega}_1}, \ldots, \overline{\widetilde{\Omega}_N}$ gehören. Wir bezeichnen Γ_j als Kante, wenn es im Schnitt von genau zwei Gebieten $\overline{\widetilde{\Omega}_i}$ liegt. Sonst nennen wir Γ_j Knoten.

Nun wollen wir den GDSW Vorkonditionierer formulieren. Wir beginnen damit, die Zwei-Level Additive-Schwarz-Methode

$$M^{-1} = \sum_{i=0}^{N} R_i^T A_i^{-1} R_i \tag{5.11}$$

wie im Kapitel zuvor aufzustellen. Das heißt die Funktionen R_i sind die Restriktionen des diskreten Operators A auf die Teilgebiete Ω_i, und $A_i := R_i^T A R_i$ für $i = 1, \ldots, N$. Dabei bezeichnet $A_0 := R_0^T A R_0$ den Grobgitteroperator. Da er nicht auf einem der Teilgebiete Ω_i, sondern auf dem vollständigen Gebiet $\Omega = \bigcup_{i=1}^{N} \Omega_i$ wirkt, schreibt man ihn auch außerhalb

der Summe auf. Wir wollen im folgenden eine spezielle Wahl für die Restriktion R_0 treffen. Deshalb schreiben wir $\Phi = R_0^T$, wodurch wir die äquivalente Formulierung

$$M^{-1} = \Phi A_0^{-1} \Phi^T + \sum_{i=1}^{N} R_i^T A_i^{-1} R_i \tag{5.12}$$

erhalten. Der GDSW Vorkonditionierer ist damit ein Spezialfall der oben vorgestelleten Zwei-Level Additiven-Schwarz-Methode.

Um Φ aufzustellen, definieren wir zunächst die Restriktionen

$$R_I := \begin{pmatrix} R_{I_1} \\ \vdots \\ R_{I_N} \end{pmatrix} \text{ und } R_\Gamma := \begin{pmatrix} R_{\Gamma_1} \\ \vdots \\ R_{\Gamma_M} \end{pmatrix} \tag{5.13}$$

des diskreten linearen Operators jeweils auf die inneren Knoten $I_1, \ldots, I_N$, und die Interfaces $\Gamma_1, \ldots, \Gamma_M$. Wir können nun den Operator A schreiben als

$$A = \begin{pmatrix} A_{II} & A_{I\Gamma} \\ A_{\Gamma I} & A_{\Gamma\Gamma} \end{pmatrix} = \begin{pmatrix} R_I^T A R_I & R_I^T A R_\Gamma \\ R_\Gamma^T A R_I & R_\Gamma^T A R_\Gamma \end{pmatrix}. \tag{5.14}$$

Schließlich definieren wir Φ über die Matrix

$$\Phi = \begin{pmatrix} \Phi_I \\ \Phi_\Gamma \end{pmatrix} := \begin{pmatrix} -A_{II}^{-1} A_{\Gamma I}^T \Phi_\Gamma \\ \Phi_\Gamma \end{pmatrix} \tag{5.15}$$

mit $\Phi_\Gamma = \begin{pmatrix} R_{\Gamma_1}^T \Phi_{\Gamma_1} & \ldots & R_{\Gamma_M}^T \Phi_{\Gamma_M} \end{pmatrix}$. Die Spalten der Matrizen Φ_{Γ_j} sollen dabei so gewählt werden, dass sie eine Basis für die Funktionen im Kern des auf Γ_j eingeschränkten Operator A bilden.

Nun können wir den Grobgitteroperator aufstellen:

$$\begin{aligned} A_0 &= \Phi^T A \Phi \\ &= \begin{pmatrix} -A_{II}^{-1} A_{\Gamma I}^T \Phi_\Gamma \\ \Phi_\Gamma \end{pmatrix}^T \begin{pmatrix} A_{II} & A_{I\Gamma} \\ A_{\Gamma I} & A_{\Gamma\Gamma} \end{pmatrix} \begin{pmatrix} -A_{II}^{-1} A_{\Gamma I}^T \Phi_\Gamma \\ \Phi_\Gamma \end{pmatrix} \\ &= \Phi_\Gamma^T \begin{pmatrix} -A_{II}^{-1} A_{\Gamma I}^T & I_\Gamma \end{pmatrix} \begin{pmatrix} 0 \\ S_{\Gamma\Gamma} \Phi_\Gamma \end{pmatrix} \\ &= \Phi_\Gamma^T S_{\Gamma\Gamma} \Phi_\Gamma \end{aligned} \tag{5.16}$$

Die Matrix I_Γ ist dabei die Identität auf Γ, und $S_{\Gamma\Gamma} = A_{\Gamma\Gamma} - A_{\Gamma I}^T A_{II}^{-1} A_{I\Gamma}$ ist das Schurkomplement von (5.14), welches wir durch das Eliminieren der Freiheitsgrade von inneren Knoten erhalten.

An dieser Stelle sei angemerkt, dass für diesen Prozess weder die Matrixinversen noch das Schurkomplement explizit berechnet werden. Stattdessen wird die Anwendung dieser Operatoren auf einen Vektor bestimmt, zum Beispiel in der Form eines linearen Gleichungssystems.

Wir erhalten schließlich die Abschätzung

$$\kappa(M^{-1}A) \leq C(1 + \frac{H}{\delta})(1 + \log(\frac{H}{h}))^2 \tag{5.17}$$

der Konditionszahl des GDSW Vorkonditionierer [4]. Dabei sind h und H die Gitterweiten der feinen und groben Triangulierung, und δ die Größe der Überschneidung. Verglichen mit (5.9) skaliert die Kondition also nicht mehr quadratisch in H.

Kapitel 6

Laufzeitexperimente

In diesem Abschnitt werten wir die Implementierung mit dem GDSW Vorkonditionierer aus. Wir verwenden dabei das Gitter von Danilov et al. [1] und Verfeinerungen davon.

Das Gitter bezeichnen wir in diesem Abschnitt über die Verfeinerungsstufen $\times n$, wobei $\times 1$ das originale Gitter ist. Die nächste Verfeinerung erhalten wir jeweils durch gleichmäßiges Aufteilen jedes Dreieckselements in vier kleinere Dreieckselemente. Das Gitter $\times 2$ besteht also aus viermal so vielen Elementen wie $\times 1$. Die Anzahl der Knoten steigt entsprechend mit an, jedoch nicht exakt auf das vierfache.

Für die Ausführung des Programms verwenden wir die Einstellungen, wie sie zu Beginn von [1, Abschnitt 4] beschrieben werden. Zum Beispiel werden keine thermodynamischen Prozesse berechnet, da wir nur die Leistung des linearen Lösers, und nicht die Genauigkeit der Simulation testen wollen.

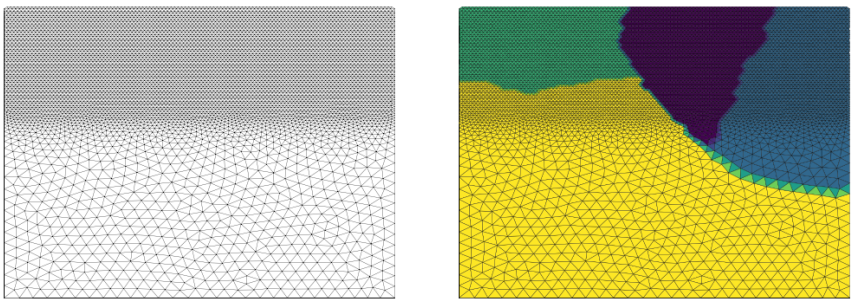

Abbildung 6.1: Der *Box Test case* aus [1]. Die Einfärbung im zweiten Bild entspricht einer von Metis [6] bestimmten Gebietszerlegung für 4 Prozesse.

Laufzeit der Modelle in Sekunden [s]						
Anzahl Prozesse		1	7	14	28	56
Löser	Gitter					
VP-GDSW	×1	2525	653	505	405	**432**
	×2	10565	2054	1311	862	844
mEVP	×1	7081	1881	1370	800	507
	×2	31077	10088	5842	3232	1904

Abbildung 6.2: Übersicht über gemessene Gesamtlaufzeiten der beiden Löser auf zwei Gittern.

Zuerst vergleichen wir die Laufzeiten des mEVP Lösers aus [1] mit dem hier vorgestellten Löser, den wir wie in Kapitel 4 mit VP-GDSW bezeichnen. Die Ergebnisse der Messung zeigen wir in Abbildung 6.2. Die Laufzeiten beziehen sich dabei auf das gesamte Modell, also die Kopplung des mEVP oder VP-GDSW Lösers an FESIM.

#Prozesse	1	4	16	56
Gesamtlaufzeit	2570s	1000s	500s	436s
Zwei-Level Additive-Schwarz-Methode				
Initialisierung	853s (33%)	200s (20%)	150s (30%)	121s (28%)
Aufstellen	1285s (50%)	400s (40%)	169s (34%)	105s (24%)
Anwenden	74s (3%)	93s (9%)	60s (12%)	62s (14%)
Sonstiges	358s (14%)	307s (31%)	121s(24%)	148s (34%)
Grobgitteroperator				
Aufstellen	-	269s (67%)	122s (72%)	80s (76%)
Anwenden	-	12s (13%)	12s (20%)	18s (29%)

Abbildung 6.3: Aufteilung der Laufzeit (in Sekunden) von VP-GDSW, hinsichtlich der Zwei-Level Additive-Schwarz-Methode. Diese Beinhaltet den Belos GMRES und den GDSW Vorkonditionierer. Die Prozentzahlen sind dabei anteilig von der Gesamtlaufzeit zu verstehen. Unter Sonstiges fällt zum Beispiel die Aufbereitung der Daten, die der Löser von FESIM erhält. Im Zweiten Abschnitt der Tabelle zeigen wir, welchen Anteil der Laufzeiten des Aufstellens und Anwendens jeweils durch den Grobgitteroperator verursacht wird.

Anhand der Abbildung 6.2 sehen wir, dass der VP-GDSW in allen Durchläufen schneller ist als der mEVP, meistens sogar zwei- bis dreimal so schnell.

Auffällig ist jedoch, dass der VP-GDSW auf dem Gitter ×1 mit 56 Prozessen länger braucht als mit 28 Prozessen. Das liegt daran, dass der Grobgitterraum bei 56 im Vergleich zum Gitter ×1 zu groß wird, sodass das Lösen des des Grobgitteroperators im Vergleich zum Lösen der Teilgebiete teuer wird.

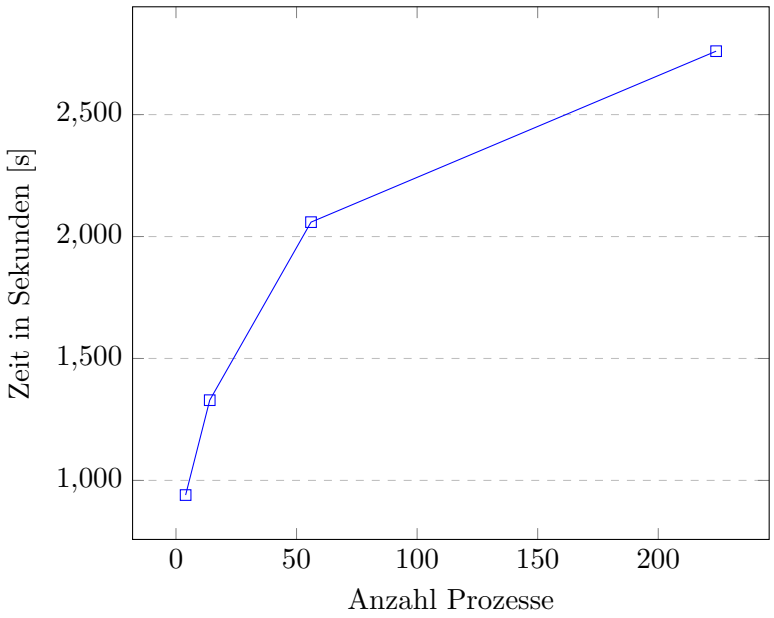

Abbildung 6.4: Schwache Skalierbarkeit des VP-GDSW, gemessen an der Gesamtlaufzeit für 4, 14, 56 und 224 Prozessen, jeweils auf dem Gitter $\times 1$, $\times 2$, $\times 3$ und $\times 4$. Jeder Prozess erhält dabei die gleiche Anzahl an Elementen.

Dieser Effekt lässt sich in anhand von Abbildung 6.3 erkennen, da der Anteil, den der Grobgitteroperator von der Berechnungszeit ausmacht, mit zunehmender Anzahl an Prozessen (und damit auch Teilgebieten) wächst.

Auch sehen wir, dass ein signifikanter Anteil der Gesamtlaufzeit für die Initialisierung und „Sonstiges“ verwendet wird. Die Zeiten in der Spalte „Sonstiges“ umfassen alle Teile des VP-GDSW, welche nicht direkt mit dem berechnen des Gleichungssystems zu tun haben. Dazu gehören das Konstruieren der Map, Matrix und Vektoren. An dieser Stelle kann die Implementierung des Lösers potentiell verbessert werden.

Gehen wir nun auf die Skalierbarkeit des VP-GDSW ein. Wir betrachten zuerst die *schwache Skalierbarkeit*, das heißt wir wählen eine feste Anzahl an Elementen pro Prozess, und betrachten die Laufzeit abhängig von der Anzahl der Prozesse. Das optimale Ergebnis wäre eine konstante Funktion. In Abbildung 6.4 sehen wir, dass wir uns mit zunehmender Anzahl an Prozessen zwar einer konstanten Funktion annähern, insgesamt können wir jedoch keine gute Skalierbarkeit in dieser Hinsicht feststellen.

Für die *starke Skalierbarkeit* erhöhen wir jeweils die Anzahl der Prozesse, ohne das betrachtete Gitter zu verfeinern. In Abbildung 6 können wir erkennen, dass der VP-GDSW im Vergleich zum mEVP wesentlich schlechter auf dem Gitter $\times 1$ skaliert. Auf dem feineren Gitter $\times 2$ sehen wir eine deutlich

bessere Skalierbarkeit auf Seiten des VP-GDSW, sie ist jedoch immer noch schlechter verglichen mit dem mEVP.

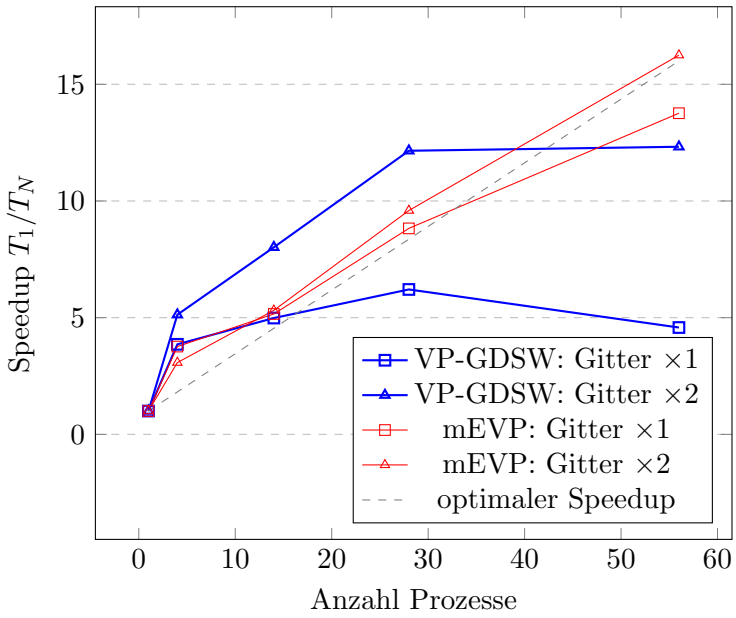

Abbildung 6.5: Speedup T_1/T_N der verschiedenen Löser, wobei N der Anzahl der Prozesse und T_N der Gesamtlaufzeit des Lösers entspricht. Wir betrachten den Speedup des mEVP und VP-GDSW für 1, 4, 14, 28 und 56 Prozesse, jeweils dem Gitter ×1 und ×2. Die absoluten Laufzeiten sind in Abbildung 6.2 eingetragen.

6.1 Fazit

Die Implementierung des Meereismodells FESIM mit dem hier betrachteten Löser VP-GDSW erzielt in unseren Laufzeitexperimenten zwar zum Teil deutlich bessere Laufzeiten als die Implementierung des mEVP-Modells, die Skalierbarkeit fällt jedoch schlechter aus als erwartet. Ein möglicher Grund dafür ist, dass das Grobgitterproblem mit zunehmender Anzahl an Prozessen zu teuer wird. Mit den verschiedenen Einstellungen, die Trilinos für den Grobgitteroperator bereitstellt, können wir ihn für diese Problematik optimieren.

Zudem ist der Grobgitteroperator stark von der Gebietszerlegung abhängig, die wir aktuell einfach von Metis auf der Triangulierung bestimmen lassen. Durch eine geschickte Wahl der Gebietszerlegung könnte das Verhalten des Operators deutlich verbessert werden.

Des Weiteren gibt es Leistungseinbußen in der aktuellen Implementierung des VP-GDSW, dadurch, dass er bei jedem Aufruf neu erstellt werden muss. Wie wir gesehen haben, macht diese Initialisierung einen großen Anteil der Laufzeit aus, sodass eine optimierte Version des Lösers vielleicht eine Beschleunigung um eine Größenordnung erreichen kann. Das heißt, selbst wenn wir keine gute Skalierbarkeit erreichen, erhalten wir mit dem VP-GDSW zumindest einen schnellen Löser für das Meereismodell.

Literaturverzeichnis

[1] S. Danilov und Q. Wang und R. Timmermann und N. Iakovlev und D. Sidorenko und M. Kimmritz und T. Jung und J. Schröter. Finite-Element Sea Ice Model (FESIM), version 2. *Geoscientific Model Development*, 8(6):1747–1761, 2015.

[2] W. D. Hibler. A Dynamic Thermodynamic Sea Ice Model. *Journal of Physical Oceanography*, 9(4):815–846, 1979.

[3] E. C. Hunke und J. K. Dukowicz. An elastic-viscous-plastic model for sea ice dynamics. *Journal of Physical Oceanography*, 27:1849–1867, 1997.

[4] A. Heinlein und A. Klawonn und O. Rheinbach. A Parallel Implementation of a Two-Level Overlapping Schwarz Method with Energy-Minimizing Coarse Space Based on Trilinos. *SIAM Journal on Scientific Computing*, 38(6):C713–C747, 2016.

[5] The Trilinos Project Team. *The Trilinos Project Website*, 2020 (accessed May 22, 2020). https://trilinos.github.io.

[6] George Karypis and Vipin Kumar. Metis, unstructured graph partitioning and sparse matrix ordering system. version 2.0. Technical report, University of Minnesota, Department of Computer Science, Minneapolis, MN 55455, August 1995.

[7] W. Demtröder. *Experimentalphysik 1: Mechanik und Wärme*. Springer Spektrum, Berlin, Heidelberg, 2018.

[8] W. D. Hibler III. A viscous sea ice law as a stochastic average of plasticity. *Journal of Geophysical Research (1896-1977)*, 82(27):3932–3938, 1977.

[9] W. B. Owens und P. Lemke. Sensitivity studies with a sea ice-mixed layer-pycnocline model in the Weddell Sea. *Journal of Geophysical Research*, 95(C6):9527–9538, 1990.

[10] Dietrich Braess. *Finite Elemente: Theorie, schnelle Löser und Anwendungen in der Elastizitätstheorie*. Springer, 2013.

[11] Rainald Löhne und Ken Morgan und Jaime Peraire und Mehdi Vahdati. Finite element flux-corrected transport (FEM–FCT) for the Euler and Navier–Stokes equations. *Wiley Online Library*, Jun 2005.

[12] R. G. Owens und T. N. Phillips. *Computational rheology*. Imperial College Press, London, 2005.

[13] Günter Bärwolff. *Höhere Mathematik für Naturwissenschaftler und Ingenieure*. Springer Berlin Heidelberg, 2017.

[14] G. K. Batchelor. *An Introduction to Fluid Dynamics*. Cambridge Mathematical Library. Cambridge University Press, 2000.

[15] J. K. Hutchings und H. Jasak und S. W. Laxon. A strength implicit correction scheme for the viscous-plastic sea ice model. *Ocean Modelling*, 7(1-2):111–133, 2004.

[16] J. Zhang und W. D. Hibler. On an efficient numerical method for modeling sea ice dynamics. *Journal of Geophysical Research: Oceans*, 102(C4):8691–8702, 1997.

[17] Victorita Dolean und Pierre Jolivet und Frédéric Nataf. *An Introduction to Domain Decomposition Methods*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2015.

[18] Susanne C. Brenner und L. Ridgway Scott. *The Mathematical Theory of Finite Element Methods*. Springer, New York, 2008.