Technische Universität
Bergakademie Freiberg

Diplomarbeit

Parallelisierung geoelektrischer Finite-Elemente-Modellrechnungen auf Linux-Clustern

Jens Oeser

15. Mai 2004

Institut für Geophysik
Gustav-Zeuner-Str. 12
09599 Freiberg

Inhaltsverzeichnis

Verzeichnis der verwendeten Symbole **4**

1 Einleitung **6**

2 Theoretische Grundlagen **9**

2.1 Die Geoelektrik 9

2.1.1 Die MAXWELLschen Gleichungen 9

2.1.2 Die Kontinuitätsgleichung 10

2.1.3 Randbedingungen 11

2.1.4 Methodische Grundlagen 12

2.2 Die Methode der Finiten Elemente 14

2.2.1 Die schwache Formulierung 14

2.2.2 Das GALERKIN-Verfahren 16

2.2.3 Einzelne Probleme der Finite-Elemente-Methode 18

2.3 Parallelisierung der Finite-Elemente-Methode 29

2.3.1 Bewertungskriterien 29

2.3.2 Ansatzpunkte für die Parallelisierung 30

2.3.3 Wahl der optimalen Prozessoranzahl 33

3 Implementierung **34**

3.1 Gittergenerierung 34

3.2 Das Finite-Elemente-Programm *dcfempar* 34

3.3 Modulkonzept 38

4 Untersuchungen **39**

4.1 Genauigkeit des Modellierungsprogrammes *dcfempar* 39

4.1.1 Der homogene Halbraum 40

4.1.2 Der geschichtete Halbraum 44

4.1.3 Die vertikale Platte im homogenen Halbraum 45

4.1.4 Konvergenzkriterium 46

4.2 Vergleich verschiedener Blockvorkonditionierer 47

4.3 Skalierung 49

5 Zusammenfassung **53**

A Verwendete Tetraedergitter **55**

B Inhalt der beiliegenden CD **58**

B.1 Installation der Softwarepakete 58

B.2 Erzeugen der Tetraedergitter 59

B.3 Programmaufruf 60

B.4 Darstellung 60

Literatur **61**

Verzeichnis der verwendeten Symbole

Die folgende Übersicht enthält eine Zusammenstellung der in dieser Arbeit mehrfach unter der gleichen Bedeutung verwendeten Symbole. Die Bedeutung für die hier nicht aufgeführten Symbole können dem entsprechenden Kontext entnommen werden.
Anstelle des Kommas für die Trennung von ganz- und bruchzahligem Anteil wird in dieser Arbeit in Anlehnung an die Rechnerausgabe der Dezimalpunkt benutzt.

Ω	dreidimensionales Gebiet
$\Gamma = \partial\Omega$	Rand des Gebietes Ω
$\overline{\Omega}$	abgeschlossenes Gebiet ($\overline{\Omega} = \partial\Omega \cup \Omega$)
Γ_1	Randstück mit Randbedingungen 1. Art
Γ_2	Randstück mit Randbedingungen 2. Art
Γ_3	Randstück mit Randbedingungen 3. Art
Γ_E	Randstück entlang der Erdoberfläche
$\Gamma_\infty = \Gamma \setminus \Gamma_E$	Randstück nicht entlang der Oberfläche
$\mathcal{T}_h$	Vernetzung (Triangularisierung) des Gebietes Ω
$T^{(r)}$	Element r der Vernetzung
$\hat{T}$	Referenzelement
R_h	Anzahl der Elemente in der Vernetzung $\mathcal{T}_h$
P_h	Anzahl der Knoten in der Vernetzung $\mathcal{T}_h$
$\hat{N}$	Anzahl der Knoten pro Element
h	Diskretisierungsparameter
$\mathbb{R}^n$	n-dimensionaler euklidischer Raum
$C(\overline{\Omega})$	Raum der in $\overline{\Omega}$ stetigen Funktionen
$C(\Omega)$	Raum der in Ω stetigen Funktionen
$C^m(\overline{\Omega})$	Raum der in $\overline{\Omega}$ *m*-mal stetig differenzierbaren Funktionen
$C^m(\Omega)$	Raum der in Ω *m*-mal stetig differenzierbaren Funktionen
$L_2(\Omega)$	Raum der über Ω quadratisch integrierbaren Funktionen
$H^m(\Omega)$	Raum der L_2-Funktionen, deren verallgemeinerten Ableitungen bis zur Ordnung *m* existieren und ebenfalls Element des Raumes $L_2(\Omega)$ sind
V	Raum der Test- und Ansatzfunktionen
V_0	Raum der Funktionen aus V, welche auf dem Randstück Γ_1 gleich Null sind
V_{g1}	Menge der Funktionen aus V, welche die Randbedingungen 1. Art erfüllen
V_h	endlichdimensionaler Teilraum von V

$a(.,.)$	Bilinearform
$l(.)$	Linearform
$(.,.)$	Euklidisches Skalarprodukt im $\mathbb{R}^n$
$\vec{v}$	Vektor
$\vec{\varphi}$	exakte Lösung eines Randwertproblems
$\vec{\varphi_h}$	Finite-Elemente-Näherungslösung
ϕ_i	Finite-Elemente-Ansatzfunktion
$\phi_\alpha^{(r)}$	Formfunktionen über dem Element $T^{(r)}$
p_α	Formfunktionen über dem Referenzelement $\hat{T}^{(r)}$
$\frac{\partial v}{\partial x_i}$	(verallgemeinerte) Ableitung der Funktion v nach x_i
$\frac{\partial v}{\partial \vec{n}}$	Normalenableitung der Funktion v
$\mathrm{grad}\, v$	Gradient von v
$\mathrm{div}\, \vec{v}$	Divergenz von $\vec{v}$
$\mathrm{rot}\, \vec{v}$	Rotation von $\vec{v}$
A_h	Steifigkeitsmatrix mit den Komponenten A_{ij}
$\vec{f_h}$	Lastvektor mit den Komponenten f_i
$A^{(r)}$	Elementsteifigkeitsmatrix für das Element $T^{(r)}$
$\vec{f}^{(r)}$	Elementlastvektor für das Element $T^{(r)}$
$\kappa(A) = \frac{\lambda_{max}(A)}{\lambda_{min}(A)}$	Konditionszahl der Matrix A
$\lambda_{min}(A)$	kleinster Eigenwert der Matrix A
$\lambda_{max}(A)$	größter Eigenwert der Matrix A
P	Anzahl an Prozessoren
$S_{par}(P)$	paralleler Speedup für P Prozessoren
$t(P)$	Laufzeit für P Prozessoren

1 Einleitung

Die Interpretation geoelektrischer Gleichstrommessungen erfordert in zunehmendem Maße den Einsatz von rechnergestützten Auswertemethoden. Auf diesem Gebiet nimmt die Vorwärtsmodellierung einen wichtigen Platz ein. Bei der Planung und Bearbeitung von Messungen, der Lösung der direkten und inversen Aufgabe ist die Vorwärtsrechnung zu einem unverzichtbaren Werkzeug geworden. Ausgehend von einer Verteilung der elektrischen Leitfähigkeit wird bei der Lösung der Vorwärtsaufgabe das elektrische Potential einer oder mehrerer Stromquellen berechnet.

Die bekanntesten Ansätze zur Lösung des Vorwärtsproblems basieren auf Finite-Differenzen- oder Finite-Elemente-Algorithmen. Die Entwicklung der Finite-Differenzen-Methode für die Geophysik begann in den 1970er Jahren (Dey und Morrison, 1979). Über die Jahre wurde das Verfahren ständig u. a. durch den Einsatz von effektiveren Gleichungslösern (Spitzer, 1995) und Techniken zur Verbesserung der Genauigkeit (Lowry, Allen und Shive, 1989) weiterentwickelt. In Arbeiten von Coggon (1971) und Pridmore et al. (1981) wird die Finite-Elemente-Methode für die Gleichstromgeoelektrik eingeführt. Weitere Veröffentlichungen zu diesem Verfahren wurden u. a. von Sasaki (1994) sowie Bing und Greenhalgh (2001) vorgelegt. Vor- und Nachteile beider Methoden werden in einem Artikel von Li und Spitzer (2002) gegenübergestellt.

Ein Vorteil der Finite-Elemente-Methode ist die enorme Flexibilität. Bei einer hohen Genauigkeit der Modellrechnungen können relativ einfach komplizierte Topographien und Leitfähigkeitsstrukturen nachgebildet werden.

In Abbildung 1.1 ist beispielhaft für ein Modell des Vulkans Merapi (Insel Java) ein unstrukturiertes Tetraedergitter[1] dargestellt (Rücker, 2003). Anhand dieses Modells, das eine Kantenlänge von $150\,\mathrm{km}$ besitzt, lässt sich erahnen wie aufwändig Modellrechnungen werden können. Wenn zusätzlich Leitfähigkeitsstrukturen in dieses Modell eingebaut werden, kann das erstellte Gitter bis zu 10^6 Knoten enthalten, für die bei der Finite-Elemente-Diskretisierung Potentialwerte zu bestimmen sind.

Die damit verbundenen Anforderungen an die Rechentechnik werden enorm. Neben effizienten Algorithmen ist die Verwendung von Parallelrechnern eine Möglichkeit, um dem entgegen zu treten. Die Kosten für Multiprozessormaschinen übersteigen aber in der Praxis häufig den Rahmen der finanziellen Möglichkeiten. Eine Verringerung der anfallenden Kosten verspricht der Einsatz normaler PC-Technik. In jüngster Vergangenheit werden immer mehr Parallelrechner durch Zusammenfassen von PCs zu Clustern realisiert. In der Studienarbeit von Oeser (2002) konnte nachgewiesen werden, dass dies ein erfolgversprechender Ansatz für die Geophysik ist.

[1] Zur Verfügung gestellt von Carsten Rücker, Universität Leipzig, Institut für Geophysik und Geologie, Talstraße 35, 04103 Leipzig, E-Mail: cruecker@uni-leipzig.de.

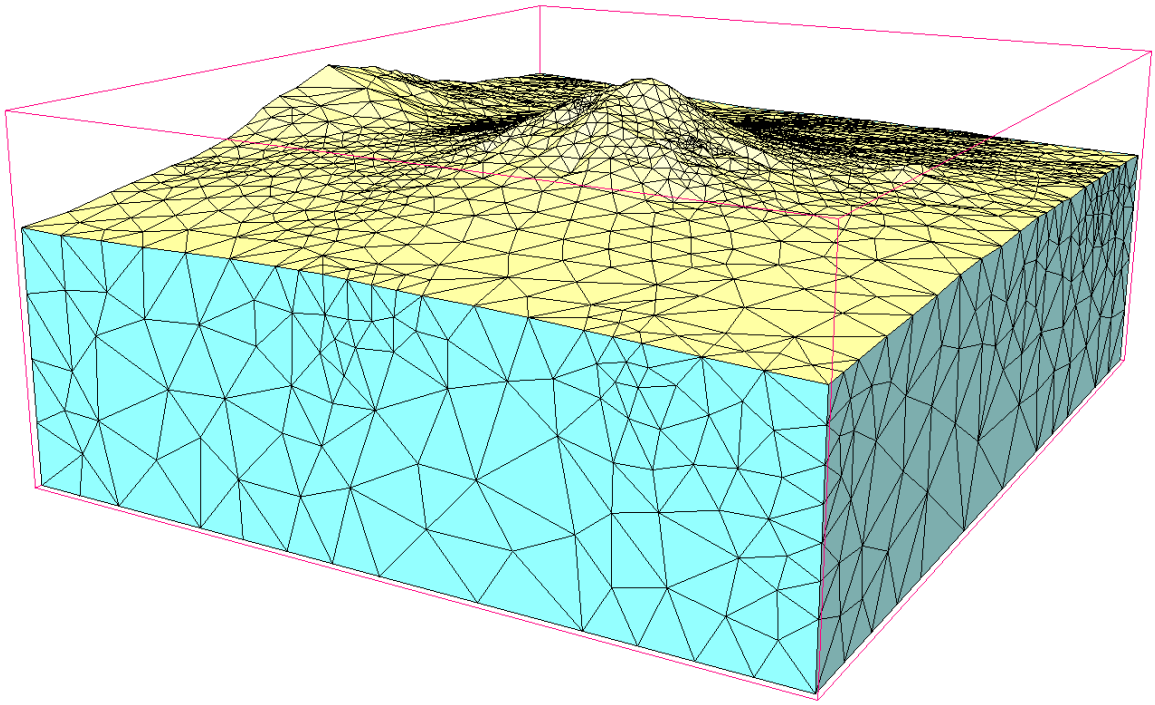


Abbildung 1.1: Unstrukturiertes Tetraedergitter für ein Modell des Vulkans Merapi

Bei großen Parallelrechnern wird speziell für den parallelen Rechenbetrieb entwickelte Hardware eingesetzt. Diese ermöglicht teilweise eine einfache programmiertechnische Umsetzung. Der Einsatz sogenannter PC- oder Linux-Cluster erfordert hingegen einen Mehraufwand bei der Implementierung der Algorithmen. Aus diesem Grund haben sich heute sogenannte nachrichtenorientierte Programmiermodelle durchgesetzt (Huber, 1997). Die Parallelisierung einer Finite-Elemente-Modellrechnung ist dennoch mit einem erheblichen Zeitaufwand verbunden.
Ein Ansatz, um dem entgegen zu wirken, ist die Verwendung von frei zugänglicher Software. Das Ziel dieser Arbeit ist es, ein solches Programm unter Nutzung vorhandener Ressourcen und freier Software zu erstellen. Die Entwicklung freier Software hat in den letzten Jahren einen bedeutenden Aufschwung erhalten, nicht zuletzt durch die Beteiligung namhafter Firmen. Die Liste entsprechender parallelisierter Programmpakete ist dennoch überschaubar. Einen hohen Entwicklungsstand weist die libMesh-Bibliothek (Kirk et al., 2004) auf.
Über diese Finite-Elemente-Bibliothek ist es möglich, die aufwändige Entwicklungsarbeit für ein parallelisiertes Vorwärtsmodellierungsprogramm auf Basis der Finite-Elemente-Methode zu reduzieren. Im Anschluss an die Implementierung (Kapitel 3) ist es notwendig die Genauigkeit des erstellten Programmes anhand analytischer Vergleichslösungen für einfache Modelle zu untersuchen. Weiterhin soll das Laufzeitverhalten mit zunehmender Anzahl an eingesetzten Prozessoren geprüft werden (Kapitel 4). Für die weitere Verwendung des Programmes ist darüber hinaus eine Dokumentation des Quelltextes erforderlich. Im folgenden Abschnitt sol-

len die physikalischen Grundlagen, die Theorie zur Finite-Elemente-Methode und Ansätze für Parallelisierung erläutert werden.

2 Theoretische Grundlagen

2.1 Die Geoelektrik

Die Grundlage aller Überlegungen zu elektromagnetischen Problemstellungen bilden die von Maxwell zusammengetragenen Gleichungen.

2.1.1 Die MAXWELLschen Gleichungen

Die MAXWELLschen Gleichungen in ihrer differentiellen Form lauten

$$\operatorname{rot} \vec{H} = \vec{j} + \dot{\vec{D}} \tag{2.1}$$

$$\operatorname{rot} \vec{E} = -\dot{\vec{B}} \tag{2.2}$$

$$\operatorname{div} \vec{B} = 0 \tag{2.3}$$

$$\operatorname{div} \vec{D} = \rho\,. \tag{2.4}$$

Die Beziehung zwischen der magnetischen Feldstärke $\vec{H}$, der elektrischen Stromdichte $\vec{j}$ und der elektrischen Flussdichte $\vec{D}$ beschreibt die erste MAXWELLsche Gleichung, auch AMPERE-sches Durchflutungsgesetz genannt. Die Wirbel des magnetischen Feldes werden durch die elektrische Stromdichte und die zeitliche Veränderung der elektrischen Flussdichte hervorgerufen. In der zweiten Gleichung, auch FARADAYsches Induktionsgesetz genannt, sind die elektrische Feldstärke $\vec{E}$ und die magnetische Flussdichte $\vec{B}$ miteinander verknüpft. Die Wirbel des elektrischen Feldes entstehen durch die zeitliche Veränderung der magnetischen Flussdichte. Die dritte Gleichung besagt, dass die magnetische Flussdichte keine Quellen besitzt. Dass die elektrische Ladungsdichte ρ die Quelle des elektrischen Feldes ist, wird in der vierten MAXWELLschen Gleichung verdeutlicht.
Die vier Feldgleichungen sind über Materialgleichungen miteinander verknüpft. Diese sind

$$\vec{B} = \mu\vec{H} \tag{2.5}$$

$$\vec{D} = \varepsilon\vec{E} \tag{2.6}$$

$$\vec{j} = \sigma\vec{E} \tag{2.7}$$

mit den Materialgrößen

$\mu = \mu_0\mu_r$	magnetische Permeabilität	$\varepsilon = \varepsilon_0\varepsilon_r$	dielektrische Permittivität
μ_0	magnetische Feldkonstante	ε_0	elektrische Feldkonstante
μ_r	relative Permeabilitätszahl	ε_r	relative Permittivität.

Die ersten beiden Materialgleichungen verknüpfen die Feldstärke (magnetische oder elektrische) mit der entsprechenden Flussdichte über einen Materialparameter. Die dritte Materialgleichung wird auch als OHMsches Gesetz bezeichnet. Sie verbindet die elektrische Stromdichte $\vec{j}$

mit der elektrischen Feldstärke über die elektrische Leitfähigkeit $\sigma = 1/\varrho$, wobei ϱ für den spezifischen elektrischen Widerstand steht. Die Materialparameter μ, ε und σ sind im Allgemeinen tensorielle Größen, die von Druck, Temperatur und den Feldgrößen selbst abhängen können. Für die später durchgeführten Modellierungen führt die Wahl von linearen, isotropen und homogenen Materialparametern für einzelne Teilgebiete zu einer Vereinfachung der Modellrechnung.

2.1.2 Die Kontinuitätsgleichung

Die folgenden Untersuchungen sollen für die Gleichstromgeoelektrik durchgeführt werden. Dieses geophysikalische Messverfahren verwendet stationäre Felder und Ströme. Die MAXWELLschen Gleichungen vereinfachen sich damit, da die partiellen Ableitungen nach der Zeit gleich Null sind. Gleichung (2.2) kann damit umgeformt werden zu

$$\operatorname{rot} \vec{E} = 0\,, \tag{2.8}$$

womit das elektrische Feld wirbelfrei ist und als Gradient des elektrischen Skalarpotentials φ dargestellt werden kann

$$\vec{E} = -\operatorname{grad} \varphi. \tag{2.9}$$

In jedem Punkt des betrachteten Gebietes Ω gilt die Kontinuitätsgleichung, die sich durch Bilden der Divergenz von Gleichung (2.1) ergibt

$$\operatorname{div} \vec{j} = Q, \tag{2.10}$$

wobei Q einen Quellterm bezeichnet. In der Geoelektrik sind die Quellen Elektroden, über die dem Boden galvanisch Strom zugeführt wird. Die Stromelektrode wird näherungsweise als Punktquelle betrachtet. Befindet sich die Quelle bei $\vec{r}_s$ und fließt ein elektrischer Strom I, dann kann Q als

$$Q = I\delta(\vec{r} - \vec{r}_s) \tag{2.11}$$

dargestellt werden. Dabei ist δ die DIRACsche Deltadistribution und $\vec{r}$ ein Ortsvektor innerhalb des Gebietes Ω. Durch Einsetzen folgt aus den Gleichungen (2.7), (2.9), (2.10) und (2.11) die partielle Differentialgleichung zweiter Ordnung

$$\operatorname{div}(\sigma \operatorname{grad} \varphi) = -I\delta(\vec{r} - \vec{r}_s). \tag{2.12}$$

Diese elliptische Differentialgleichung dient als Ausgangspunkt für alle weiteren Untersuchungen. Liegt die Quelle der Stromeinspeisung an der Oberfläche eines homogenen Halbraumes (homogen bedeutet, σ ist konstant in Ω), so hat Gleichung (2.12) mit der Randbedingung $\lim_{|\vec{r}|\to\infty} \varphi = 0$ die Lösung

$$\varphi(\vec{r}) = \frac{I}{2\pi\sigma|\vec{r} - \vec{r}_s|}\,. \tag{2.13}$$

2.1.3 Randbedingungen

Neben der partiellen Differentialgleichung (2.12) sind weitere Bedingungen an das Potential φ zu stellen. Da Gleichung (2.12) nur für Gebiete Ω_i $(i = 1, 2, \dots)$ gilt, in denen σ einmal stetig differenzierbar $\sigma \in C^1(\Omega_i)$ ist, müssen folgende Bedingungen an Diskontinuitäten von σ gefordert werden:

- Das Potential muss stetig sein

$$\varphi_1 = \varphi_2 \,. \tag{2.14}$$

- Die Normalkomponente der Stromdichte $\frac{\partial \vec{j}}{\partial \vec{n}} = \sigma \frac{\partial \varphi}{\partial \vec{n}}$ muss stetig sein

$$\sigma_1 \frac{\partial \varphi_1}{\partial \vec{n}} = \sigma_2 \frac{\partial \varphi_2}{\partial \vec{n}} \,. \tag{2.15}$$

An den äußeren Rändern des Modells müssen entsprechende Randbedingungen vorgegeben werden, die aus den physikalischen Gegebenheiten formuliert werden können. Für einen Rand $\Gamma = \partial\Omega$ des Gebietes Ω werden die CAUCHYschen Randbedingungen wie folgt beschrieben

$$\alpha \left(\varphi - \varphi_0\right) + \beta \frac{\partial \varphi}{\partial \vec{n}} + j_0 = 0 \quad \text{auf} \quad \Gamma. \tag{2.16}$$

Dabei bezeichnet $\vec{n}$ den äußeren Normalenvektor auf Γ, φ_0 und j_0 sind angenommene Werte des Potentials bzw. der Normalkomponente der Stromdichte auf Γ.
Über die Wahl der Werte für α und β wird der Typ der Randbedingung festgelegt. Wählt man $\alpha \neq 0$ und $\beta = 0$, so erhält man eine DIRICHLETsche Randbedingung. Wenn $\alpha = 0$ und $\beta \neq 0$ sind, spricht man von NEUMANNschen Randbedingungen. Gemischte Randbedingungen erhält man, wenn $\alpha, \beta \neq 0$ sind. Von einer homogenen Randbedingung vom jeweiligen Typ spricht man, wenn $\varphi_0 = 0$ und/oder $j_0 = 0$ ist.
Für ein Modell als Teil des Erdkörpers handelt es sich bei der Oberseite Γ_E in der Regel um die Erdoberfläche. Da durch diese Randfläche kein Strom fließen darf, fordert man homogene Randbedingungen 2. Art (NEUMANNsche Randbedingung)

$$\frac{\partial \varphi}{\partial \vec{n}} = 0 \quad \text{auf} \quad \Gamma_E. \tag{2.17}$$

Für den anderen Teil der Berandung Γ_∞ ($\Gamma = \Gamma_E \cup \Gamma_\infty$) können entweder Randbedingungen 1. oder 3. Art (DIRICHLETsche oder gemischte Randbedingungen) gefordert werden, die das Verhalten $\lim_{|\vec{r}|\to\infty} \varphi = 0$ approximieren. Für eine Randbedingung vom Typ DIRICHLET nimmt man an, dass das Potential auf dem Rand vernachlässigbar klein ist

$$\varphi = 0 \quad \text{auf} \quad \Gamma_\infty. \tag{2.18}$$

Dey und Morrison (1979) schlugen folgende gemischte Randbedingungen vor

$$\frac{\cos(\vec{r},\vec{n})}{|\vec{r}|}\,\varphi + \frac{\partial\varphi}{\partial\vec{n}} = 0 \quad \text{auf} \quad \Gamma_\infty. \tag{2.19}$$

Den Normalenvektor an einem Punkt auf dem Rand bezeichnet $\vec{n}$, wohingegen $\vec{r}$ den Vektor von der Quelle ($\vec{r}_s = (0,0,0)$) zum Punkt auf dem Rand bezeichnet. Dies wird in Abbildung 2.1 verdeutlicht. Durch Gleichung (2.19) wird ein asymptotisches Verhalten von $\varphi \sim \frac{1}{|\vec{r}|}$ auf dem Rand gefordert.

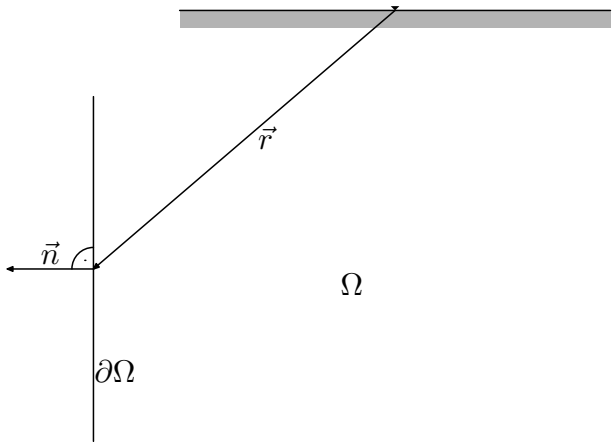


Abbildung 2.1: Gemischte Randbedingungen

2.1.4 Methodische Grundlagen

Das Verfahren der Gleichstromgeoelektrik verwendet künstlich in die Erde eingespeiste Gleichströme, um über die Messung von Potentialdifferenzen Informationen über die Verteilung der elektrischen Leitfähigkeit im Untergrund zu erhalten. Der Zusammenhang zwischen elektrischem Feld und Stromdichte ist durch das OHMsche Gesetz (Gleichung (2.7)) gegeben. Beide Größen sind aber nicht direkt bestimmbar. Messbar sind der über die Elektroden A und B eingespeiste Strom I und der damit zwischen den Elektroden M und N erzeugte Spannungsabfall U. In Abbildung 2.2 ist schematisch eine geoelektrische Messanordnung dargestellt.
Der spezifische elektrische Widerstand ϱ eines homogenen Halbraums ergibt sich aus dem Quotienten von U und I, der mit einem Konfigurationsfaktor k multipliziert wird

$$\varrho = k\frac{U}{I} \quad \text{mit} \quad I = I_B = -I_A\,. \tag{2.20}$$

Für Elektrodenanordnungen an der Erdoberfläche wird der Konfigurationsfaktor berechnet über

$$k = \frac{2\pi}{\frac{1}{|\overline{AM}|} - \frac{1}{|\overline{AN}|} - \frac{1}{|\overline{BM}|} + \frac{1}{|\overline{BN}|}}\,. \tag{2.21}$$

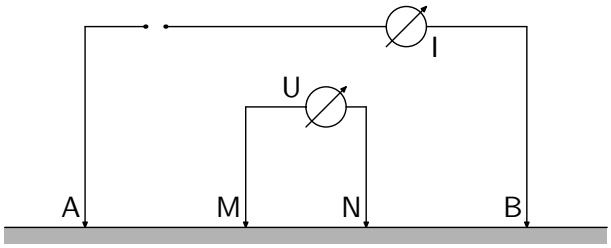


Abbildung 2.2: Geoelektrische Messanordnung

Auch wenn es sich bei dem zu untersuchenden Gebiet nicht um einen homogenen Halbraum handelt, sondern um ein Gebiet mit inhomogener Verteilung der elektrischen Leitfähigkeit, wird analog zu den Gleichungen (2.20) und (2.21) eine Größe

$$\varrho_s = k\frac{U}{I} \tag{2.22}$$

berechnet, die jetzt als scheinbarer spezifischer elektrischer Widerstand bezeichnet wird. Durch Verändern der Lage der Elektroden für die Stromeinspeisung (A und B) und der Elektroden für die Messung des Spannungsabfalls (M und N) kann der laterale bzw. vertikale Untersuchungsbereich verändert werden. Für spezielle Anordnungen der Elektroden kann die Berechnungsvorschrift für den Konfigurationsfaktor vereinfacht werden. Diese Anordnungen erhalten meist auch eine eigene Bezeichnung. Wird die Elektrode B im Unendlichen platziert, spricht man von einer Pol-Quelle und die Terme in Gleichung (2.21), in denen B vorkommt, gehen gegen Null und entfallen. Im Gegensatz dazu handelt es sich um eine Dipol-Quelle, wenn beide Stromelektroden (A und B) genutzt werden. Wendet man die gleichen Überlegungen auf die Spannungselektroden an, so führt dies in Kombination mit den Aussagen für die Stromelektroden zu charakteristischen Anordnungen. Man spricht von Dipol-Dipol-, Dipol-Pol-, Pol-Dipol- und Pol-Pol-Anordnungen. Nehmen die Abstände zwischen den Elektroden gewisse Vielfache voneinander ein, so spricht man von Wenner- und Schlumberger-Anordnungen (Abbildung 2.3). Weitere Details sind in Knödel et al. (1997) sowie Militzer und Weber (1985) zu finden.

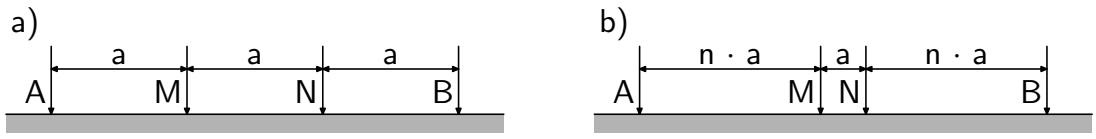


Abbildung 2.3: a) Wenner- und b) Schlumberger-Anordnung

2.2 Die Methode der Finiten Elemente

Eine kurze Einführung in die historische Entwicklung der Finite-Elemente-Methode (FEM) aus Sicht der Mathematik und Ingenieurwissenschaften befindet sich in Jung und Langer (2001). Für die Geophysik und speziell die Geoelektrik kann eine erste Beschreibung der Methode der Finiten Elemente (FE) in einem Artikel von Coggon (1971) nachgelesen werden. In der Folge erschienen zahlreiche weitere Arbeiten u. a. von Pridmore et al. (1981), Sasaki (1994), Bing und Greenhalgh (2001) sowie Li und Spitzer (2002). An deutschen Hochschulen wurden zu dieser Thematik in jüngster Vergangenheit Arbeiten von Rücker (1999) und Kemna (2000) vorgelegt. In den folgenden Abschnitten wird ein kurzer Abriss der notwendigen Schritte zur Diskretisierung mittels der Methode der Finiten Elemente gegeben. Dafür dient das Buch von Jung und Langer (2001) als Vorlage. In Braess (1997) sind ebenfalls ausführliche Darstellungen zu finden.

2.2.1 Die schwache Formulierung

Im Gegensatz zu Finite-Differenzenverfahren bildet für die FE-Diskretisierung nicht die klassische Formulierung (Gleichungen (2.12), (2.14), (2.15) und (2.16))

Gesucht ist $\varphi \in C^2(\Omega) \cap C^1(\Omega \cup \Gamma_2 \cup \Gamma_3) \cap C(\overline{\Omega})$, sodass

$$\begin{aligned} \operatorname{div}(\sigma \operatorname{grad} \varphi) &= -I\delta(\vec{r} - \vec{r}_s) \quad \text{auf } \Omega, \\ \varphi &= g_1 \quad \text{auf } \Gamma_1, \\ \frac{\partial \varphi}{\partial \vec{n}} &= g_2 \quad \text{auf } \Gamma_2, \\ \frac{\partial \varphi}{\partial \vec{n}} + \alpha\varphi &= \alpha\varphi_0 \quad \text{auf } \Gamma_3, \end{aligned} \tag{2.23}$$

gilt mit $\Gamma = \partial\Omega = \overline{\Gamma}_1 \cup \overline{\Gamma}_2 \cup \overline{\Gamma}_3$ und $\Gamma_i \cap \Gamma_j = \emptyset$ für $i \neq j$, $\sigma \in C^1(\Omega)$.

den Ausgangspunkt, sondern die Variationsformulierung, die auch als schwache Formulierung bezeichnet wird. Für die Ableitung der Variationsformulierung zur Randwertaufgabe (2.23) wird der Raum von Testfunktionen

$$V_0 = \{\psi \in H^1(\Omega) : \psi \equiv 0 \quad \text{auf } \Gamma_1\} \tag{2.24}$$

eingeführt, der alle quadratische integrierbaren Funktionen enthält, die auf Γ_1 verschwinden und deren erste verallgemeinerte Ableitungen quadratisch integrierbar sind. Nach Multiplikation der Differentialgleichung der Randwertaufgabe (2.23) mit einer Testfunktion $\psi \in V_0$ und Integration über Ω erhält man

$$\int_\Omega \operatorname{div}(\sigma \operatorname{grad} \varphi)\, \psi \, d\vec{x} = \int_\Omega -I\delta(\vec{r} - \vec{r}_s)\, \psi \, d\vec{x}\,. \tag{2.25}$$

Die Anwendung der GREENschen Formel

$$\int_\Omega \operatorname{div}(\lambda \operatorname{grad} u)\, v\, d\vec{x} = -\int_\Omega (\operatorname{grad} v)^T \lambda \operatorname{grad} u\, d\vec{x} + \int_{\partial\Omega} \lambda \frac{\partial u}{\partial \vec{n}}\, v\, ds \tag{2.26}$$

liefert die äquivalente Beziehung zu Gleichung (2.25)

$$\int_\Omega (\operatorname{grad}\psi)^T \sigma \operatorname{grad}\varphi\, d\vec{x} - \int_{\partial\Omega} \sigma \frac{\partial\varphi}{\partial\vec{n}}\,\psi\, ds = \int_\Omega I\delta(\vec{r}-\vec{r}_s)\,\psi\, d\vec{x}\,. \tag{2.27}$$

Im nächsten Schritt werden die auf Γ_2 und Γ_3 vorgegebenen Randbedingungen einbezogen. Wegen $\psi \equiv 0$ auf Γ_1 ist

$$\int_{\partial\Omega} \sigma \frac{\partial\varphi}{\partial\vec{n}}\,\psi\, ds = \int_{\Gamma_1} \sigma \frac{\partial\varphi}{\partial\vec{n}}\,\psi\, ds + \int_{\Gamma_2} \sigma \frac{\partial\varphi}{\partial\vec{n}}\,\psi\, ds + \int_{\Gamma_3} \sigma \frac{\partial\varphi}{\partial\vec{n}}\,\psi\, ds \tag{2.28}$$

$$= 0 + \int_{\Gamma_2} \sigma\, g_2\,\psi\, ds + \int_{\Gamma_3} \sigma\,\alpha(\varphi_0-\varphi)\,\psi\, ds \tag{2.29}$$

und es gilt

$$\int_\Omega (\operatorname{grad}\psi)^T \sigma \operatorname{grad}\varphi\, d\vec{x} + \int_{\Gamma_3} \sigma\,\alpha\varphi\,\psi\, ds = \int_\Omega I\delta(\vec{r}-\vec{r}_s)\,\psi\, ds + \int_{\Gamma_2} \sigma\, g_2\,\psi\, ds + \int_{\Gamma_3} \sigma\,\alpha\varphi_0\,\psi\, ds\,. \tag{2.30}$$

Unter Beachtung der Randbedingungen 1. Art legt man die Menge der zulässigen Funktionen

$$V_{g_1} = \{\varphi \in H^1(\Omega) : \varphi = g_1 \quad \text{auf } \Gamma_1\}\,. \tag{2.31}$$

fest, in der die Lösung φ gesucht wird. Damit lautet die Variationsformulierung zu (2.23):

Gesucht ist $\varphi \in V_{g_1}$, sodass

$$a(\varphi,\psi) = l(\psi) \quad \forall\psi \in V_0 \tag{2.32}$$

gilt mit

$$a(\varphi,\psi) = \int_\Omega (\operatorname{grad}\psi)^T \sigma \operatorname{grad}\varphi\, d\vec{x} + \int_{\Gamma_3} \sigma\,\alpha\varphi\,\psi\, ds\,,$$

$$l(\psi) = \int_\Omega I\delta(\vec{r}-\vec{r}_s)\,\psi\, ds + \int_{\Gamma_2} \sigma\, g_2\,\psi\, ds + \int_{\Gamma_3} \sigma\,\alpha\varphi_0\,\psi\, ds\,,$$

$$V_{g_1} = \{\varphi \in H^1(\Omega) : \varphi = g_1 \quad \text{auf } \Gamma_1\}\,,$$

$$V_0 = \{\psi \in H^1(\Omega) : \psi = 0 \quad \text{auf } \Gamma_1\}\,,$$

$$\sigma \in L^2(\Omega), \sigma \in L^2(\Gamma_3).$$

Wie aus der Aufgabe (2.32) hervorgeht, nehmen die Randbedingungen in unterschiedlicher Weise Einfluss auf die Variationsformulierung. Auf die lineare Mannigfaltigkeit, d.h. auf die Definition der zulässigen Funktionen V_{g_1} und die Menge der Testfunktionen V_0 hat die Randbedingung 1. Art (wesentliche Randbedingung) Einfluss. Dahingegen führen die Randbedingungen 2. und 3. Art (natürliche Randbedingungen) zu weiteren Termen in der symmetrischen, positiv definiten Bilinearform $a(.,.)$ und der Linearform $l(.)$.
Die Variationsformulierung oder schwache Formulierung als eine Verallgemeinerung der klassischen Formulierung stellt geringere Glattheitsforderungen an die Eingangsdaten σ und die Lösung φ der Aufgabe als die klassische Formulierung. Wendet man die aus den physikalischen Gegebenheiten gewonnene Formulierung für die gemischten Randbedingungen auf Γ_∞ und NEUMANNsche Randbedingungen auf Γ_E (Abschnitt 2.1.3) auf die schwache Formulierung (2.32) an, so erhält man die schwache Formulierung der gleichstromgeoelektrischen Randwertaufgabe:

Gesucht ist $\varphi \in V$, sodass

$$a(\varphi, \psi) = l(\psi) \quad \forall \psi \in V \tag{2.33}$$

gilt mit

$$a(\varphi,\psi) = \int_\Omega (\operatorname{grad}\psi)^T \,\sigma \operatorname{grad}\varphi \, d\vec{x} + \int_{\Gamma_\infty} \sigma \,\frac{\cos(\vec{r},\vec{n})}{|\vec{r}|}\varphi\,\psi\, ds\,,$$

$$l(\psi) = \int_\Omega I\delta(\vec{r}-\vec{r}_s)\,\psi\, ds\,,$$

$$V = \{\varphi, \psi \in H^1(\Omega)\}\,, \sigma \in L^2(\Omega), \sigma \in L^2(\Gamma_\infty).$$

2.2.2 Das GALERKIN-Verfahren

Die schwache Formulierung (2.33) benutzt einen unendlichdimensionalen Funktionenraum. Da dieser aber zu groß und komplex ist, wird beim GALERKIN-Verfahren der Raum V durch einen endlichdimensionalen Raum

$$V_h \subset V\,, \quad dim\, V_h = n < \infty \tag{2.34}$$

approximiert. Damit lautet die diskrete schwache Formulierung zu (2.33):

Gesucht ist $\varphi_h \in V_h$, sodass

$$a(\varphi_h, \psi_h) = l(\psi_h) \qquad \forall \psi_h \in V_h \tag{2.35}$$

gilt.

Da die Dimension des Raumes V_h kleiner als unendlich ist, gibt es einen endlichen Satz linear unabhängiger Funktionen $\phi_i \in V_h \quad (i = 1, \ldots, n)$, die die Basis von V_h bilden. Diese Funktionen werden auch als Ansatz- oder Testfunktionen bezeichnet. Damit gibt es für jedes $v \in V_h$ eine Darstellung der Form

$$v = \sum_{i=1}^{n} v_i \phi_i \,. \tag{2.36}$$

Unter Ausnutzung der Linearitätseigenschaften von $a(.,.)$ kann $a(\varphi_h, \psi_h)$ dargestellt werden als

$$a(\varphi_h, \psi_h) = a(\sum_{i=1}^{n} \varphi_i \phi_i, \psi_h) = \sum_{i=1}^{n} \varphi_i a(\phi_i, \psi_h) \,. \tag{2.37}$$

Mit der zu (2.35) äquivalenten Bedingung

$$a(\varphi_h, \phi_j) = l(\phi_j) \qquad \forall j = 1, \ldots, n \tag{2.38}$$

lautet die Aufgabe (2.35):

Gesucht ist $\{\varphi_i\}_{i=1}^{n}$, sodass

$$\sum_{i=1}^{n} \varphi_i a(\phi_i, \phi_j) = l(\phi_j) \qquad \forall j = 1, \ldots, n \,. \tag{2.39}$$

Mit den Bezeichnungen $A_h \in \mathbb{R}^{n \times n}$, $\vec{f}_h, \vec{\varphi}_h \in \mathbb{R}^n$,

$$\begin{aligned} A_h &= (a_{ij})_{i,j=1}^{n} \,, a_{ij} = a(\phi_i, \phi_j) \,, \\ \vec{f}_h &= (f_1, \ldots, f_n)^T \,, f_j = l(\phi_j) \,, \\ \vec{\varphi}_h &= (\varphi_1, \ldots, \varphi_n)^T \end{aligned}$$

kann das lineare Gleichungssystem (2.39) in der Matrixform

$$A_h \vec{\varphi}_h = \vec{f}_h \tag{2.40}$$

geschrieben werden. Über die Lösung des linearen Gleichungssystems nach den Koeffizienten $\{\varphi_i\}_{i=1}^{n}$ erhält man die approximierte Lösung $\varphi_h(\vec{r}) = \sum_{i=1}^{n} \varphi_i \phi_i(\vec{r}), \vec{r} \in \Omega$ des Randwertproblems der Gleichstromgeoelektrik.

2.2.3 Einzelne Probleme der Finite-Elemente-Methode

Die Umsetzung der Methode der Finiten Elemente ist im Vergleich zum klassischen Finite-Differenzenverfahren schwieriger. Von der diskreten schwachen Formulierung (2.35) ausgehend soll nun näher eingegangen werden auf

- die Diskretisierung des Gebietes Ω,
- die Wahl der Ansatzfunktionen,
- den Aufbau des linearen Gleichungssystems,
- die numerische Integration und
- die Lösung des linearen Gleichungssystems.

Diskretisierung des Gebietes Ω

Das der Modellrechnung zu Grunde liegende, endliche Gebiet Ω, in dem man die Lösung des Randwertproblems sucht, wird in finite Elemente $T^{(r)}$ zerlegt. Eine solche Zerlegung

$$\mathcal{T}_h = \{T^{(r)} : r = 1, 2, \ldots, R_h\}\,, \tag{2.41}$$

auch Triangulierung oder Vernetzung genannt, soll folgende Eigenschaften besitzen:

- $\overline{\Omega} = \bigcup_{r=1}^{R_h} \overline{T}^{(r)}$ bzw. $\overline{\Omega}_h = \bigcup_{r=1}^{R_h} \overline{T}^{(r)} \longrightarrow \overline{\Omega}$ für $h \to 0$, d. h. die Vereinigung aller Elemente $T^{(r)}$ soll das Gebiet Ω exakt überdecken oder bei kleiner werdender Diskretisierungsschrittweite immer besser approximieren.
- Für alle $r, r' \in \{1, 2, \ldots, R_h\}$ mit $r \neq r'$ ist

$$\overline{T}^{(r)} \cap \overline{T}^{(r')} = \begin{cases} \emptyset & \text{oder} \\ \text{eine gemeinsame Ecke} & \text{oder} \\ \text{eine gemeinsame Kante} & \text{oder} \\ \text{eine gemeinsame Fläche .} \end{cases} \tag{2.42}$$

Gebräuchliche finite Elemente für drei Raumrichtungen sind Tetraeder und Hexaeder. Unter dem nun schon häufiger genutzten Zusatz h wird der maximale Elementdurchmesser aller Elemente verstanden,

$$h = \max_{r=1,2,\ldots,R_h} h^{(r)}\,,$$

wobei der Elementdurchmesser $h^{(r)}$ für ein Element $T^{(r)}$ als maximaler Abstand zweier beliebiger Punkte in $\overline{T}^{(r)}$ festgelegt ist.

Wenn

- die Anzahl der Elemente gegen unendlich strebt und h gegen Null konvergiert sowie
- das Verhältnis zwischen $h^{(r)}$ und dem Durchmesser $d^{(r)}$ der größten in $T^{(r)}$ enthaltenen Kugel nach oben durch eine positive Konstante C beschränkt ist,

spricht man von einer regulären Vernetzung $\mathcal{T}_h$ des Gebietes Ω. Zur Beschreibung einer Vernetzung werden alle Knoten $P_i\,(i = 1, 2, \ldots, P_h)$ und Elemente $T^{(r)}\,(r = 1, 2, \ldots, R_h)$ global nummeriert. Zusätzlich werden in jedem Element $T^{(r)}$ die Knoten $P_\alpha^{(r)}$ lokal von $\alpha = 1, \ldots, \hat{N}$ nummeriert. $\hat{N}$ bezeichnet die Anzahl der Knoten pro Element. In Abbildung 2.4 ist die lokale Knotennummerierung für ein lineares Tetraederelement (Tetraeder mit linearen Ansatzfunktionen s. u.) dargestellt. Für die rechentechnische Umsetzung ist eine Elementzusammenhangs-

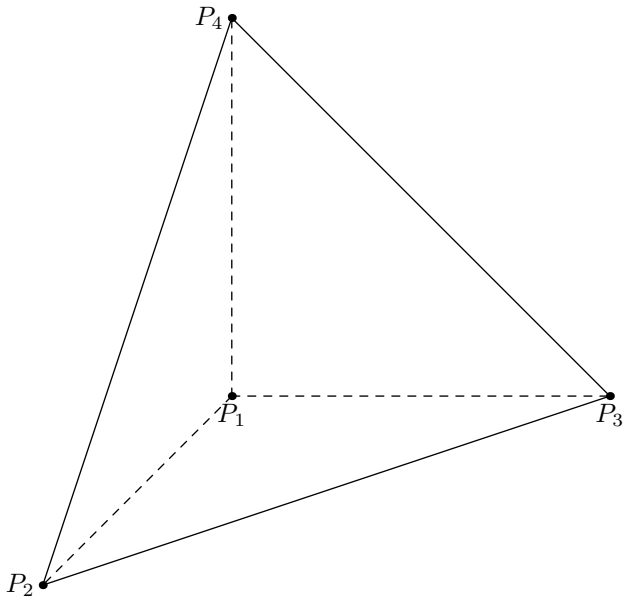


Abbildung 2.4: Lokale Knotennummerierung für ein lineares Tetraederelement

tabelle notwendig. In dieser wird jeder lokalen Knotennummer eines Elementes eine globale Knotennummer zugeordnet. In einer weiteren Tabelle werden für jeden global nummerierten Knoten die Raumkoordinaten festgehalten. Damit kann die Lage jedes Elementes im Raum bestimmt werden.

Zur Generierung einer Vernetzung existieren verschiedene Methoden. Eine vollständige Beschreibung der Verfahren übersteigt den Rahmen dieser Arbeit. Darum sollen nur kurz die Schritte während der automatischen Generierung einer Vernetzung dargestellt werden. Einen detaillierteren Überblick über die verschiedenen Verfahren findet man z. B. in Owen (1998). Den Ausgangspunkt bildet die geometrische Beschreibung des Gebietes durch Punkte, Linien oder Flächen im Raum. Durch Zusammenfassen von Gebieten gleicher Materialparameter

(z. B. gleicher elektrischer Leitfähigkeit) wird garantiert, dass Materialübergänge mit den Seitenflächen/Kanten von Elementen zusammenfallen. Für jedes Teilgebiet mit festem Materialparameter wird eine Oberflächenvernetzung generiert. Diese muss so erzeugt werden, dass sie an den Schnittflächen zweier Teilgebiete identisch ist. Die Vernetzung der Oberfläche bildet den Ausgangspunkt für die Generierung von inneren Knoten und Elementen. Nachdem alle Teilgebiete relativ grob vernetzt sind, werden alle Elemente bestimmt, für die das Verhältnis aus Elementdurchmesser zu Durchmesser der größten enthaltenen Kugel einen vorgegebenen Wert übersteigt. Diese Elemente werden entweder regulär oder irregulär verfeinert. Beide Varianten sind in den Abbildungen 2.5 und 2.6 für ein Tetraederelement dargestellt. Die Elemente der erzeugten Triangulierung werden so lange verfeinert, bis das Verhältnis die vorgegebene Schranke unterschreitet. Eine weitere Qualitätsverbesserung darüber hinaus ist durch eine Netzglät-

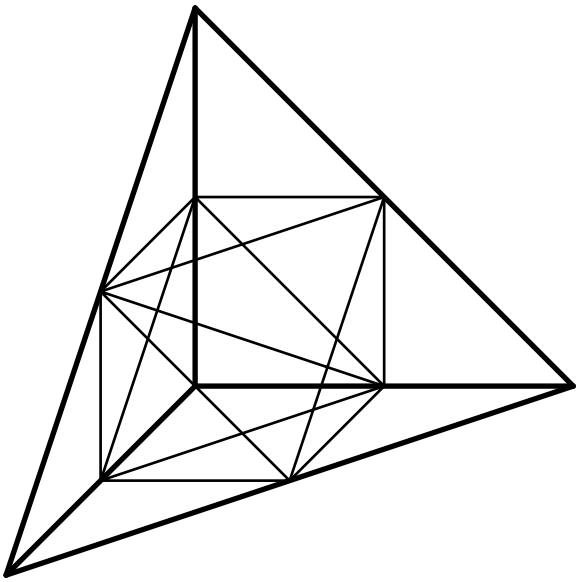


Abbildung 2.5: Reguläre Verfeinerung eines Tetraederelementes in 8 Tetraeder

tung möglich. Dabei werden alle Knoten in den Schwerpunkt ihrer Nachbarknoten verschoben. Knoten auf den Randflächen oder an Materialübergängen dürfen nur entlang dieser Grenzen verschoben werden.

Wahl der Ansatzfunktionen

Die grundlegende Idee der FEM ist es, als Ansatz- oder Testfunktionen ϕ_i solche zu verwenden, die einen kleinen Träger besitzen. Man wählt global stetige, stückweise polynomiale Funktionen, die nur über sehr wenigen Elementen von Null verschieden sind. Die Wahl von Ansatzfunktionen mit lokalem Träger führt zu einer schwach besetzten Steifigkeitsmatrix A_h. Die Ansatz- und Testfunktionen ϕ_i werden lokal über den finiten Elementen $T^{(r)}$, die den Knoten P_i enthalten, durch Formfunktionen $\phi_\alpha^{(r)}$ definiert. Diese werden durch Abbildung von Formfunktionen

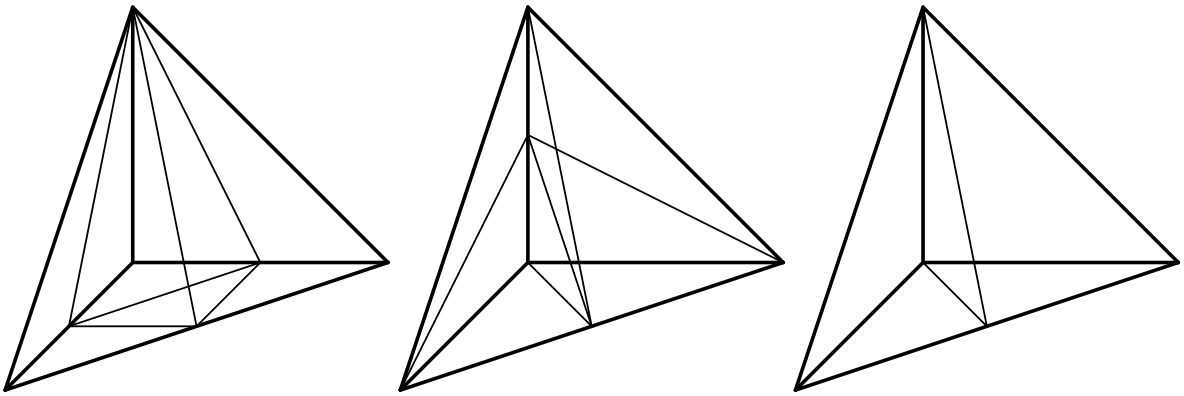


Abbildung 2.6: Irreguläre Verfeinerungsvarianten für ein Tetraederelement in 4, 4 bzw. 2 Tetraeder

erhalten, die auf Referenzelementen festgelegt sind. Für eine Vernetzung mit Tetraederelementen ist das Referenztetraeder $\hat{T}$ (Abbildung 2.7) definiert durch

$$\hat{T} = \{(\xi_1, \xi_2, \xi_3) : 0 \leq \xi_1, \xi_2, \xi_3 \leq 1, \xi_1 + \xi_2 + \xi_3 \leq 1\}. \tag{2.43}$$

Über die affine Transformationsvorschrift

$$\vec{x} = \vec{x}_{T^{(r)}}(\vec{\xi}) = J^{(r)}\vec{\xi} + \vec{x_1}^{(r)} \tag{2.44}$$

wird die Abbildung des Referenztetraeders $\hat{T}$ auf ein beliebiges Tetraeder $T^{(r)}$ der Vernetzung realisiert. In ausführlicher Schreibweise hat diese die Form

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} x_{2,1}^{(r)} - x_{1,1}^{(r)} & x_{3,1}^{(r)} - x_{1,1}^{(r)} & x_{4,1}^{(r)} - x_{1,1}^{(r)} \\ x_{2,2}^{(r)} - x_{1,2}^{(r)} & x_{3,2}^{(r)} - x_{1,2}^{(r)} & x_{4,2}^{(r)} - x_{1,2}^{(r)} \\ x_{2,3}^{(r)} - x_{1,3}^{(r)} & x_{3,3}^{(r)} - x_{1,3}^{(r)} & x_{4,3}^{(r)} - x_{1,3}^{(r)} \end{pmatrix} \begin{pmatrix} \xi_1 \\ \xi_2 \\ \xi_3 \end{pmatrix} + \begin{pmatrix} x_{1,1}^{(r)} \\ x_{1,2}^{(r)} \\ x_{1,3}^{(r)} \end{pmatrix}. \tag{2.45}$$

Die Koordinaten der Knoten $P_\alpha^{(r)}, \alpha \in [1, 2, \ldots, \hat{N}]$ im Tetraederelement $T^{(r)}$ sind dabei durch $(x_{\alpha,1}^{(r)}, x_{\alpha,2}^{(r)}, x_{\alpha,3}^{(r)})$ gegeben. Die Transformation des Referenztetraeders auf ein beliebiges Tetraeder der Vernetzung wird in Abbildung 2.7 verdeutlicht. Über die Zuordnungsvorschrift

$$\alpha(r) \leftrightarrow i = i(r, \alpha) \tag{2.46}$$

zwischen der globalen Knotennummerierung i und der lokalen Knotennummerierung α in jedem Element $T^{(r)}$ werden die globalen Ansatzfunktionen definiert durch

$$\phi_i(\vec{x}) = \begin{cases} \phi_\alpha^{(r)}(\vec{x}) & \vec{x} \in \overline{T}^{(r)}, \quad r \in B_i = \{r : P_i \in \overline{T}^{(r)}\} \\ 0 & \text{sonst.} \end{cases} \tag{2.47}$$

Dabei ist B_i die Indexmenge aller Elemente $T^{(r)}$, für die $P_i \in \overline{T}^{(r)}$ gilt. Die $\phi_\alpha^{(r)}$ erhält man durch Transformation der auf dem Referenzelement definierten Formfunktionen p_α

$$\phi_\alpha^{(r)}(\vec{x}) = p_\alpha(\vec{\xi}_{T^{(r)}}(\vec{x})), \qquad \vec{x} \in \overline{T}^{(r)} \,. \tag{2.48}$$

Über dem Referenzelement werden polynomiale Funktionen p_α als Formfunktionen gewählt, für die in den Knoten $\hat{P}_\beta$

$$p_\alpha(\vec{\xi}_\beta) = \delta_{\alpha\beta} = \begin{cases} 1 & \alpha = \beta \\ 0 & \alpha \neq \beta \end{cases} \tag{2.49}$$

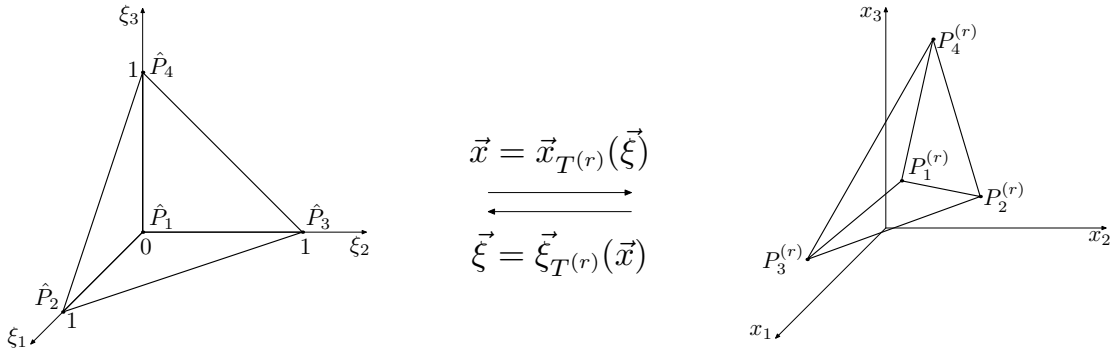
erfüllt ist, wobei $\alpha, \beta \in [1, 2, \ldots, \hat{N}]$. Damit gilt

$$\phi_i(P_j) = \delta_{ij} \tag{2.50}$$

für die in Gleichung (2.47) festgelegten Funktionen ϕ_i. Die beiden Indizes i und j gehören der Menge der globalen Knotennummern an. Lineare Formfunktionen über dem Referenztetraeder $\hat{T}$ haben die Gestalt

$$\begin{aligned} p_1 &= 1 - \xi_1 - \xi_2 - \xi_3 \,, \\ p_2 &= \xi_1 \,, \\ p_3 &= \xi_2 \,, \\ p_4 &= \xi_3 \,. \end{aligned}$$

Die Funktionen aus dem Raum V lassen sich bei feiner werdender Zerlegung immer besser durch Funktionen aus V_h approximieren, $\lim_{h\to 0} V_h = V$.



Referenztetraeder $\hat{T}$ beliebiges Tetraeder $T^{(r)}$

Abbildung 2.7: Abbildung zwischen dem Referenztetraeder $\hat{T}$ und einem beliebigen Tetraeder $T^{(r)} \in \mathcal{T}_h$ der Vernetzung

Aufbau des linearen Gleichungssystems

Das lineare Gleichungssystem wird elementweise aufgebaut. Dabei wird für jedes Element $T^{(r)} \in \mathcal{T}_h$ sein Beitrag zur Steifigkeitsmatrix A_h und zum Lastvektor $\vec{f}_h$ berechnet. Für ein Element $T^{(r)}$ mit $\hat{N}$ Knoten erhält man die sogenannte Elementsteifigkeitsmatrix $A_h^{(r)}$ der Größe $\hat{N} \times \hat{N}$. Diese wird dann durch Aufaddieren an den entsprechenden Stellen (Knoten) in die globale Steifigkeitsmatrix A_h eingebaut. Auf gleiche Weise wird der Lastvektor $\vec{f}_h$ aus den einzelnen Elementlastvektoren $\vec{f}_h^{(r)}$ assembliert. Beispielhaft soll hier die Berechnung der Elementsteifigkeitsmatrix ohne Berücksichtigung der Randbedingungen beschrieben werden. Den Ausgangspunkt bildet Gleichung (2.39), die für die spezielle Wahl einer Ansatzfunktion ϕ_i und Testfunktion ϕ_j jeweils einen Eintrag in der Steifigkeitsmatrix liefert. Ist σ elementweise konstant, gilt:

$$a(\phi_i, \phi_j) = \int_\Omega (\operatorname{grad} \phi_i)^T \sigma (\operatorname{grad} \phi_j) d\vec{x} \tag{2.51}$$

$$= \sum_{r=1}^{R_h} \sigma \int_{T^{(r)}} (\operatorname{grad} \phi_i)^T (\operatorname{grad} \phi_j) d\vec{x}\,. \tag{2.52}$$

Über die Verknüpfungen

$$\alpha \leftrightarrow i = i(r, \alpha) \qquad \text{und} \qquad \beta \leftrightarrow j = j(r, \beta)$$

zwischen der globalen Knotennummerierung (i und j) und der lokalen Knotennummerierung (α und β) ergibt sich mit der Definition (2.47) für ein Element $T^{(r)}$

$$\int_{T^{(r)}} (\operatorname{grad} \phi_i)^T (\operatorname{grad} \phi_j) d\vec{x} = \int_{T^{(r)}} (\operatorname{grad} \phi_\alpha^{(r)})^T (\operatorname{grad} \phi_\beta^{(r)}) d\vec{x}\,. \tag{2.53}$$

Die Berechnung des Integrals wird durch eine Variablensubstitution in die Integration über das Referenzelement $\hat{T}$ überführt. Gemäß den Abbildungsvorschriften (2.44) und (2.45) wird der Gradient transformiert zu

$$\operatorname{grad}_x = (J^{(r)})^{-T} \operatorname{grad}_\xi\,.$$

Mit Beziehung (2.48) und $\phi_\beta^{(r)}(x) = p_\beta(\xi_{T^{(r)}}(x)) = p_\beta(\xi)$ kann das Integral dargestellt werden als

$$\int_{T^{(r)}} \left[(\operatorname{grad} \phi_\alpha^{(r)})^T \operatorname{grad} \phi_\beta^{(r)}\right] d\vec{x}$$

$$= \int_{\hat{T}} \left[(\operatorname{grad}_\xi p_\alpha(\xi))^T (J^{(r)})^{-1} (J^{(r)})^{-T} \operatorname{grad}_\xi p_\beta(\xi)\right] |det\, J^{(r)}| d\xi\,. \tag{2.54}$$

Zur Berechnung der Elementsteifigkeitsmatrix werden nur die Ableitungen der Formfunktionen über dem Referenzelement und die Matrizen $J^{(r)}$, $(J^{(r)})^{-1}$ benötigt. Zur Assemblierung der Steifigkeitsmatrix A_h wird die Verknüpfung zwischen der globalen Knotennummerierung und der lokalen Knotennummerierung verwendet. Die Einträge der Matrix $A_h^{(r)}$ werden additiv an den entsprechenden Positionen in A_h eingefügt. Der Aufbau der rechten Seite $\vec{f}_h$ und der Einbau der natürlichen Randbedingungen erfolgt analog. Alle Elemente der Hauptdiagonalen von $A_h^{(r)}$ werden auf Eins und die anderen Elemente auf Null gesetzt, um homogene Randbedingung 1. Art zu integrieren.

Numerische Integration

Die während des Aufbaus der Elementsteifigkeitsmatrix und des Elementlastvektors zu bestimmenden Integrale können nicht immer analytisch berechnet werden. Deshalb nutzt man numerische Integrationsformeln zur näherungsweisen Berechnung. Wie im vorherigen Abschnitt dargestellt werden die Integrale nicht über dem Element $T^{(r)}$, sondern über dem Referenzelement $\hat{T}$ berechnet. Die verwendeten Quadraturformeln haben folgende Gestalt

$$\int_{\hat{T}} w(\xi) d\xi \approx \sum_{i=1}^{l} \alpha_i w(\xi_i)\,, \tag{2.55}$$

wobei $w(\xi_i)$ der Integrand, α_i die Quadraturgewichte und ξ_i die Quadraturstützstellen sind. Bei NEWTON-COTES-Quadraturformeln werden die Stützstellen vorgegeben und die Gewichte so gewählt, dass ein möglichst hoher Exaktheitsgrad erreicht wird. Dem gegenüber wird bei GAUSS-Quadraturformeln neben den Gewichten auch die Lage der Stützstellen verändert, um eine Maximierung des Exaktheitsgrades zu erreichen. Dies führt oft zu weniger Stützstellen bei gleichem Exaktheitsgrad im Vergleich zu den NEWTON-COTES-Formeln. Wenn auf den Elementen als Basisfunktionen Polynome der Ordnung p verwendet werden, müssen zur exakten Auswertung von Gleichung (2.54) die Quadraturformeln Polynome vom Grad $2p-2$ exakt integrieren.

Lösung des linearen Gleichungssystems

Das Resultat der Finite-Elemente-Diskretisierung ist ein lineares Gleichungssystem $A_h\vec{\varphi}_h = \vec{f}_h$. Auf Grund der symmetrischen und positiven Bilinearform ist die im Gleichungssystem enthaltene Systemmatrix A_h symmetrisch und positiv definit. Das entstehende Gleichungssystem ist im Allgemeinen sehr groß. Dabei wächst die Dimension mit kleiner werdendem Diskretisierungsparameter h an. Eine Halbierung von h hat eine Verachtfachung der Anzahl der Unbekannten in dreidimensionalen Gebieten zur Folge. Durch die Wahl von Ansatz- und Testfunktionen mit lokalem Träger ist die Systemmatrix schwach besetzt, d.h. in jeder Zeile der Matrix A_h sind nur wenige Einträge von Null verschieden. Bei einer geeigneten Knotennummerierung

hat die Systemmatrix eine Bandstruktur. Die aufgezählten Eigenschaften können geschickt für die effiziente Lösung des linearen Gleichungssystems ausgenutzt werden.
Zur Lösung des linearen Gleichungssystems unterscheidet man direkte und iterative Verfahren. Die exakte Lösung nach einer endlichen Anzahl arithmetischer Operationen bei exakter Arithmetik wird durch direkte Verfahren bestimmt. Dagegen berechnen iterative Verfahren den Lösungsvektor $\vec{\varphi}_h$ ausgehend von einer Startnäherung $\vec{\varphi}_h^{(0)}$ als Grenzwert einer Folge von Näherungslösungen $\vec{\varphi}_h^{(k)}, k = 1, 2, \ldots$, die gegen die exakte Lösung konvergieren. Bei der Erläuterung der einzelnen Methoden wird im Folgenden der Diskretisierungsparameter h als Index der Einfachheit halber weggelassen.
Als ein direktes Verfahren ist der Gauß-Algorithmus bekannt. Sollen die Symmetrie und positive Definitheit der Systemmatrix ausgenutzt werden, wird häufig das Cholesky-Verfahren angewandt. Bei diesem Verfahren wird die Systemmatrix in eine obere Dreiecksmatrix S transformiert, sodass

$$A = S^T S$$

gilt. Die Lösung von $A\vec{\varphi} = \vec{f}$ wird in zwei Abschnitten durchgeführt. Durch Vorwärtseinsetzen wird zunächst

$$S^T \vec{y} = \vec{f} \tag{2.56}$$

und im nächsten Schritt

$$S\vec{\varphi} = \vec{y} \tag{2.57}$$

durch Rückwärtseinsetzen gelöst. Man erhält die Lösung des linearen Gleichungssystems und damit der Randwertaufgabe. Bei der Durchführung dieser Faktorisierung bleiben gewisse Besetztheitsstrukturen der Matrix A erhalten. Die Kenntnis dieser Tatsache kann bei der kompakten Abspeicherung der Matrizen ausgenutzt werden und führt zu einer besseren Nutzung der vorhandenen Ressourcen.
Zu den stationären iterativen Gleichungslösern zählen das Jacobi-, Gauß-Seidel-, SOR- (successive overrelaxation) und SSOR-Verfahren (symmetric SOR). Sie sind algorithmisch einfach und damit leicht zu implementieren, konvergieren aber nur sehr langsam gegen die gesuchte Lösung. Die drei genannten Verfahren lassen sich in einer einheitlichen Schreibweise formulieren.

Gegeben sei eine Startnäherung $\vec{\varphi}^{(0)}$.
Iteration: $(k = 1, 2, \ldots)$

$$\vec{r}^{(k-1)} = \vec{f} - A\vec{\varphi}^{(k-1)}$$
$$\text{Löse} \quad C\vec{\omega}^{(k-1)} = \vec{r}^{(k-1)} \quad \text{nach} \quad \vec{\omega}^{(k-1)}$$
$$\vec{\varphi}^{(k)} = \vec{\varphi}^{(k-1)} + \tau\vec{\omega}^{(k-1)}$$

Durch die additive Zerlegung $A = L + D + L^T$ der Matrix A werden eine strenge untere Dreiecksmatrix L und $D = diag(A)$ als die Diagonalmatrix von A definiert. Die Wahl der Matrix C und des Iterationsparameters τ bestimmt den Typ des Iterationsverfahren:

$C = D\,, \tau = 1$	$\longrightarrow$	Jacobi-Verfahren
$C = D + L\,, \tau = 1$	$\longrightarrow$	Gauß-Seidel-Verfahren
$C = D + \omega L\,, \tau = \omega \in (0, 2)$	$\longrightarrow$	SOR-Verfahren
$C = (D + \omega L)D^{-1}(D + \omega L^T)\,, \tau = \omega(2 - \omega)\,, \omega \in (0, 2)$	$\longrightarrow$	SSOR-Verfahren.

Mit ω wird der Relaxationsparameter bezeichnet. Aus der einheitlichen Darstellung für die Iterationsverfahren kann eine allgemeine Darstellung für den Iterationsfehler $\vec{e}^{(k)} = \vec{\varphi} - \vec{\varphi}^{(k)}$ abgeleitet werden:

$$\vec{e}^{(k)} = M\vec{e}^{(k-1)} = M^k\vec{e}^{(0)} \quad \text{mit} \quad M = I - \tau C^{-1}A\,.$$

Die Matrix M bezeichnet man als Iterationsmatrix. Über diese wird der Fehler der $(k-1)$-ten Iterierten in den Fehler der k-ten Iterierten überführt.
Zur Lösung großer FE-Gleichungssysteme eignen sich die bisher vorgestellten Iterationsverfahren auf Grund ihrer hohen Iterationszahlen nicht. Eine wesentlich bessere Konvergenz bietet die Methode der konjugierten Gradienten. Dieses Verfahren zählt zu den instationären iterativen Gleichungslösern. Die Lösung des Gleichungssystems $A\vec{\varphi} = \vec{f}$ ist dabei äquivalent zur Minimierung eines quadratischen Funktionals und bildet den Ausgangspunkt für die nun folgenden Überlegungen. Die Matrix A sei symmetrisch und positiv definit. Dann ist $\vec{\varphi}$ die Lösung des Gleichungssystems $A\vec{\varphi} = \vec{f}$, wenn

$$J(\vec{\varphi}) = min_{\vec{v}\in\mathbb{R}^n} J(\vec{v}) \quad \text{mit} \quad J(\vec{v}) = \frac{1}{2}(A\vec{v}, \vec{v}) - (\vec{f}, \vec{v}) \tag{2.58}$$

gilt. Mit $(.,.)$ wird das Skalarprodukt zweier Vektoren bezeichnet. Zur Minimierung des Funktionals $J(\vec{v})$ kann ein Gradientenverfahren eingesetzt werden. In jedem Iterationsschritt wird die Richtung des Gradienten

$$\vec{r} = \mathrm{grad}\, J(\vec{v}) = A\vec{v} - \vec{f} \tag{2.59}$$

bestimmt. Da dieser in Richtung der lokal stärksten Zunahme des zu minimierenden Funktionals zeigt, entspricht $-\mathrm{grad}\, J(\vec{v})$ der Richtung der lokal stärksten Abnahme. Ausgehend von einer gegebenen Näherungslösung $\vec{\varphi}^{(k-1)}$ und der Suchrichtung $\vec{s}^{(k)} = \vec{r}^{(k)} + \beta^{(r)}\vec{s}^{(k-1)}$ als Linearkombination der negativen Gradientenrichtung $\vec{r}^{(k)} = \vec{f} - A\vec{\varphi}^{(k-1)}$ und der vorherigen Suchrichtung $\vec{s}^{(k-1)}$ wird die neue Näherung über

$$\vec{\varphi}^{(k)} = \vec{\varphi}^{(k-1)} + \alpha^{(k)}\vec{s}^{(k-1)}$$

berechnet. Der Relaxationsparameter $\alpha^{(k)}$ wird so gewählt, dass

$$J(\vec{\varphi}^{(k)}) = \underset{\alpha \in \mathbb{R}^1}{min}\, J(\vec{\varphi}^{(k-1)} + \alpha^{(k)} \vec{s}^{(k-1)})$$

gilt. Dies ist ein Minimierungsproblem einer reellen Variablen, dessen Lösung man mit

$$\alpha^{(k)} = \frac{(\vec{r}^{(k-1)}, \vec{r}^{(k-1)})}{(A\vec{s}^{(k-1)}, \vec{s}^{(k-1)})}$$

erhält. Der Parameter $\beta^{(k)}$ wird so gewählt, dass zwei aufeinanderfolgende Suchrichtungen $\vec{s}^{(k-1)}$ und $\vec{s}^{(k)}$ A-orthogonal sind, d. h. es gilt $(A\vec{s}^{(k)}, \vec{s}^{(k-1)}) = 0$. Da man auch von konjugierten Suchrichtungen spricht, wird diese Idee als das Verfahren der konjugierten Gradienten (CG-Verfahren – Conjugate Gradient Method) bezeichnet. Ein entsprechender Algorithmus lässt sich wie folgt darstellen:

Start: Wahl der Startnäherung $\vec{\varphi}^{(0)}$

$$\vec{r}^{(0)} = \vec{f} - A\vec{\varphi}^{(0)}$$
$$\vec{s}^{(0)} = \vec{r}^{(0)}$$

Iteration: $(k = 1, 2, \dots)$

$$\alpha^{(k)} = (\vec{r}^{(k-1)}, \vec{r}^{(k-1)})/(A\,\vec{s}^{(k-1)}, \vec{s}^{(k-1)})$$
$$\vec{\varphi}^{(k)} = \vec{\varphi}^{(k-1)} + \alpha^{(k)} \vec{s}^{(k-1)}$$
$$\vec{r}^{(k)} = \vec{r}^{(k-1)} - \alpha^{(k)} A\,\vec{s}^{(k-1)}$$
$$\text{Wenn} \quad (\vec{r}^{(k)}, \vec{r}^{(k)}) \leq \epsilon^2 (\vec{r}^{(0)}, \vec{r}^{(0)}) \quad \longrightarrow \quad \text{STOP}$$
$$\beta^{(k)} = (\vec{r}^{(k)}, \vec{r}^{(k)})/(\vec{r}^{(k-1)}, \vec{r}^{(k-1)})$$
$$\vec{s}^{(k)} = \vec{r}^{(k)} + \beta^{(k)} \vec{s}^{(k-1)}$$

Der Parameter ϵ gibt eine zu erreichende relative Genauigkeit für das Residuum an. Für Aussagen zur Konvergenz wird auf die Fachliteratur verwiesen (Barrett et al., 1994). Bei der Implementierung des CG-Verfahrens kann ebenfalls die Struktur der schwach besetzten Systemmatrix A für die Speicherung und die Matrix-Vektor-Operationen ausgenutzt werden. Der benötigte Speicherplatzbedarf ist proportional zur Anzahl an Unbekannten. Bei den notwendigen arithmetischen Operationen müssen nur die Nicht-Null-Elemente für die Berechnung herangezogen werden. Die Iterationszahlen liegen in der gleichen Größenordnung wie beim SOR-Verfahren mit optimal gewähltem Relaxationsparameter ω. Eine schnellere Konvergenz des CG-Verfahrens wird erreicht, wenn das Gleichungssystem besser konditioniert ist.
Deshalb soll nun die Methode der konjugierten Gradienten mit Vorkonditionierung kurz vorgestellt werden. Anstelle des zu lösenden Gleichungssystems $A\vec{\varphi} = \vec{f}$ wird das äquivalente

Gleichungssystem

$$\tilde{A}\tilde{\vec{\varphi}} = \tilde{\vec{f}} \tag{2.60}$$

mit $\tilde{A} = C^{-1}\,A$, $\tilde{\vec{\varphi}} = \vec{\varphi}$ und $\tilde{\vec{f}} = C^{-1}\vec{f}$ betrachtet, wobei C eine symmetrische positiv definite Vorkonditionierungsmatrix ist. Diese wird so gewählt, dass die Konditionszahl $\kappa(\tilde{A}) \ll \kappa(A)$ ist. Das PCG-Verfahren (Preconditioned Conjugate Gradient Method) kann unter Beibehaltung der Matrix A und der rechten Seite $\vec{f}$ wie folgt formuliert werden:

Start: Wahl der Startnäherung $\vec{\varphi}^{(0)}$

$$\begin{aligned} \vec{r}^{(0)} &= \vec{f} - A\vec{\varphi}^{(0)} \\ \text{Löse} \quad C\,\vec{\omega}^{(0)} &= \vec{r}^{(0)} \quad \text{nach} \quad \vec{\omega}^{(0)} \\ \vec{d}^{(0)} &= \vec{\omega}^{(0)} \end{aligned}$$

Iteration: $(k = 1, 2, \dots)$

$$\begin{aligned} \tilde{\alpha}^{(k)} &= (\vec{\omega}^{(k-1)}, \vec{r}^{(k-1)})/(A\,\vec{d}^{(k-1)}, \vec{d}^{(k-1)}) \\ \vec{\varphi}^{(k)} &= \vec{\varphi}^{(k-1)} + \tilde{\alpha}^{(k)}\vec{d}^{(k-1)} \\ \vec{r}^{(k)} &= \vec{r}^{(k-1)} - \tilde{\alpha}^{(k)}A\,\vec{d}^{(k-1)} \\ \text{Löse} \quad C\,\vec{\omega}^{(k)} &= \vec{r}^{(k)} \quad \text{nach} \quad \vec{\omega}^{(k)} \\ \text{Wenn}(\vec{\omega}^{(k)}, \vec{r}^{(k)}) &\leq \epsilon^2(\vec{\omega}^{(0)}, \vec{r}^{(0)}) \quad \longrightarrow \quad \text{STOP} \\ \tilde{\beta}^{(k)} &= (\vec{\omega}^{(k)}, \vec{r}^{(k)})/(\vec{\omega}^{(k-1)}, \vec{r}^{(k-1)}) \\ \vec{d}^{(k)} &= \vec{\omega}^{(k)} + \tilde{\beta}^{(k)}\vec{d}^{(k-1)} \end{aligned}$$

Die Konditionszahl $\kappa(\tilde{A}) = \kappa(C^{-1}A)$ sollte nahe bei Eins liegen, um eine schnelle Konvergenz des PCG-Verfahrens zu erreichen. Wenn die Vorkonditionierungsmatrix C gleich der Systemmatrix A ist, dann ist $\kappa = 1$. Das bringt aber keine Vorteile, da wieder ein Gleichungssystem mit der Systemmatrix zu lösen ist. Es müssen also Matrizen C gefunden werden, mit denen die Konditionszahl $\kappa(C^{-1}A)$ nahe bei Eins ist und für die $C\vec{\omega}^{(k)} = \vec{r}^{(k)}$ leicht zu lösen ist. Der Aufwand sollte proportional zur Anzahl der Unbekannten sein. Es bieten sich dafür die bereits beschriebenen direkten und stationären iterativen Gleichungslöser an. Dabei wird $\omega^{(k)}$ durch den Einsatz einer unvollständigen Cholesky-Zerlegung (ICC) der Systemmatrix A oder mit nur wenigen Iterationsschritten des Jacobi- oder SSOR-Verfahrens unter Verwendung von $C = A$ berechnet. Dies hat zum einen eine kleinere Konditionszahl und zum anderen einen vertretbaren numerischen Aufwand zur Folge.

2.3 Parallelisierung der Finite-Elemente-Methode

Heute wird eine Vielzahl an unterschiedlichen Parallelrechnern angeboten. Diese lassen sich grob nach ihrem Speicherschema in Rechner mit gemeinsamem und verteiltem Speicher klassifizieren. Weiterhin ist eine Unterteilung nach der zu Grunde liegenden Prozessorarchitektur möglich. Diese ersten Einteilungsmöglichkeiten legen bereits nahe, dass es unbedingt notwendig ist, für jeden Typ gesonderte Untersuchungen zu Parallelisierungsstrategien durchzuführen. Das Ziel dieser Arbeit ist es, für Linux-Cluster Möglichkeiten der Parallelisierung aufzuzeigen. Bei diesem Parallelrechnertyp handelt es sich um einen MIMD-Rechner mit verteiltem Speicher. Die Abkürzung MIMD steht für ”multiple instruction stream, multiple data stream”. Auf verschiedene Teilbereiche der Daten können zugleich verschiedene Instruktionen angewendet werden. Zudem verfügt jeder Rechner über seinen eigenen Speicher und ist über ein Netzwerk mit den anderen verbunden.
Im Folgenden soll speziell auf diesen Parallelrechnertyp eingegangen werden. In der Studienarbeit von Oeser (2002) hat sich herausgestellt, dass die Verwendung des nachrichtenorientierten Programmiermodells (engl. "message passing model") für einen Linux-Cluster günstig ist. In diesem wird die Parallelisierung über das Versenden und Empfangen von Daten und Nachrichten realisiert. Ein kurzer Einblick in die verschiedenen Parallelisierungsansätze ist neben Kriterien für die Bewertung des erreichten Parallelisierungsgrades in der Studienarbeit (Oeser, 2002) zu finden.

2.3.1 Bewertungskriterien

Als Maß für den erreichten Parallelisierungsgrad wird im Allgemeinen der Speedup verwendet. Er charakterisiert den Faktor des Geschwindigkeits- bzw. Laufzeitgewinns. Für eine feste Problemgröße lässt sich der parallele Speedup über

$$S_{par}(P) = \frac{t(1)}{t(P)} \tag{2.61}$$

berechnen, wobei P die Anzahl der verwendeten Rechner beschreibt. Die Bestimmung der Laufzeit t erfolgt für einen und P Prozessoren mit ein und demselben parallelen Algorithmus. Der erreichbare parallele Speedup für P Prozessoren ist im Optimalfall gleich P. Dies ist gleichbedeutend mit einer perfekten Skalierbarkeit des parallelen Algorithmus, d. h. beim Einsatz von P Prozessoren reduziert sich die Laufzeit auf $1/P$ der Laufzeit für einen Prozessor. Durch die Definition des parallelen Speedup ist es möglich, verschiedene Implementierungen eines Algorithmus für eine Rechnerarchitektur zu vergleichen. Allerdings sind große Speedup-Werte noch kein Hinweis auf eine gute Parallelisierung, da die Speedup-Zahlen für die Implementierung eines Algorithmus auf verschiedenen Architekturen unterschiedlich sein können. Es

muss zusätzlich noch ein Vergleich der Laufzeiten vorgenommen werden. Da es immer einzelne Programmteile gibt, die rein sequentiell abgearbeitet werden müssen, wird sich der erreichte Speedup nur an P annähern. Im AMDAHLschen Gesetz wird daher der Speedup unter Beachtung des sequentiellen (A_s) und parallelen Anteils (A_p) am Algorithmus wie folgt berechnet:

$$S = \frac{1}{A_s + \frac{A_p}{P}} \,. \tag{2.62}$$

Bei der Implementierung sollte daher auf einen geringen Anteil an rein sequentiell abzuarbeitenden Programmcode geachtet werden.

2.3.2 Ansatzpunkte für die Parallelisierung

Die Analyse der einzelnen Teilschritte innerhalb der Finite-Elemente-Diskretisierung zeigt die Ansatzpunkte für eine Parallelisierung auf. Das in Huber (1997) eingeführte Schichtenmodell für die verschiedenen Parallelisierungsgrade legt die Ausnutzung der Datenparallelität nahe, um einen möglichst hohen Speedup zu erreichen. In dieser Ebene wird mittels Datenstrukturen parallelisiert, d.h. es wird zur gleichen Zeit auf verschiedene Teile der Daten der gleiche Algorithmus angewendet. Um herauszufinden, welche Programmablaufpunkte besonders für eine Parallelisierung geeignet sind, müssen diese gezielt untersucht werden.

Gittergenerierung

Parallele Algorithmen zur Generierung einer Vernetzung $\mathcal{T}_h$ existieren bereits, sind aber nicht frei verfügbar. Daher wird im Folgenden von einer bereits vorhandenen Verteilung der Datenstrukturen auf die einzelnen Prozessoren ausgegangen. Es müssen also die Elementzusammenhangstabelle und die Knotentabelle möglichst in zusammenhängenden Blöcken auf die Prozessoren verteilt werden. Eine Einführung in die verschiedenen Algorithmen zur Zerlegung von Gittern auf eine Anzahl von Prozessoren ist in Saad (1996) enthalten.

Aufbau des linearen Gleichungssystems

Nach erfolgreicher Verteilung der Datenstrukturen auf die einzelnen Rechner muss das lineare Gleichungssystem aufgebaut werden. Wie bereits dargestellt, kann dies elementweise erfolgen. Damit ist eine parallele Abarbeitung möglich. Jedem der P Prozessoren ist eine Anzahl von Elementen $T^{(r)}$ aus der Vernetzung $\mathcal{T}_h$ des Gebietes Ω zugeordnet. Im Idealfall entspricht die Menge der Elemente eines Prozessors genau dem P-ten Teil von R_h. Die Verteilung der Elemente in zusammenhängenden Blöcken führt zu einem geringen Kommunikationsaufwand zwischen den Prozessoren beim Aufbau des Gleichungssystems. Die Einträge in die Steifigkeitsmatrix an den Grenzen der einzelnen Blöcke müssen durch Versenden und Empfangen von Nachrichten auf den richtigen Prozessor verteilt werden. Damit liegt das Gleichungssystem zei-

lenweise aufgeteilt vor. Bei optimaler Aufteilung der Elemente auf die einzelnen Prozessoren sollte der Aufbau des Gleichungssystems nahezu linear skalieren, d. h. mit einer Verdopplung der Prozessoranzahl halbiert sich die benötigte Zeit.

Lösung des linearen Gleichungssystems

Beim Übergang zum nächsten Schritt, dem Lösen des linearen Gleichungssystems, ist kaum Kommunikation notwendig, da die Zeilen des Gleichungssystems bereits in zusammenhängenden Blöcken auf die Prozessoren verteilt sind. Der Parallelisierungsaufwand für die vorgestellten Gleichungslöser und Vorkonditionierer unterscheidet sich teilweise erheblich. In den Veröffentlichungen von Saad (1996), Demmel et al. (1993), Van der Vorst (1993) und Balay et al. (1997) finden sich detaillierte Ausführungen zu dieser Problemstellung. Bei der Lösung eines linearen Gleichungssystems mittels des PCG-Verfahrens verursachen

- die Matrix-Vektor-Produkte,
- die Vektor-Vektor-Produkte und
- die Vorkonditionierung

den größten numerischen Aufwand. Diese vier Punkte gilt es geeignet zu parallelisieren. Um Leerlaufzeiten zu vermeiden, sollte die Kommunikation durch notwendige Berechnungen überlagert werden.

Die Zeilen der Matrix A_h und Elemente des Vektors $\vec{f_h}$ sind in Blöcken auf die Prozessoren verteilt. Entsprechend der schematischen Darstellung für die Berechnung des Matrix-Vektor-Produktes in Abbildung 2.8 müssen die von den anderen Prozessoren benötigten Vektorelemente versendet werden. Prozessor 2 muss demnach seine Elemente (c und d) an die Prozes-

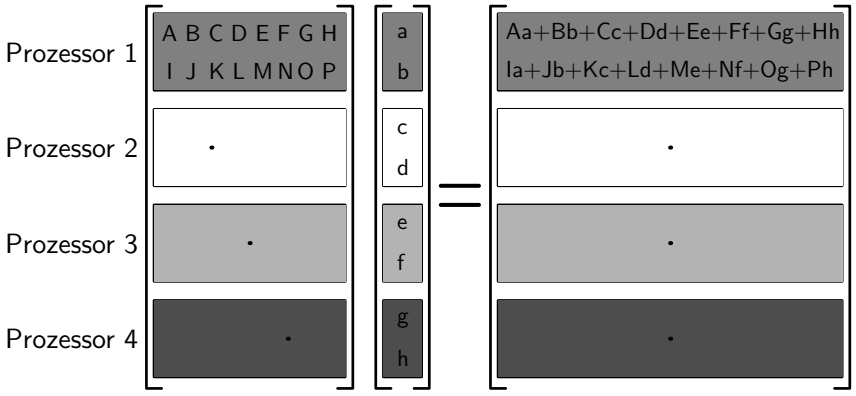


Abbildung 2.8: Berechnung des Matrix-Vektor-Produktes

soren 1, 3 und 4 verschicken. Analog übertragen die anderen Prozessoren ihre Elemente. Nach

dem Erhalt der Vektorelemente kann jeder Prozessor die Multiplikationen und Additionen für seinen Block durchführen. Am Ende liegen die Elemente des neu berechneten Vektors auf die gleiche Weise wie der Ausgangsvektor verteilt vor. Die Kommunikationszeit kann bei dünn besetzten Matrizen verringert werden, da nicht alle Vektorelemente zum Bilden des Matrix-Vektor-Produktes notwendig sind. Der damit verbundene Mehraufwand beim Ermitteln der zu übertragenden Elemente wiegt den Zeitvorteil beim Versenden der Elemente möglicherweise wieder auf. Im Allgemeinen ist aber ein Laufzeitgewinn mit zunehmender Größe des Gleichungssystems zu erwarten.

Die Berechnung des Vektor-Vektor-Produktes (inneres Produkt) ist einfacher zu parallelisieren. Wie man in Abbildung 2.9 nachvollziehen kann, berechnet jeder Prozessor das innere Produkt für alle seine Elemente und verschickt das Ergebnis an die anderen Prozessoren. Mit den lokalen Vektor-Vektor-Produkten aller anderen Prozessoren ist es für jeden Prozessor möglich, das vollständige Vektor-Vektor-Produkt zu bestimmen. Bei Iterationsverfahren vom Typ CG

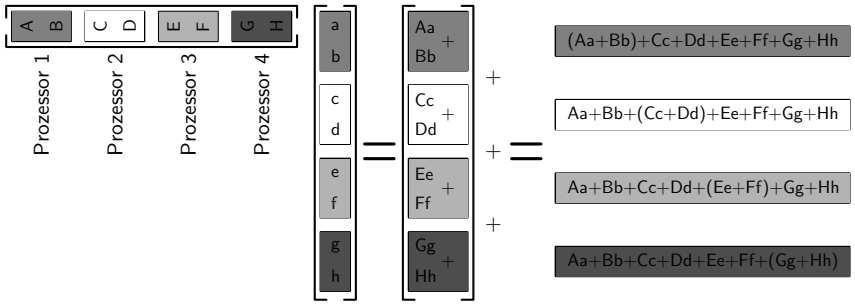


Abbildung 2.9: Berechnung des Vektor-Vektor-Produktes

fungiert die Bildung des Vektor-Vektor-Produktes als Synchronisationspunkt, da alle weiteren Berechnungen erst danach starten können. Aus diesem Grund existieren abgewandelte CG-Schemata für Parallelrechner. Bei diesen wird durch Umsortieren der einzelnen CG-Schritte die Kommunikationszeit mit notwendiger Berechnung überlagert. Die Vor- und Nachteile für drei verschiedene Umsortierungen des CG-Algorithmus werden in Demmel et al. (1993) näher untersucht.

Die Parallelisierung der unterschiedlichen Vorkonditionierer ist schwieriger. Da die meisten Vorkonditionierer nicht im Hinblick auf eine Verwendung mit Parallelrechnern entwickelt wurden, kann es zu Problemen bei der Umsetzung kommen. Relativ einfach zu implementieren ist der Block-Jacobi-Vorkonditionierer. Bei diesem Typ wird auf den lokalen Teil des Systemmatrixblockes $(A_h)_P$ ein sequentieller Vorkonditionierer angewendet. Zu den möglichen sequentiellen Vorkonditionierern zählen das SSOR-Verfahren, die unvollständige LU-Zerlegung (ILU) sowie die unvollständige Cholesky-Zerlegung (ICC).

Einsammeln der Lösung

Die auf die einzelnen Rechner verteilte Lösung des linearen Gleichungssystems muss im letzten Schritt eingesammelt werden. Zu diesem Zweck wird von allen Prozessoren der Teil der Lösung, der in ihrem lokalen Speicher vorhanden ist, an Prozessor 1 versendet. Dieser fügt die empfangenen Elemente zu den eigenen hinzu. Die damit auf Prozessor 1 vorhandene Kopie der Lösung kann für weitere Berechnungen genutzt werden.

2.3.3 Wahl der optimalen Prozessoranzahl

Nachdem alle notwendigen Schritte innerhalb der FE-Diskretisierung parallelisiert sind, stellt sich die Frage, welche Teilschritte am meisten von einer Parallelisierung profitieren. Für eine hohe Effizienz bei der Nutzung des verteilten Speichers ist es notwendig, vom Programmstart bis zum Ende alle Daten gleichmäßig und effektiv zu verteilen. Dabei sollte darauf geachtet werden, dass nach Möglichkeit keine Daten mehrfach zwischen den Prozessoren ausgetauscht werden, da dies zu einem erhöhten Kommunikationsaufwand führt und der Speedup sinkt. Bei zunehmender Rechenkapazität durch eine größere Anzahl an Prozessoren wird die Lösung des linearen Gleichungssystems einen erheblichen Laufzeitgewinn verzeichnen. Wenn die lokale Submatrix mit größer werdender Prozessoranzahl immer kleiner wird, überwiegt jedoch die Kommunikationszeit und der Laufzeitgewinn sinkt im schlimmsten Fall wieder ab.

3 Implementierung

Die Komplexität bei der Diskretisierung mit der Methode der Finiten Elemente erfordert ein Umdenken in der bisherige Strategie bei der Entwicklung von Modellierungsprogrammen in der Geophysik. Es ist nicht mehr möglich, alle einzelnen Schritte selbst programmiertechnisch umzusetzen, sondern man sollte auf bereits vorhandene Algorithmen und Softwarepakete aufbauen. Das hochgesteckte Ziel eines parallelisierten Finite-Elemente-Programmes ist nur dann mit vertretbarem Aufwand zu erreichen, wenn auf vorhandene Ressourcen zurückgegriffen werden kann.

3.1 Gittergenerierung

Beim Verfahren der Gleichstromgeoelektrik bildet die Vernetzung des Modellgebietes mit enthaltenen Leitfähigkeitsstrukturen den Ausgangspunkt für die FEM-Modellrechnungen. Bei der Generierung einer solchen Vernetzung können verschiedene Elementtypen benutzt werden. Mit dem Softwarepaket TetGen (Si, 2003) können unstrukturierte Tetraedergitter erstellt werden. Über die von Carsten Rücker entwickelte Programmbibliothek *myfemlib*[2] stehen verschiedene Methoden zur Erzeugung einer Eingabedatei für TetGen zur Verfügung. Über diesen Zugang ist es möglich, durch die Nutzung von vorhandenen Ressourcen, die relativ schwierige Gittergenerierung zu vereinfachen. Da sich die Programmbibliothek *myfemlib* noch in der Entwicklungsphase befindet, kann in dieser Arbeit keine genauere Beschreibung der darin vorhandenen Methoden erfolgen. Eine umfangreiche Dokumentation für *myfemlib* wird derzeit vom Autor erstellt. In der Beschreibung zu TetGen sind alle Optionen und Dateiformate ausführlich erläutert. Die dieser Arbeit beiliegende CD enthält Beispieldateien für TetGen inklusive aller notwendigen Optionen für das Generieren der unstrukturierten Tetraedergitter.

3.2 Das Finite-Elemente-Programm *dcfempar*

Die frei erhältliche FEM-Bibliothek libMesh (Kirk et al., 2004) dient als Grundlage für das in dieser Arbeit erstellte, parallelisierte Finite-Elemente-Programm *dcfempar*. In libMesh wird das nachrichtenorientierte Programmiermodell umgesetzt, sodass eine möglichst hohe Portabilität gewährleistet wird. Die Basis bildet eine entsprechende Implementierung dieses Programmiermodells, wie zum Beispiel MPICH (Gropp und Lusk, 1996). Mit den darin zur Verfügung stehenden Funktionen werden die benötigten Methoden für den parallelen Rechenbetrieb bereitgestellt. Zur Lösung des entstehenden linearen Gleichungssystems wird die PETSc-(Portable Extensible Toolkit for Scientific Computation) Bibliothek (Balay et al., 2001) verwendet. Die

[2] persönliche Kommunikation

gesamte Umsetzung wird in der Programmiersprache C++ realisiert. Durch objektorientiertes Programmieren (Datenabstraktion, Datenkapselung und Polymorphie) und die Verwendung von C++ ist es möglich, einen portablen Programmcode zu erstellen, der einfach zu warten und zu erweitern ist.
Die Nachvollziehbarkeit der einzelnen Diskretisierungsschritte wird durch die vollständige Dokumentation des Quelltextes erleichtert. Auf der beiliegenden CD befindet sich der gesamte Quelltext des Programmes *dcfempar*. Mehr über den Inhalt der CD und eine Anleitung zum Übersetzen des Programmes in Maschinencode ist im Abschnitt B.1 zu finden.

Einlesen des Gitters

Im ersten Schritt der Umsetzung des gleichstromgeoelektrischen Vorwärtsproblems war es notwendig, die libMesh-Bibliothek um Methoden zum Einlesen des Gitters und der Metadaten (Leitfähigkeit und Elektrodenposition) zu erweitern. Mittlerweile ist die entsprechende Funktionalität offiziell in libMesh integriert. Nach dem Einlesen des Gitters und einigen vorab notwendigen Schritten (Auswertung von Programmparametern, Lokation der Elektrodenpositionen) kann das lineare Gleichungssystem parallel aufgebaut und gelöst werden.

Assemblierung des linearen Gleichungssystems

Im Folgenden soll exemplarisch kurz die programmiertechnische Umsetzung des Matrixaufbaus erläutert werden. Der Einbau der Randbedingungen und die vorhandenen Kommentare wurden entfernt, um die Übersicht zu wahren.
Der Zugriff auf die Elemente der Vernetzung ist nur dann möglich, wenn eine Referenz auf das `mesh`-Objekt vorhanden ist (Zeile 1 des Quelltextauszuges). Über den folgenden Methodenaufruf wird die Dimension des Gitters in der Variablen `dim` gespeichert. Im nächsten Schritt wird der Elementtyp für die erste Variable des Systems, das Potential φ, bestimmt. In dieser Arbeit werden lineare Tetraederelemente verwendet. Intern wird dieser Typ in libMesh als `TET4`-Element bezeichnet. Dem damit erzeugten Finite-Elemente-Objekt `fe` wird über das `qrule`-Objekt die verwendete numerische Quadraturformel übermittelt (Zeilen 4, 5 und 6). Im Programm wird die GAUSSsche Quadraturformel 5. Ordnung eingesetzt.

```
1  const MeshDC& mesh = es.get_meshdc();
2  const unsigned int dim = mesh.mesh_dimension();
3  FEType fe_type = es("DCGeo").get_dof_map().variable_type(0);
4  AutoPtr<FEBase> fe (FEBase::build(dim, fe_type));
5  QGauss qrule (dim, FIFTH);
6  fe->attach_quadrature_rule (&qrule);
7
8  const std::vector<Real>& JxW=fe->get_JxW();
```

```
 9  const std::vector<std::vector<Real> >& phi=fe->get_phi();
10  const std::vector<std::vector<RealGradient>>& dphi=fe->get_dphi();
11  const DofMap& dof_map = es("DCGeo").get_dof_map();
12
13  DenseMatrix<Number> Ke;
14  std::vector<unsigned int> dof_indices;
15  std::vector<Number> ElemData;
16  double sigma=0.0;
```

Ist dieser Punkt erreicht, müssen Referenzen auf elementspezifische Daten erzeugt werden (Zeilen 8 bis 11). Diese werden während der Assemblierung des Gleichungssystems benötigt. Dazu zählt die Transformationsmatrix J (vgl. Gl. (2.45)) vom Referenzelement auf das eigentliche Element der Vernetzung (`JxW`), der Wert der Ansatz- und Testfunktion (`phi`) und der Gradient dieser (`dphi`) an den Quadraturpunkten. Über das Objekt `dof_map` erfolgt die Zuordnung der lokalen Knotennummern auf die globalen. Anschließend (Zeilen 13 bis 16) wird Speicher für die Elementsteifigkeitsmatrix `Ke`, den Indexvektor `dof_indices`, den Vektor `ElemData` und die Leitfähigkeit `sigma` reserviert.
Im nächsten Abschnitt werden von jedem Prozessor sogenannte Iteratoren (`el` und `end_el`) erzeugt, mit denen es möglich ist, durch die lokal auf einem Prozessor vorhandenen Elemente zu iterieren (Zeilen 17 und 18). In einer Schleife wird für jedes lokale Elemente die Elementsteifigkeitsmatrix `Ke` gebildet. Dazu müssen zuerst im Vektor `dof_indices` die Position der einzelnen Einträge in der Steifigkeitsmatrix abgelegt und Speicher für `Ke` bereitgestellt sowie alle Daten für das aktuelle Element initialisiert werden (Zeilen 22 bis 24). Jedem Element wird während der Gittergenerierung der spezifische elektrische Widerstand, entweder direkt oder über eine Zahl codiert, zugeordnet. Wird beim Programmstart eine Datei mit der entsprechenden Zuordnung übergeben, berechnet sich die elektrische Leitfähigkeit über diese Zahlenwerte, ansonsten entspricht der dem Element zugeordnete Wert bereits dem spezifischen elektrischen Widerstand (Quelltextzeilen 26 bis 32).

```
17  const_local_elem_iterator           el (mesh.elements_begin());
18  const const_local_elem_iterator end_el (mesh.elements_end());
19
20  for ( ; el != end_el; ++el){
21    const Elem* elem = *el;
22    dof_map.dof_indices (elem, dof_indices);
23    fe->reinit (elem);
24    Ke.resize (dof_indices.size(), dof_indices.size());
25
26    if (mesh.data.has_data (elem)) {
```

```
27      ElemData = mesh.data.get_data (elem);
28      if (DCConfig::getAttributeMapFile() != "dummy") {
29        sigma = 1. / DCConfig::getAttributeMap(ElemData[0]);
30      } else {
31        sigma = 1. / ElemData[0]; }
32    }
33    for (unsigned int qp=0; qp<qrule.n_points(); qp++){
34      for (unsigned int i=0; i<phi.size(); i++)
35        for (unsigned int j=0; j<phi.size(); j++)
36          Ke(i,j) += sigma * JxW[qp] * (dphi[i][qp]*dphi[j][qp]);
37    }
38    es("DCGeo").matrix->add_matrix (Ke, dof_indices);
39  }
```

Von da an sind alle Variablen mit den entsprechenden Werten für das aktuelle Element belegt. In einer Schleife über die Quadraturpunkte `qp` (Zeilen 33 bis 37) kann die numerische Integration ausgeführt werden. Im Anschluss werden die Einträge der Elementsteifigkeitsmatrix an ihrer korrekten Position in der Steifigkeitsmatrix eingefügt (Zeile 38).
Analog erfolgt die Assemblierung der rechten Seite und der Einbau der Randbedingungen. Im Anschluss daran wird das lineare Gleichungssystem gelöst.

Nachbearbeitung

Nach dem Lösen des linearen Gleichungssystems sind einige Schritte zur Bearbeitung der Ergebnisse (Potentialwerte an den Knoten) notwendig. Im Programm *dcfempar* stehen zur Simulation von gleichstromgeoelektrischen Messungen zwei Möglichkeiten zur Verfügung. Die Programmoption `-M s` führt eine inverse Schlumberger-Sondierung, ausgehend von der Mitte der Elektrodenkette, durch. Über die Option `-M t` wird eine Pol-Pol-Messung für alle möglichen Elektrodenkombinationen simuliert. Das Ergebnis dieser Simulation wird in der über die Option `-o` angegebenen Datei gespeichert. Das Format der Ausgabedatei wurde so gewählt, dass die von Thomas Günther[3] implementierten Visualisierungswerkzeuge zur geeigneten Darstellung der Ergebnisse genutzt werden können. Mehr zu diesen MATLAB-Routinen[4] ist im Abschnitt B.1 zu finden.

[3] persönliche Kommunikation
[4] Copyright 1984-2004, The MathWorks, Inc.

3.3 Modulkonzept

Das Konzept für die Entwicklung eines parallelisierten Finite-Elemente-Programmes durch Verwendung von frei verfügbaren oder in der Arbeitsgruppe vorhandenen Softwaremodulen hat sich als effektiv herausgestellt. So war es innerhalb kurzer Zeit möglich, weitere Funktionalität in die Bibliothek libMesh zu integrieren. Auf den Erfahrungen von Carsten Rücker mit der Generierung von unstrukturierten Tetraedergittern aufbauend war es schnell möglich, eine erste funktionierende Version des Programmes *dcfempar* zu implementieren. Ausgehend davon ist die libMesh-Bibliothek sukzessiv um weitere Funktionalität erweitert worden, um das Programm so effizient wie möglich zu gestalten. Im weiteren Verlauf der Arbeit kam es zu Synergieeffekten mit den Programmentwicklungen von Thomas Günther. Ohne dieses modulare Konzept würde die Entwicklungsarbeit den zeitlichen Rahmen einer solchen Arbeit übersteigen. Durch die ausführliche Dokumentation der Bibliothek libMesh empfiehlt sich diese für weitere Arbeiten auf dem Gebiet der Vorwärtsmodellierung mit der Finite-Elemente-Methode.

4 Untersuchungen

Bei der Entwicklung numerischer Simulationssoftware stehen die zwei Schwerpunkte Genauigkeit und Optimierungsmöglichkeiten im Vordergrund. Daneben ist bei parallelisierten Programmen das Laufzeitverhalten mit zunehmender Anzahl an Prozessoren interessant, also wie das Programm mit erhöhter Prozessoranzahl skaliert. In den folgenden Abschnitten sollen die Genauigkeit des Modellierungsprogrammes überprüft und eine optimale Kombination aus Gleichungslöser und Vorkonditionierer gefunden werden. Im Anschluss daran wird in Skalierungstests der Laufzeitgewinn für verschiedene Modelle untersucht.

4.1 Genauigkeit des Modellierungsprogrammes *dcfempar*

Aussagen über die Genauigkeit der FEM-Modellrechnung lassen sich durch den Vergleich mit analytischen Lösungen für einfache Modelle treffen. Zu diesen zählen der homogene Halbraum, der geschichtete Halbraum und die vertikale Platte im homogenen Halbraum. Weiterhin werden Ergebnisse anderer numerischer Modellierungsprogramme zum Vergleich herangezogen.
Auf die Genauigkeit haben eine Vielzahl von Faktoren Einfluss. Eine systematische Untersuchung aller ist im Rahmen dieser Arbeit nicht möglich. Deshalb soll im Folgenden der Einfluss der Randbedingungen, der TetGen-Parameter und das Abbruchkriterium des Gleichungslösers genauer betrachtet werden.

- Im Programm *dcfempar* sind homogene DIRICHLETsche Randbedingungen und homogene gemischte Randbedingungen implementiert.

- Über den TetGen-Parameter `-q` wird eine zu erreichende Qualität der Vernetzung gefordert. Das Verhältnis aus dem Radius der kleinsten das Tetraeder enthaltenden Kugel und der Länge der kürzesten Kante muss für alle Tetraeder der Triangulierung gleich oder kleiner dem angegebenen Wert sein. Auf den Grad der Verfeinerung um die Elektrodenposition hat der Abstand dz eines Verfeinerungspunktes Einfluss. Die Elektrodenposition und der Verfeinerungspunkt werden als Knoten der Vernetzung festgelegt.

- Zur Bestimmung der Konvergenz des iterativen Gleichungslösers wird in PETSc ein relatives Maß verwendet. Wenn das Residuum der k-ten Iterierten $\|\vec{r}^{(k)}\|_2$ eine gegebene Schranke unterschreitet

$$\|\vec{r}^{(k)}\|_2 = \|\vec{f}_h - A_h\vec{\varphi}_h^{(k)}\|_2 < rtol * \|\vec{f}_h - A_h\vec{\varphi}_h^{(0)}\|_2 = rtol * \|\vec{r}^{(0)}\|_2 \,, \qquad (4.1)$$

  dann ist Konvergenz im Rahmen der über $rtol$ und des Residuum der 0-ten Iterierten festgelegten Grenze erreicht. Wenn nicht anders angegeben, wird für alle folgenden Untersuchungen $rtol$ auf 10^{-12} festgelegt.

Der relative Fehler der numerischen Modellrechnung

$$\epsilon = \frac{\varrho_{snumerisch} - \varrho_{sanalytisch}}{\varrho_{sanalytisch}} * 100\% \tag{4.2}$$

wird zur Abschätzung der erreichten Genauigkeit verwendet.

4.1.1 Der homogene Halbraum

Das Modell des homogenen Halbraumes erstreckt sich in x- und y-Richtung von $-290\,\mathrm{m}$ bis $290\,\mathrm{m}$ und in z-Richtung von $0\,\mathrm{m}$ bis $-406\,\mathrm{m}$. Für den spezifischen elektrischen Widerstand wurde $100\,\Omega \cdot \mathrm{m}$ angenommen. In Abbildung A.1 (Anhang) ist das unstrukturierte Tetraedergitter für den homogenen Halbraum dargestellt. Es wurde eine Pol-Pol-Sondierung simuliert. Der Einspeisepol befindet sich bei $(0, 0, 0)$. Entlang eines Profiles auf der y-Achse ist zwischen $-40\,\mathrm{m}$ und $40\,\mathrm{m}$ aller $1\,\mathrm{m}$ an den Potentialelektroden der Wert für den scheinbaren spezifischen Widerstand ϱ_s bestimmt worden.
Das Modell beinhaltet keine Leitfähigkeitskontraste. Damit hängt im Wesentlichen die Genauigkeit von der Struktur des Tetraedergitters ab. Dadurch eignet es sich für eine Untersuchung zum Einfluss der TetGen-Parameter `-q` und dz.
In den Grafiken der Abbildung 4.1 sind für Qualitätsparameter zwischen 1.1 und 2.0 bei festem Abstand des Verfeinerungspunktes von der Elektrodenposition ($dz = 0.10\,\mathrm{m}$) die relativen Fehler (Gleichung (4.2)) über dem Abstand vom Einspeisepol dargestellt. Die Diagramme in Abbildung 4.2 zeigen für einen festen Qualitätsparameter (`-q 1.2`) bei variierendem dz zwischen $0.01\,\mathrm{m}$ und $0.50\,\mathrm{m}$ die Ergebnisse für den relativen Fehler über dem Abstand vom Einspeisepol. Rot gekennzeichnet sind die Modellrechnungen mit gemischten Randbedingungen und Blau die für DIRICHLETsche Randbedingungen. Mit Pluszeichen werden die relativen Fehler für das Programm *dcfempar* und mit Kreisen die von *ggfem3d*[5] hervorgehoben.
In allen Abbildungen sind zum Rand hin steigende relative Fehler für homogene DIRICHLETsche Randbedingungen festzustellen. Dieser Rand ist nicht wie angenommen im Unendlichen ($\lim_{|\vec{r}|\to\infty} \varphi = 0$), womit die erzwungene Abnahme des Potentials auf Null ($\varphi = 0$) Fehler verursacht. Dagegen sind die homogenen gemischten Randbedingungen für dieses Modell exakt, was sich in geringen relativen Fehlern äußert. Diese liegen unterhalb von $5\,\%$ und nehmen mit dem Wert für den Qualitätsparameter `-q` ab. Ein Resteinfluss bleibt durch das grobe Gitter am Rand.
Bei größer werdenden Abständen dz des Verfeinerungspunktes von der Elektrodenposition erhöhen sich die relativen Fehler in der Nähe vom Einspeisepol, was durch einen geringeren Verfeinerungsgrad um die Elektrodenposition erklärt werden kann.

[5] Carsten Rücker, persönliche Kommunikation, FEM-Modellrechnung, Totalpotential

Vergleicht man die Resultate für beide Programme, so zeigt sich eine gute Übereinstimmung, was auch bedingt durch die Verwendung derselben Tetraedergitter erwartet wird. Die Unterschiede in den relativen Fehlern für DIRICHLETschen Randbedingungen bedürfen allerdings noch einer genaueren Untersuchung.
Wie aus den Abbildungen 4.1 und 4.2 zu entnehmen ist, beeinflussen beide TetGen-Parameter die Genauigkeit in unterschiedlicher Weise. Während die Genauigkeit mit kleiner werdendem Qualitätsparameter über das gesamte Gebiet ansteigt, bewirken kleine Abstände des Verfeinerungspunktes eine Verbesserung der Genauigkeit in der Nähe des Einspeisepols. Der Typ der Randbedingung beeinflusst im Wesentlichen die Genauigkeit der Modellierung zum Rand hin. Gute Ergebnisse bei einer annehmbaren Gittergröße zeigt die Kombination aus `-q 1.2` und $dz = 0.10\,\mathrm{m}$ mit homogenen gemischten Randbedingungen.

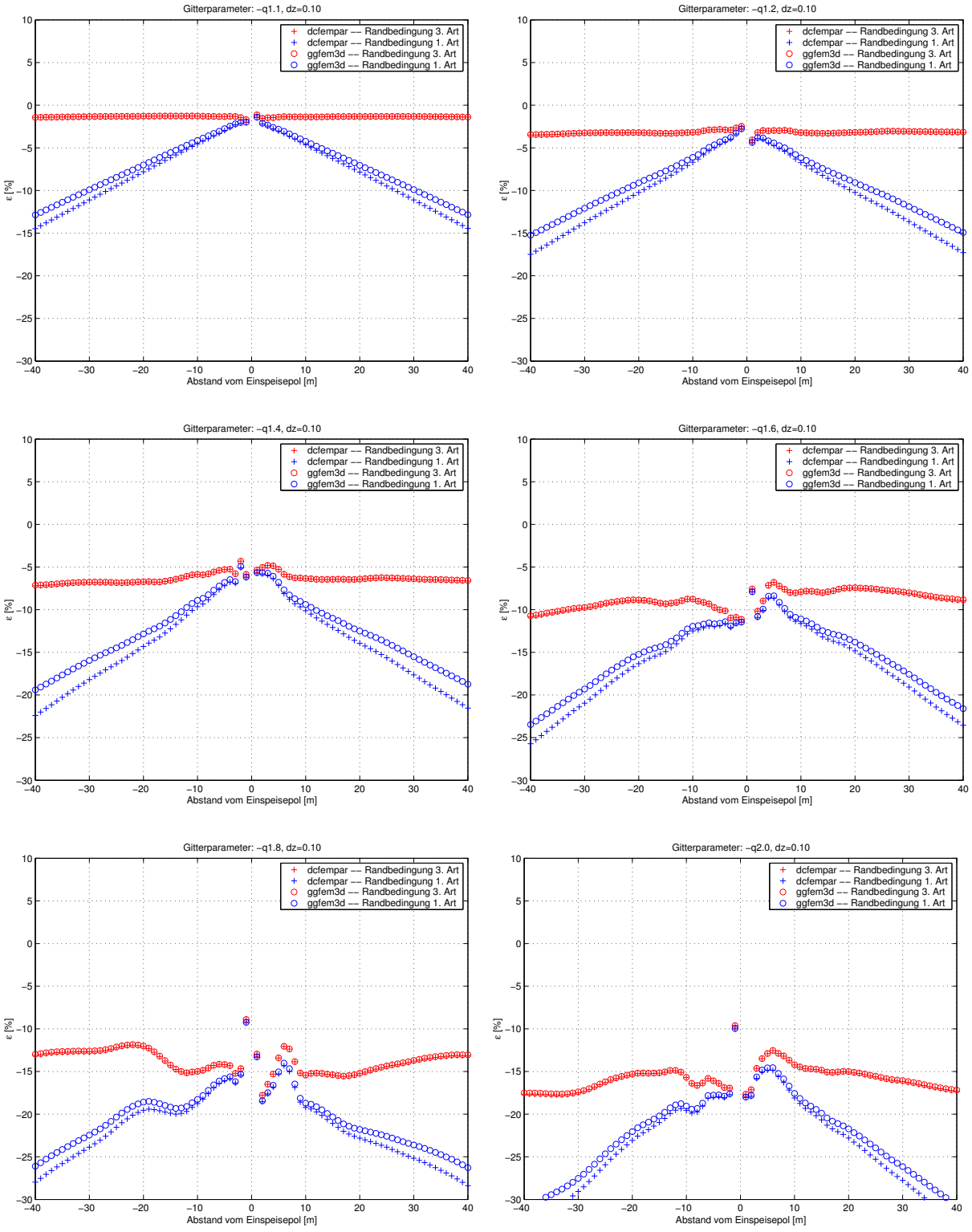


Abbildung 4.1: Relativer Fehler der Vorwärtsrechnung für variierenden Qualitätsparameter `-q` und festen Abstand des Verfeinerungspunktes ($dz = 0.10\,\mathrm{m}$), Pol-Pol-Sondierung

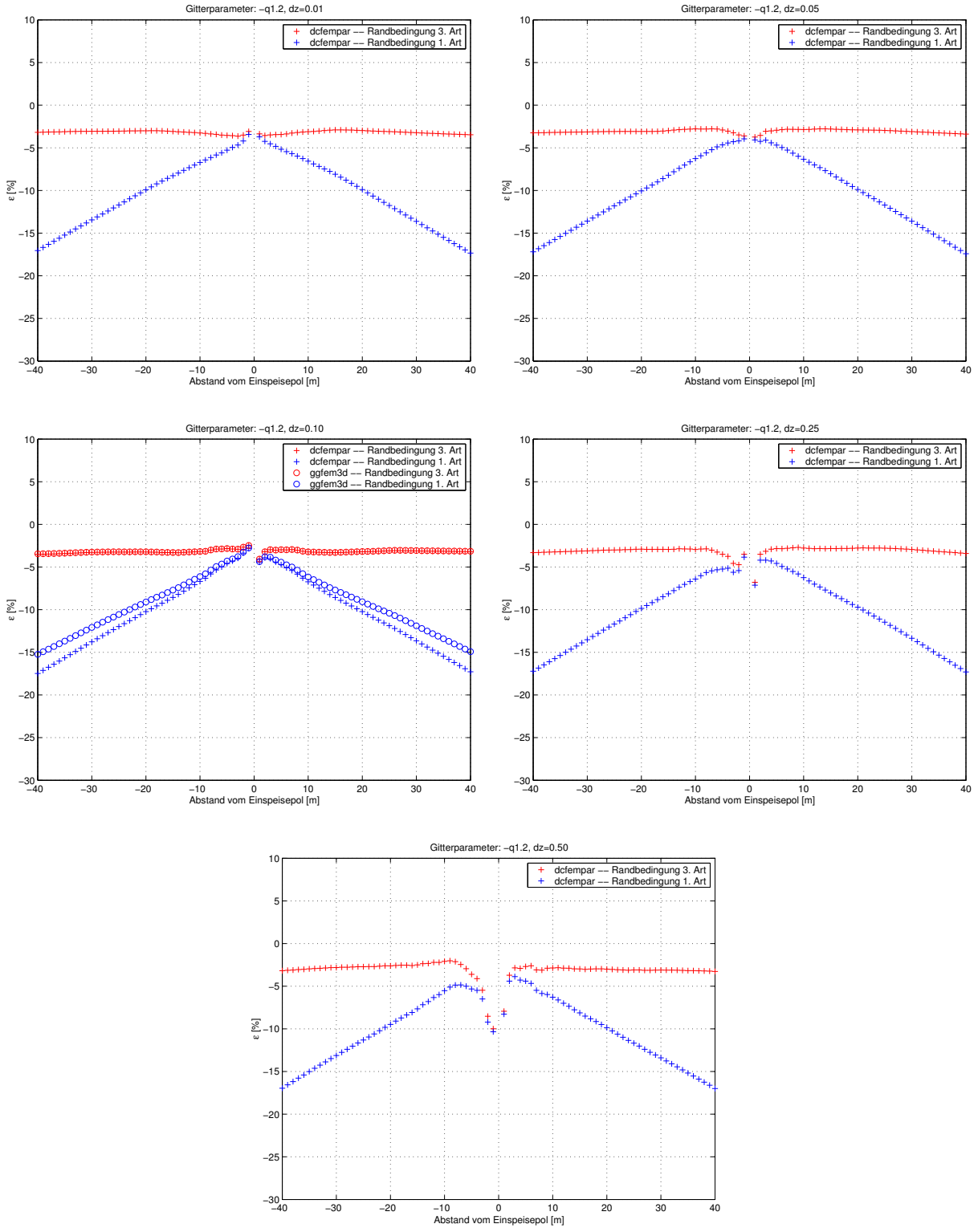


Abbildung 4.2: Relativer Fehler der Vorwärtsrechnung für einen festen Qualitätsparameter (`-q 1.2`) und variierenden Abstand des Verfeinerungspunktes dz, Pol-Pol-Sondierung

4.1.2 Der geschichtete Halbraum

Als Modell für den geschichteten Halbraum wurde der Dreischichtfall gewählt. Die Ausdehnungen in x-, y- und z-Richtung sind identisch mit dem Modell des homogenen Halbraums. Die Mächtigkeit der ersten Schicht beträgt $10\,\mathrm{m}$ und der spezifische elektrische Widerstand $10\,\Omega \cdot \mathrm{m}$. Die nächste Schicht hat $100\,\Omega \cdot \mathrm{m}$ bei einer Mächtigkeit von $30\,\mathrm{m}$. Für die darunterliegende dritte Schicht wurde ein spezifischer elektrischer Widerstand von $1\,\Omega \cdot \mathrm{m}$ angenommen. Eine Darstellung des Tetraedergitters für dieses Modell ist in Abbildung A.2 zu finden.
In der vorangegangenen Untersuchung wurde festgestellt, dass mit `-q 1.2` und $dz = 0.10\,\mathrm{m}$ eine gute Genauigkeit bei homogenen gemischten Randbedingungen erreicht wird. Bei der Generierung der Vernetzung für den Dreischichtfall fanden diese Verwendung. Die relativen Fehler für eine inverse Schlumbergersondierung mit den Einspeisepunkten $(0, -1\,\mathrm{m}, 0)$ und $(0, 1\,\mathrm{m}, 0)$ sind über dem halben Separationsabstand in Abbildung 4.3 dargestellt. Die analytische Lösung wurde mit einem Programm von Christoph Schwarzbach[6] berechnet. Durch rote Kreuze sind die relativen Fehler für das Programm *dcfempar* und durch blaue Kreise die von *ggfem3d* her-

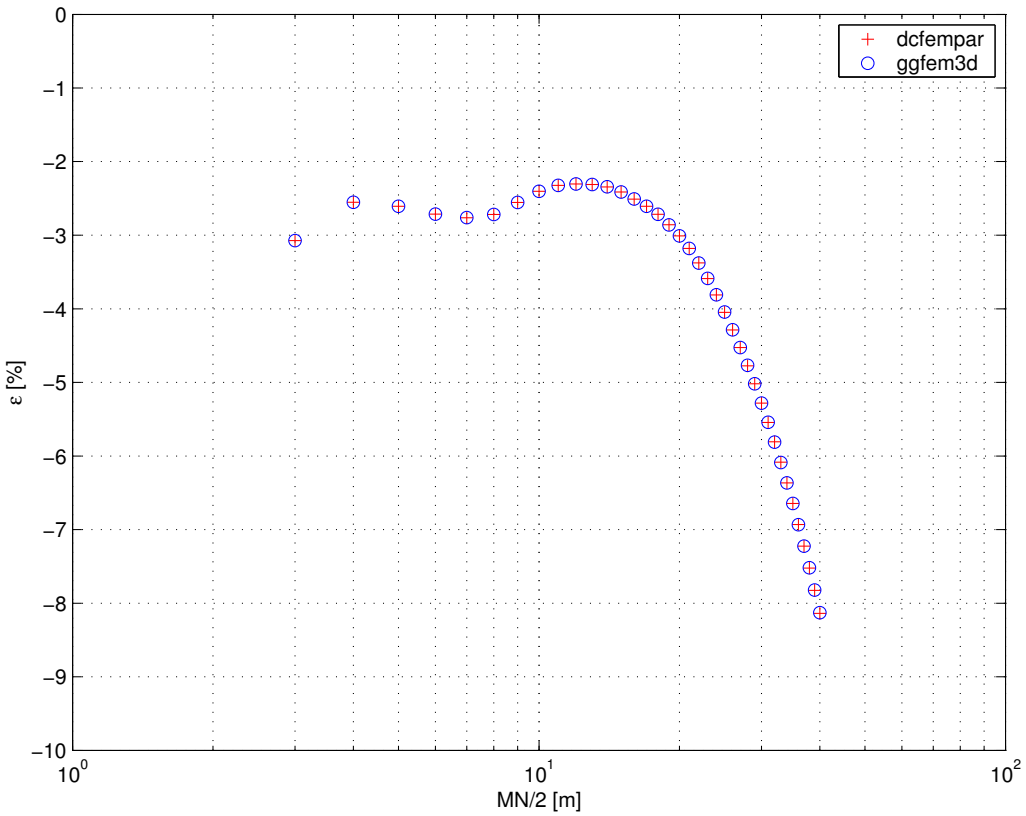


Abbildung 4.3: Relativer Fehler der Vorwärtsrechnung für den Dreischichtfall, inverse Schlumbergersondierung

[6] persönliche Kommunikation

vorgehoben. Es zeigt sich eine nahezu vollkommene Übereinstimmung für beide Programme. In der Nähe der Dipolquelle liegt der relative Fehler bei $3\,\%$ und wird zum Rand hin größer. Dies kann durch die Verwendung von ungeeigneten Randbedingungen erklärt werden.

4.1.3 Die vertikale Platte im homogenen Halbraum

Das Modell der vertikalen Platte im homogenen Halbraum ist ein in der Literatur häufig verwendetes Beispiel (Li und Spitzer, 2002). Über eine $5\,\mathrm{m}$ breite in x-Richtung zwischen $20\,\mathrm{m}$ und $25\,\mathrm{m}$ liegende, in y- und z-Richtung "unendlich" ausgedehnte vertikale Platte wurde eine Pol-Pol-Sondierung simuliert. Das Modell erstreckt sich in x- Richtung von $-850\,\mathrm{m}$ bis $850\,\mathrm{m}$, in y-Richtung von $-1050\,\mathrm{m}$ bis $1050\,\mathrm{m}$ und in z-Richtung von $0\,\mathrm{m}$ bis $-1020\,\mathrm{m}$ und ist in Abbildung A.3 dargestellt. Der Einspeisepol befand sich bei $(0, 0, 0)$, $20\,\mathrm{m}$ neben der vertikalen Platte. Der spezifische elektrische Widerstand für den homogenen Hintergrund wurde auf $100\,\Omega \cdot \mathrm{m}$ und für den Dike auf $10\,\Omega \cdot \mathrm{m}$ festgelegt. In Abbildung 4.4 ist der relative Fehler für drei verschiedene Modellierungsprogramme über dem Abstand vom Einspeisepol aufgetragen. Die analytische Lösung wurde mit dem Programm von Tillman Hanstein[7] berechnet. Rot kenn-

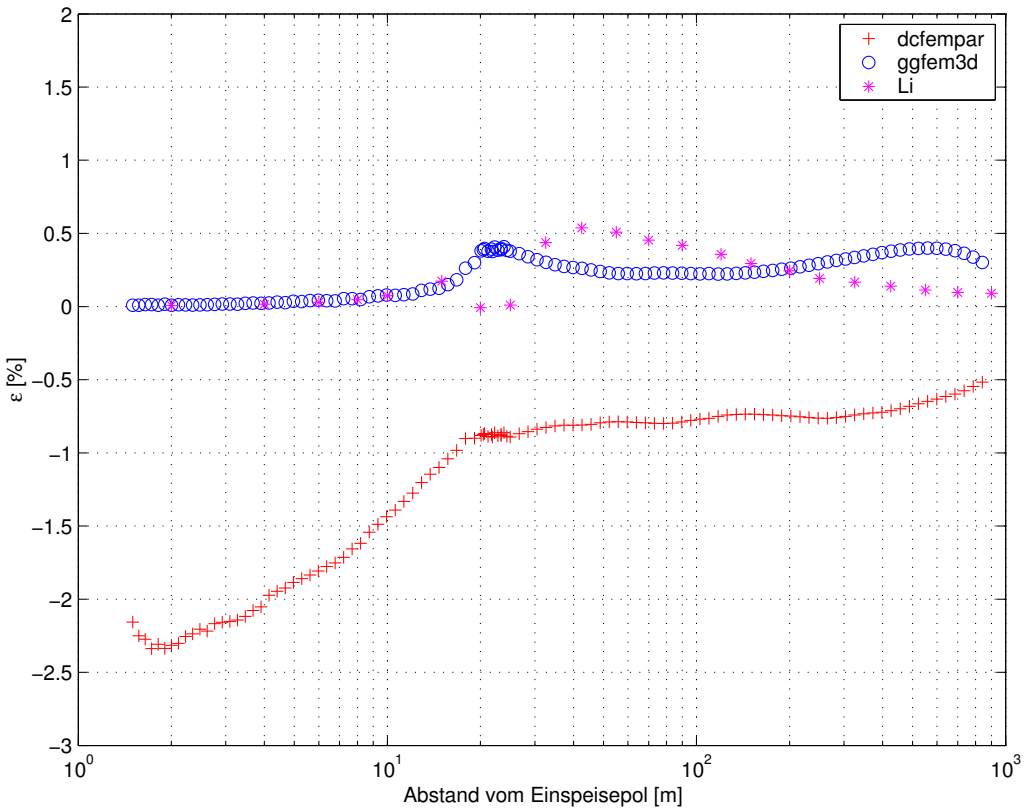


Abbildung 4.4: Relativer Fehler der Vorwärtsrechnung für die vertikale Platte im homogenen Halbraum, Pol-Pol-Sondierung

zeichnet die Ergebnisse mit dem Programm *dcfempar*, Blau die mit *ggfem3d* (Rücker, 2003) und Magenta die von Yuguo Li aus der Veröffentlichung (Li und Spitzer, 2002) zur Verfügung gestellten Ergebnisse. Die beiden zum Vergleich herangezogenen Programme verwenden zur Lösung der Vorwärtsaufgabe eine Technik zur Beseitigung von Quellsingularitäten (Lowry et al., 1989). Dementsprechend ergeben sich in der Nähe des Einspeisepols für diese Programme sehr geringe Fehler. Am Leitfähigkeitskontrast erreichen die relativen Fehler $0.5\,\%$ und nehmen zum Rand leicht ab. Für das in dieser Arbeit entwickelte Programm *dcfempar* liegen die Fehler in Quellnähe unter $2.5\,\%$ und nehmen mit größerer Entfernung von der Quelle auf unter $0.5\,\%$ ab.

4.1.4 Konvergenzkriterium

Für alle bisher gezeigten Vergleiche mit analytisch berechneten Lösungen zeigt sich eine gute Genauigkeit für das erstellte Programm. Neben den Randbedingungen und Parametern für die Gittergenerierung hat auch das verwendete Konvergenzkriterium des linearen Gleichungslösers einen Einfluss auf die erzielte Genauigkeit. In Abbildung 4.5 ist für das Modell der vertikalen Platte der relative Fehler für verschiedene Werte von $rtol$ über dem Abstand vom Einspeisepol dargestellt. Entsprechend der Legende ist jedem Wert von $rtol$ ein farbliches Symbol zugeord-

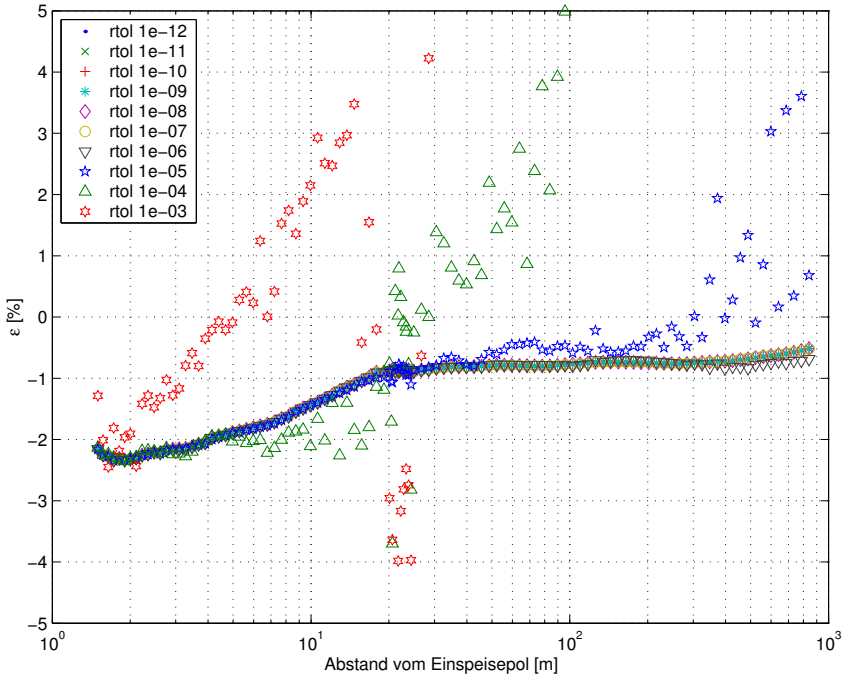


Abbildung 4.5: Relativer Fehler der Vorwärtsrechnung für verschiedene Werte von $rtol$, Pol-Pol-Sondierung

[7] persönliche Mitteilung

net. Es ist zu erkennen, dass Werte größer als 10^{-06} zu teilweise erheblichen Fehlern führen. Als guter Wert für die relative Konvergenzschranke kann 10^{-08} bis 10^{-09} gesehen werden.

4.2 Vergleich verschiedener Blockvorkonditionierer

Um ein Optimum an Effizienz zu erreichen, ist es notwendig eine geschickte Kombination aus iterativem Gleichungslöser und Vorkonditionierer zu finden. Die Systemmatrix A_h ist symmetrisch und positiv definit, weshalb als Gleichungslöser das PCG-Verfahren eingesetzt wird. Als Vorkonditionierer werden für parallele Implementierungen sogenannte Blockvorkonditionierer angewendet. In PETSc steht dafür der Block-Jacobi-Vorkonditionierer zur Verfügung. Für jeden lokalen Matrixblock (= ein Block des Blockvorkonditionierers) ist ein sequentieller Vorkonditionierer zu finden, für den einerseits das Programm mit zunehmender Prozessoranzahl skaliert und andererseits die Laufzeit des iterativen Gleichungslösers minimiert wird. Dazu können in PETSc für die einzelnen Blöcke der SSOR-, ILU- oder ICC-Vorkonditionierer verwendet werden.
Im Vorfeld des Vergleichs für die drei Blockvorkonditionierer muss für das SSOR-Verfahren ein optimaler Wert für den Relaxationsparameter ω bestimmt werden. Mit zwei Prozessoren

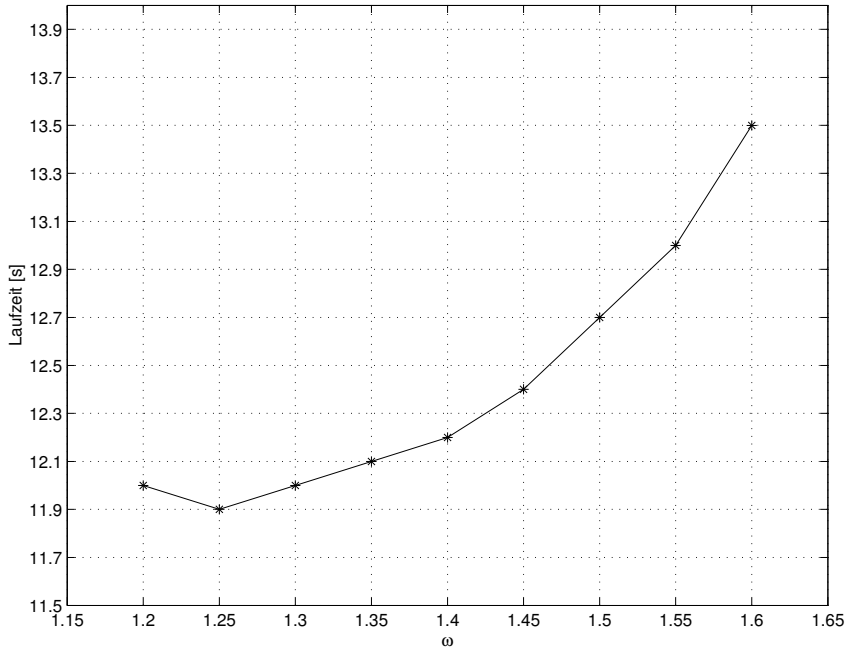


Abbildung 4.6: Vergleich der Laufzeit für verschiedene Werte von ω auf zwei Prozessoren

des Linux-Clusters[8] wurden unterschiedliche Werte zwischen 1.2 und 1.6 getestet. Bei dem verwendeten Modell handelt es sich um den Dreischichtfall, der im vorhergehenden Abschnitt

[8]Neun PCs, Institut für Geophysik, TU Bergakademie Freiberg

erläutert wurde. Die TetGen-Parameter `-q 1.1` und $dz = 0.05\,\mathrm{m}$ führen für dieses Modell auf ein Gleichungssystem mit 100500 zu bestimmenden Unbekannten. Die gemittelte Laufzeit für eine Lösung des linearen Gleichungssystems ist in Abbildung 4.6 über ω aufgetragen. Für die Mittelwertbildung wurden 243 Werte für einen ω-Wert benutzt. Mit ansteigendem Relaxationsparameter nimmt die Laufzeit des iterativen Gleichungslösers zu. Das Minimum bei $\omega = 1.25$ wird für die weiteren Untersuchungen verwendet.
Für die drei Blockvorkonditionierer ist in Abbildung 4.7 die Zeit für das Lösen eines linearen Gleichungssystems (Mittelwert aus 243 Werten) über der Prozessoranzahl dargestellt. Durch das entsprechende Symbol in der Legende werden die drei Blockvorkonditionierer unterschieden. Der Dreischichtfall wurde als Modell beibehalten und damit auch die Anzahl der Unbekannten. Bedingt durch einen Hardwaredefekt konnten bei den weiteren Untersuchungen teil-

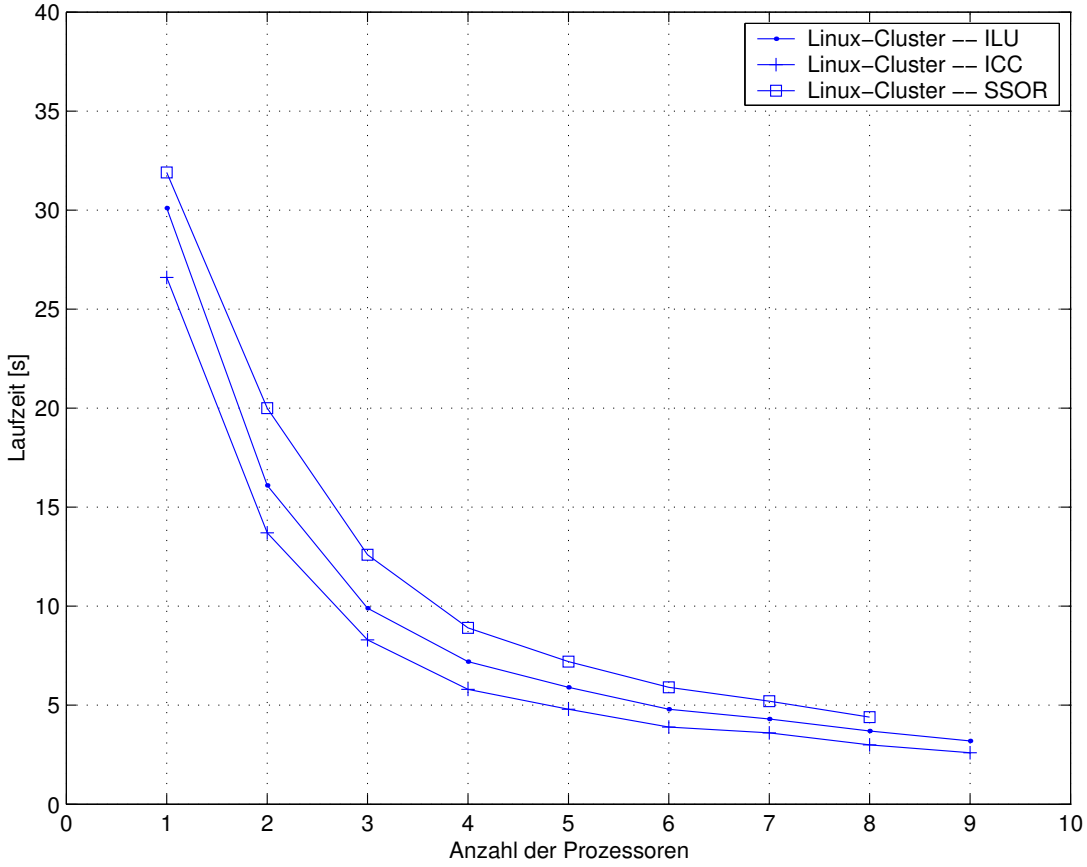


Abbildung 4.7: Vergleich der verschiedenen Blockvorkonditionierer

weise nur acht Prozessoren verwendet werden. Für alle drei Vorkonditionierer ist eine nahezu ideale Laufzeitabnahme mit der Anzahl an Prozessoren festzustellen. Die benötigte Zeit zum Lösen des linearen Gleichungssystems steigt vom ICC- über den ILU- zum SSOR-Blockvorkonditionierer an. Da die absoluten Laufzeiten für den ICC-Vorkonditionierer am geringsten sind, wird dieser für alle weiteren Untersuchungen eingesetzt.

4.3 Skalierung

Die in Abschnitt 2.3.2 aufgeführten Ansatzpunkte zur Parallelisierung sollen nun hinsichtlich ihrer effizienten Implementierung durch Laufzeitmessungen untersucht werden. Die Aufteilung des Gitters auf die einzelnen Prozessoren kann nur unzureichend durch Laufzeiten quantifiziert werden. Eine gute Aufteilung der Elemente beeinflusst direkt die Laufzeit des gesamten Programmes, da nur durch eine gute Balance gewährleistet werden kann, dass keine Leerlaufzeiten für einzelne Prozessoren entstehen. Wenn die Erwartungen an den Aufbau des linearen Gleichungssystems erfüllt werden sollen, dann muss eine lineare Abnahme der Laufzeit zu erkennen sein. Um diesen Sachverhalt genauer zu untersuchen, ist für zwei Modelle (100500 und 400500 Knoten) der Mittelwert für die Zeit zur Assemblierung bestimmt worden. Dieser ist über der Prozessoranzahl in Abbildung 4.8 für das SMP-System[9] (Symmetrisches Multiprozessorsystem) `geos3` (Rot) und den Linux-Cluster (Blau) dargestellt. Beim Modell mit 100500 Unbe-

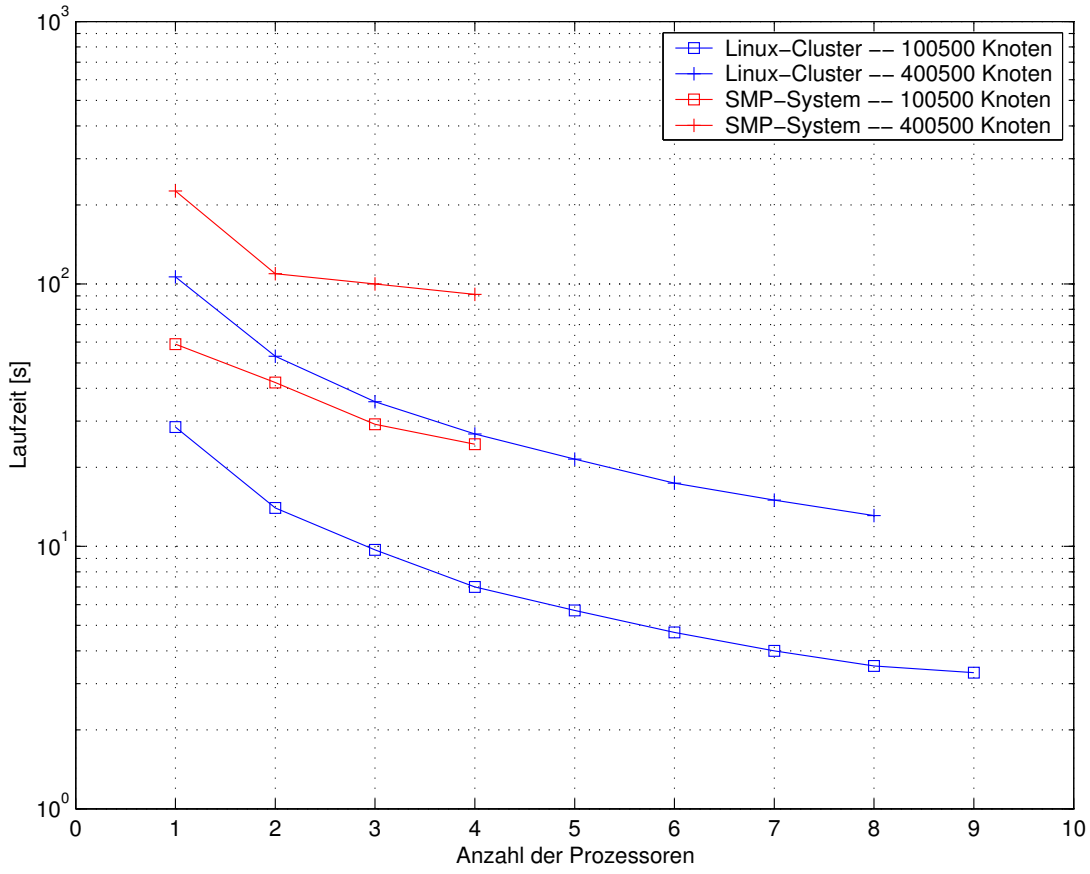


Abbildung 4.8: Laufzeit für den Aufbau des linearen Gleichungssystems

kannten handelt es sich wiederum um den Dreischichtfall aus dem vorhergehenden Abschnitt. Für das Modell eines Flussdeiches ergeben sich 400500 Knoten in der Triangulierung (Abbil-

[9] Vier Prozessoren, geos3.geo.tu-freiberg.de

dung A.4). Die Ergebnisse für den Linux-Cluster spiegeln die gestellten Erwartungen wieder. Es ist klar zu erkennen, dass sich mit Verdopplung der Prozessoranzahl die Laufzeit halbiert. Dieses Verhalten gilt für beide betrachteten Modelle. Für das SMP-System ist dies nicht der Fall. Es kann lediglich eine Halbierung der Laufzeit für vier Prozessoren erreicht werden.
Der nächste Ansatzpunkt für die Parallelisierung ist die Lösung des linearen Gleichungssystems. Es wurden wieder die gleichen Modelle benutzt. In Abbildung 4.9 ist der Mittelwert der Laufzeit für das Lösen eines linearen Gleichungssystems über der Anzahl an Prozessoren für den Linux-Cluster (Blau) und das SMP-System (Rot) aufgetragen. Die guten Resultate für den

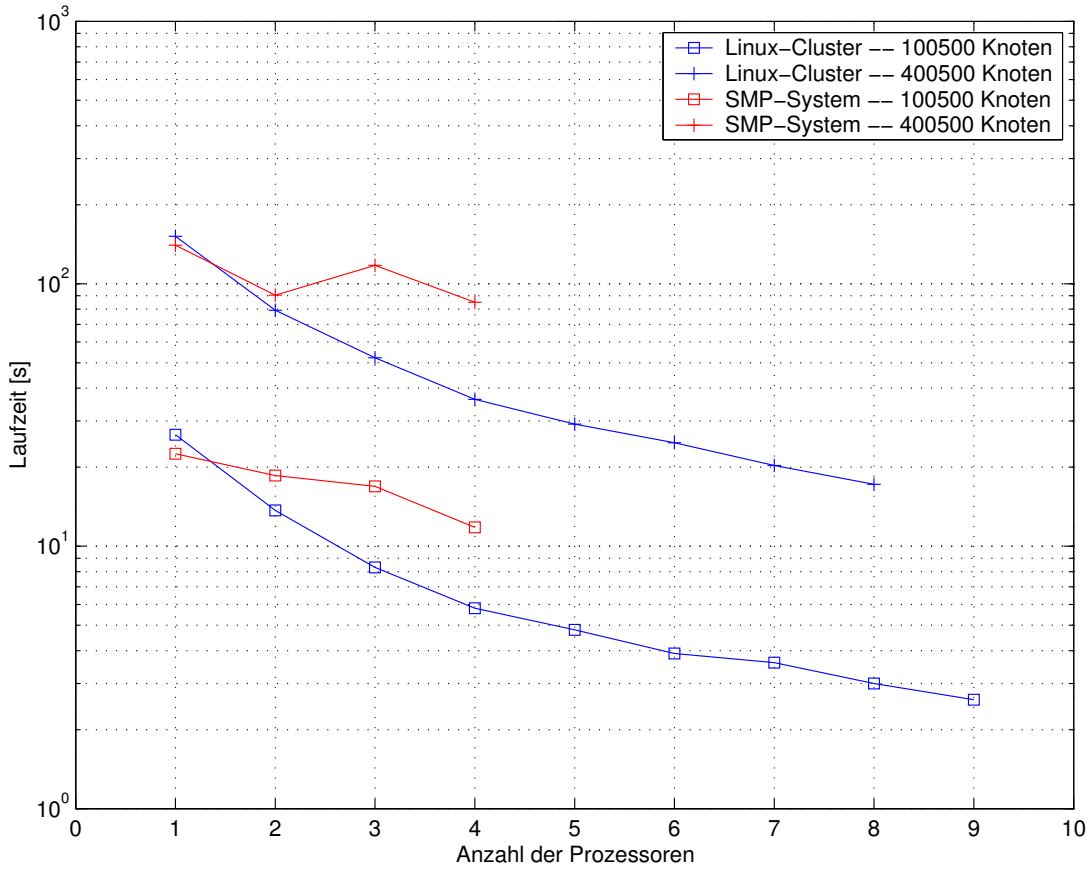


Abbildung 4.9: Laufzeit für das Lösen des linearen Gleichungssystems

Linux-Cluster bestätigen sich auch bei der iterativen Lösung des linearen Gleichungssystems. In diesem Test konnte eine nahezu ideale Laufzeitabnahme mit der Anzahl an eingesetzten CPUs gemessen werden. Das im Vergleich dazu negative Ergebnis des SMP-Systems wirft Fragen auf. Rein von den Hardwarevoraussetzungen sollte mit diesem Rechner ein vergleichbarer Laufzeitgewinn zu erreichen sein. Dass dies nicht so ist, kann u. U. durch ein ungünstiges ”Job-Scheduling” im Betriebssystemkern erklärt werden. Dies würde bedeuten, dass die parallelen Prozesse zwischen den zur Verfügung stehenden Prozessoren verschoben werden und damit

einhergehend Laufzeiteinbußen zu beobachten sind. Dieser Umstand könnte auch für die unzureichende Reproduzierbarkeit der Laufzeitmessungen auf dem SMP-System mit zwei und drei Prozessoren verantwortlich sein.
Nachdem der Nachweis erbracht werden konnte, dass die einzelnen Schritte gut parallelisiert sind, muss das Verhalten des gesamten Programmes genauer betrachtet werden. Wie durch das AMDAHLsche Gesetz belegt ist, hat der rein sequentiell auszuführende Programmcode einen wesentlichen Anteil am maximal erreichbaren Speedup. Deshalb war bei der Implementierung auf einen möglichst geringen Prozentsatz an rein sequentiell abzuarbeitenden Code zu achten. Wie der Mittelwert der Gesamtlaufzeit aus drei Durchläufen des Programmes mit der Anzahl an Prozessoren skaliert, wird in Abbildung 4.10 veranschaulicht. Bei dem kleineren Modell ist

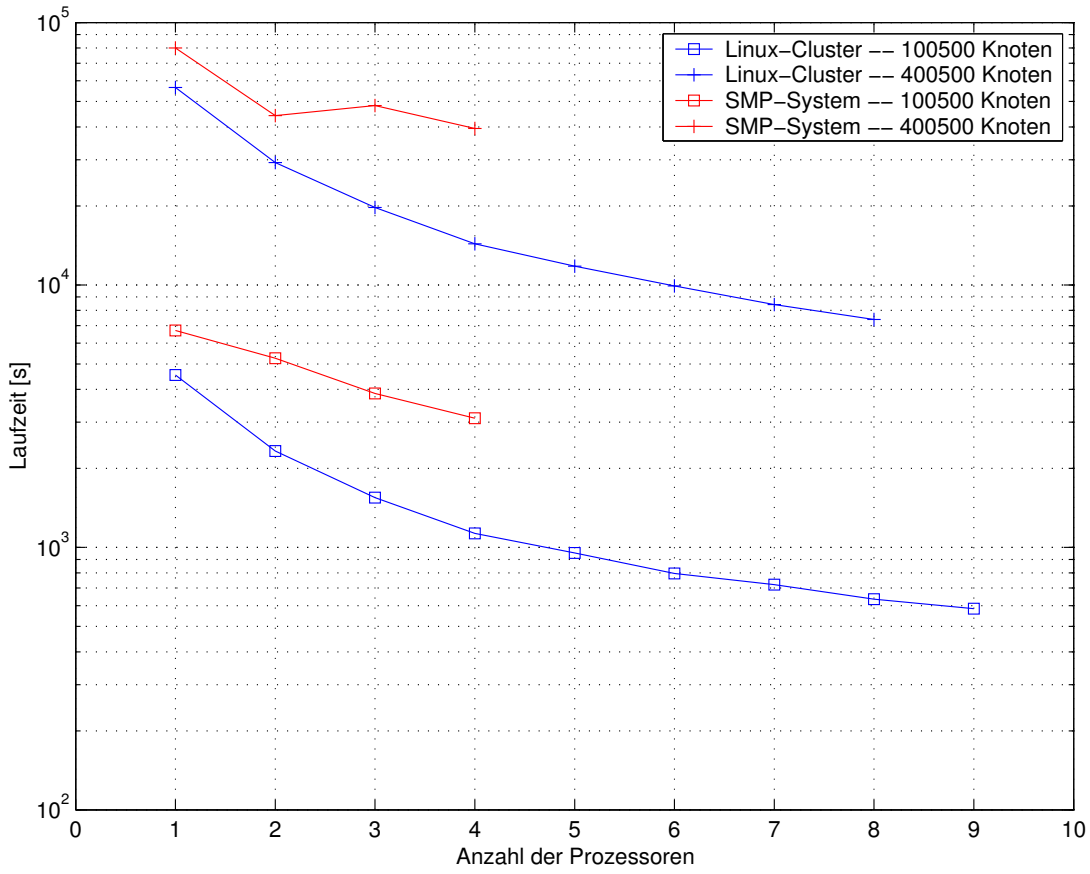


Abbildung 4.10: Gesamtlaufzeit des Programmes

die Potentialverteilung für 81 Quellpositionen und bei dem größeren für 217 Quellen zu bestimmen. Man erkennt wieder eine nahezu ideale Abnahme der Gesamtlaufzeit mit der Anzahl an Prozessoren für den Linux-Cluster. Die ungenügenden Resultate aus den Einzeluntersuchungen für das SMP-System werden durch die dargestellten Laufzeitergebnisse bestätigt. Für beide Modelle zeigen sich keine Unterschiede in der Abnahme der Laufzeit.

Wird der parallele Speedup (Gleichung (2.61)) berechnet und für beide Modelle sowie Rechnerarchitekturen über der Anzahl an Prozessoren aufgetragen, so ergibt sich der in Abbildung 4.11 dargestellte Verlauf. Für das SMP-System ist ein paralleler Speedup von zwei bei vier einge-

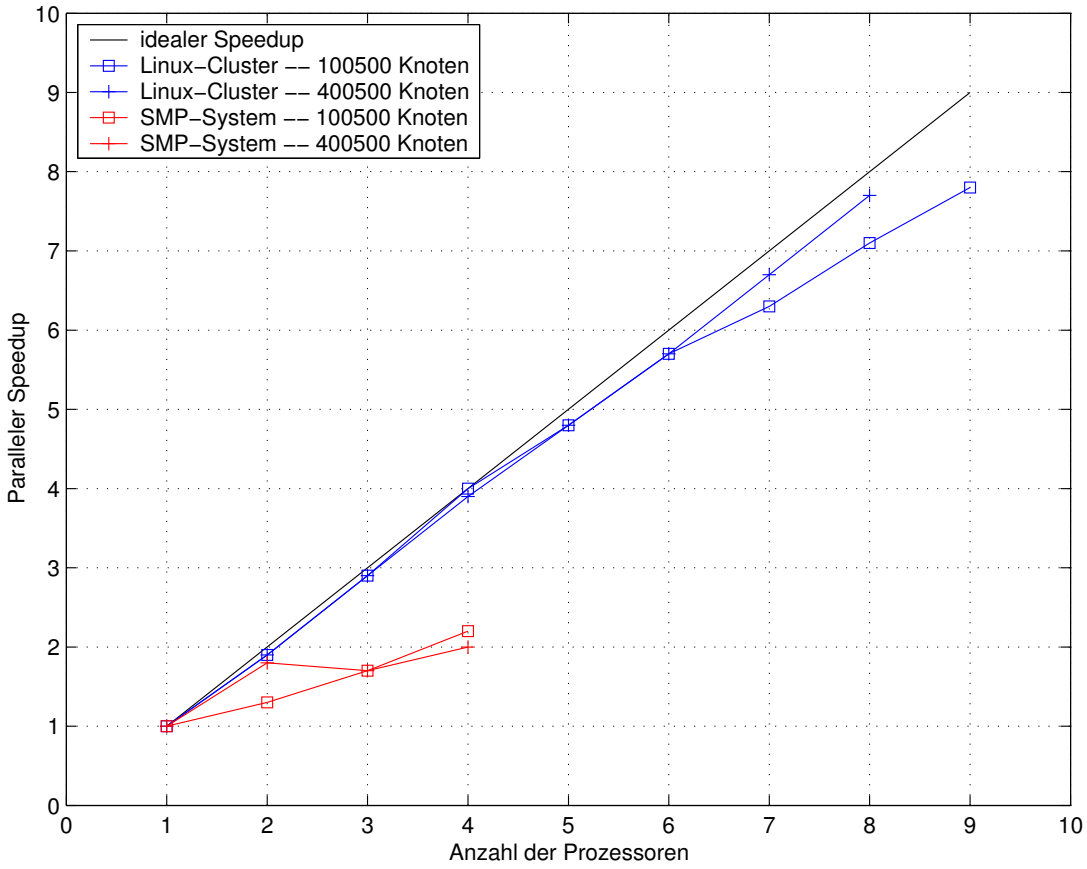


Abbildung 4.11: Paralleler Speedup für das gesamte Programm

setzten Prozessoren zu verzeichnen. Dem gegenüber wird auf dem Linux-Cluster ein Speedup von fast acht bei einer Prozessoranzahl von acht erreicht. Für das Modell mit 100500 zu bestimmenden Unbekannten verringert sich die Zunahme des Speedup ab sieben Prozessoren. Ab 10 bis 12 Prozessoren könnte der Punkt erreicht sein, ab dem der Speedup nicht mehr ansteigt. Das würde bedeuten, es wird mehr Zeit für die Kommunikation als für die Berechnung aufgewendet, weil die einzelnen Systemmatrixblöcke zu klein sind. Der gleiche Effekt würde für das Modell mit 400500 Knoten ab 20 bis 25 Prozessoren eintreten.
Insgesamt konnten die gestellten Erwartungen mehr als übertroffen werden. Bei acht Prozessoren ist ein paralleler Speedup von 7.7 für ein Modell mit 400500 Knoten erreichbar.
Für alle in dieser Arbeit dargestellten Laufzeiten liegt der relative Fehler der Laufzeitmessung unter 1% (Verhältnis von Standardabweichung zu Mittelwert der Laufzeit).

5 Zusammenfassung

Ziel dieser Diplomarbeit war es, ein parallelisiertes Vorwärtsmodellierungsprogramm für die Gleichstromgeoelektrik auf Basis der Finite-Elemente-Methode zu erstellen. Bei der Entwicklung des Programmes wurde auf vorhandene und frei verfügbare Programmpakete zurückgegriffen. Dadurch konnte innerhalb dieser Arbeit in einem relativ kurzen Zeitraum ein paralleles, 3D FEM-Modellierungsprogramm, *dcfempar*, entwickelt werden. Zu den verwendeten Paketen zählen der Gittergenerator TetGen (Si, 2003), die FEM-Bibliothek libMesh (Kirk et al., 2004) und als Gleichungslöser PETSc (Balay et al., 2001) sowie die MPI-Implementierung MPICH (Gropp und Lusk, 1996). In den Entwicklungsprozess sind ebenso die Erfahrungen von Carsten Rücker mit der Generierung von unstrukturierten Tetraedergittern und die Programmentwicklungen von Thomas Günther zur Visualisierung der Ergebnisse eingeflossen. Das modulare Entwicklungskonzept hat sich als vorteilhaft erwiesen und ist für die Lösung zukünftiger geophysikalischer Problemstellungen weiterzuempfehlen.

Die verschiedenen Genauigkeitsuntersuchungen für das erstellte Programm *dcfempar* zeigten im Vergleich zu den Ergebnissen anderer Programme gute Resultate. Der Vergleich der Modellierungsergebnisse mit denen eines zweiten FE-Programmes, auf der Basis desselben unstrukturierten Tetraedergitters, führte zu einer Übereinstimmung innerhalb numerischer Fehler. Es konnte gezeigt werden, dass auf die Genauigkeit mehrere Faktoren Einfluss nehmen. Dies sind insbesondere der Typ der Randbedingung, die TetGen-Parameter und der Wert des Konvergenzkriteriums. Durch eine günstige Kombination der Einflussfaktoren konnte eine hohe Genauigkeit und eine optimale Nutzung der Ressourcen erzielt werden. In zukünftigen Betrachtungen sollte der Einfluss des Gitters auf die Genauigkeit intensiver untersucht werden.

Hinsichtlich der Laufzeit, bei wachsender Zahl an eingesetzten Prozessoren erwies sich das PCG-Verfahren mit dem Block-Jacobi-Vorkonditionierer und der unvollständigen Cholesky-Zerlegung als Blockvorkonditionierer als günstigste Kombination. Daneben sind in Zukunft die Verwendung weiterer PETSc-Parameter und anderer Gleichungslöser (Multigrid) in Betracht zu ziehen.

Den durch die Parallelisierung erreichbaren Laufzeitgewinn beeinflussen die sequentiellen Programmteile in einem geringeren Maße als erwartet. In den Skalierungstests konnte ein maximaler paralleler Speedup von 7.7 bei acht Prozessoren festgestellt werden. Damit sind die an das entwickelte Programm gestellten Erwartungen übertroffen worden.

Ansatzpunkte für weitere Verbesserungen sind vor allem auf zwei Gebieten zu suchen. Durch verbesserte Randbedingungen und die Technik der Singularitätenbeseitigung kann eine weitere Reduzierung der Fehler am Rand und in Quellennähe erreicht werden. Die Anzahl der benötigten Freiheitsgrade kann bei gleicher Genauigkeit u. U. durch die Verwendung von An-

satzfunktionen höherer Ordnung verringert werden. In weiterführenden Arbeiten müssen andere Elementtypen hinsichtlich der damit erzielbaren Genauigkeit im Verhältnis zum Aufwand der Gittergenerierung und zur Anzahl der Knoten untersucht werden. Innerhalb der libMesh-Bibliothek sollte vorrangig die Implementierung der Ein- und Ausgaberoutinen auf die parallele Abarbeitung angepasst werden. Außerdem sind weitere Verbesserungen bei der Verwaltung der Datenstrukturen und die Beseitigung kleinerer Probleme wünschenswert. Erst mit diesen Änderungen wird es möglich sein, das Konvergenzverhalten des FEM-Programmes für zunehmend verfeinerte Gitter zu überprüfen und damit die Fehler unter ein gewünschtes Maß zu senken. Mit einer graphischen Benutzeroberfläche könnte das Programm nutzerfreundlich gestaltet werden.

A Verwendete Tetraedergitter

Zur Visualisierung der unstrukturierten Tetraedergitter in den Abbildungen A.1 bis A.4 wurde das frei verfügbare Programm MEDIT (INRIA-Rocquencourt, 2003) verwendet. Die verschiedenen Farben heben die im entsprechenden Modell vorhandenen Leitfähigkeitsstrukturen hervor. Die Lage der Elektrodenkette an der Oberfläche wird durch die Verdichtung der Tetraederelemente in der Mitte der Abbildungen verdeutlicht.
Die Vorteile von unstrukturierten Tetraedergittern durch ihre hohe Flexibilität können anhand der Darstellungen A.1 bis A.4 veranschaulicht werden. Mit diesen Gittern ist es möglich, nahezu jede Modellgeometrie nachzuempfinden. Darüber hinaus ist eine Minimierung der Fehler an Leitfähigkeitskontrasten und Quellsingularitäten durch eine lokale Verringerung der Elementgröße möglich.
Die Schnitte durch die Modelle in den Abbildungen A.1, A.2 und A.3 zeigen an der Oberfläche die enorme Verfeinerung um die Elektrodenposition durch sehr kleine Elemente. Die gleichzeitig in diesen Modellen vorhandene Vergrößerung der Tetraederelemente in den Gebieten wo ohnehin nur eine geringere Genauigkeit gefordert ist (am Rand), bewirkt eine effiziente Nutzung des Speichers. Abbildung A.4 stellt einen Ausschnitt aus dem Modell eines Deichköpers dar. Die bei unstrukturierten Tetraederdiskretisierungen vorhandene Flexibilität wird durch die Nachbildung der Topographie und der enthaltenen Leitfähigkeitsstrukturen verdeutlicht.

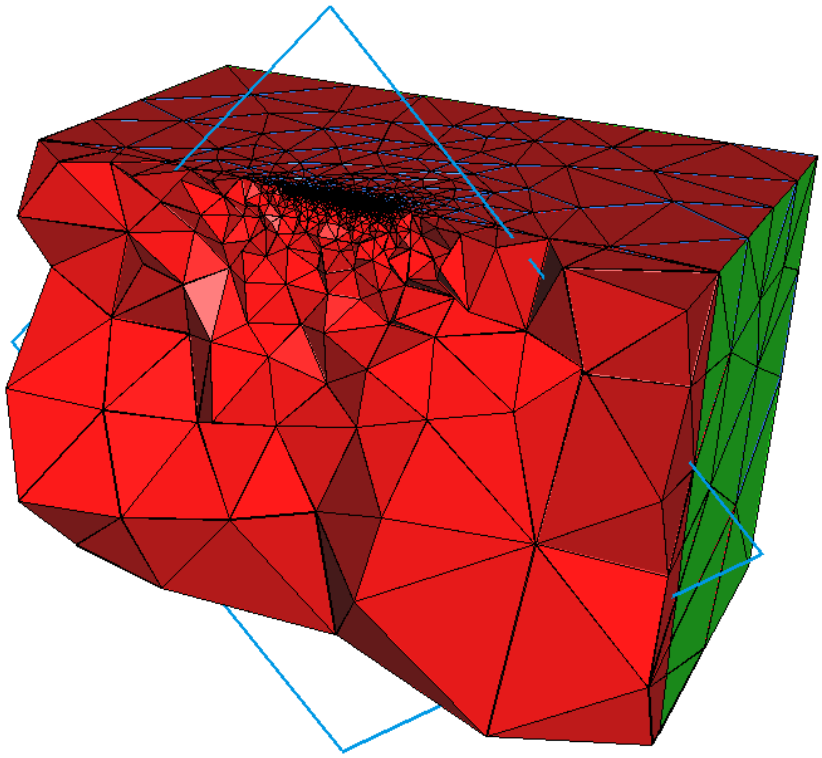


Abbildung A.1: Unstrukturiertes Tetraedergitter für den homogenen Halbraum

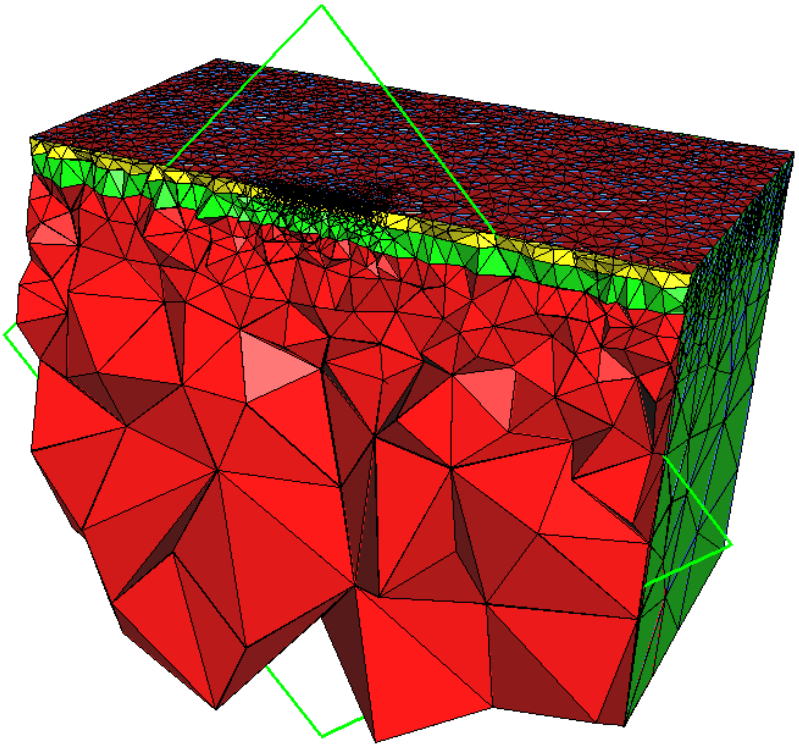


Abbildung A.2: Unstrukturiertes Tetraedergitter für den Dreischichtfall

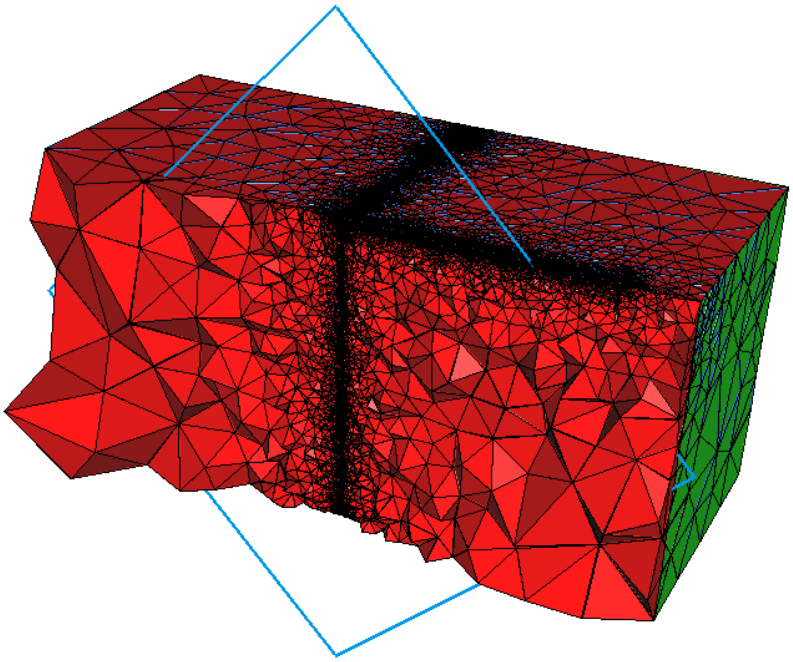


Abbildung A.3: Unstrukturiertes Tetraedergitter für die vertikale Platte im homogenen Halbraum

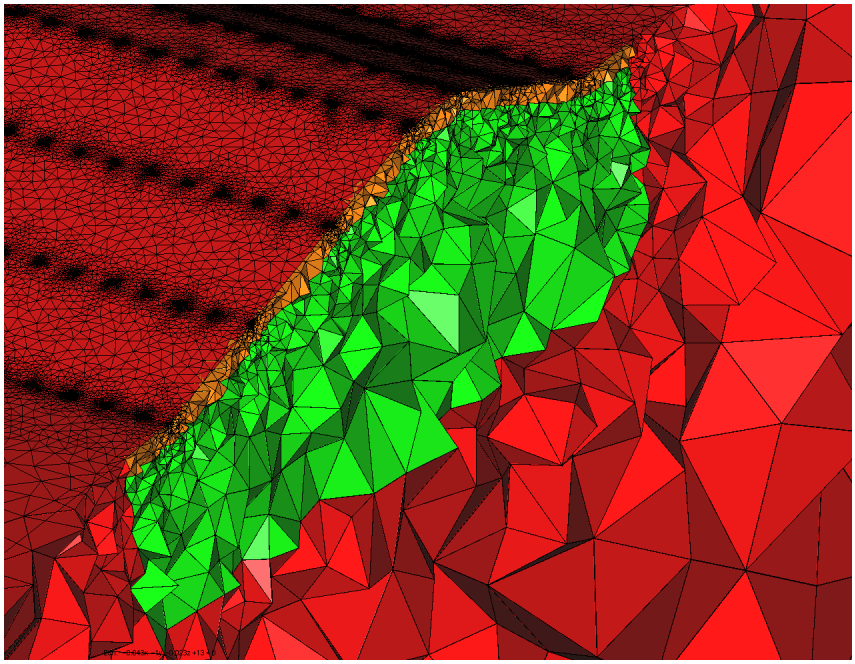


Abbildung A.4: Unstrukturiertes Tetraedergitter für das Deichmodell

B Inhalt der beiliegenden CD

Dieser Arbeit ist eine CD beigelegt, auf der sich die verwendeten Softwarepakete, der Quelltext des Programmes *dcfempar* und die Eingabedateien im TetGen-Format für die verwendeten Modelle sowie einige kleine MATLAB-Routinen befinden.
Der Ordner `Source` enthält den Quelltext von TetGen in der Version $1.2c$. Weiterhin sind die libMesh-Bibliothek in der Version $0.4.1$ und ein notwendiger Patch für libMesh beigelegt. Den Quelltext des erstellten Programmes *dcfempar* enthält der Ordner `DCFEMPar`. Für die in dieser Arbeit verwendeten Modelle befinden sich in `Tetraedergitter` die Eingabedateien (Endung `poly`) und die zugehörigen Dateien für die Kodierung des spezifischen elektrischen Widerstands (Endung `rho`). Im Verzeichnis `Darstellung` sind die notwendigen MATLAB-Routinen[10] zur Visualisierung der Ergebnisse gespeichert. Die Datei `ReadMe.txt` enthält gleichfalls die hier dargestellten Erläuterungen.

B.1 Installation der Softwarepakete

In dieser Arbeit wurde als Betriebssystem ein Debian GNU/Linux verwendet. Prinzipiell können die einzelnen Komponenten auch auf jeder anderen Plattform übersetzt werden, die von dem entsprechenden Softwarepaket unterstützt wird. Im Folgenden sollen kurz die notwendigen Schritte zum Erzeugen der ausführbaren Dateien dargestellt werden. Diese können aber von System zu System geringfügig differieren.
Bevor die einzelnen Pakete übersetzt werden, ist es ratsam einen neuen Ordner anzulegen. In diesen kopiert man die Archivdateien von TetGen und libMesh, sowie den Ordner `DCFEMPar`. Zuerst sind die Quelltexte von TetGen über einen Aufruf von:

```
tar xzf tetgen1.2c_2.tgz
```

zu entpacken. Danach kann in dem Ordner `tetgen1.2c_2` mit dem Unterverzeichnis `src` über `make` der Quelltext des Gittergenerators in Maschinencode übersetzt werden. Das ausführbare Programm *tetgen* befindet sich daraufhin im Unterverzeichnis `bin`.
Um die Parallelisierung des Programmes *dcfempar* nutzen zu können, muss auf dem System eine entsprechende MPI- und PETSc-Installation vorhanden sein. Was dabei zu tun ist, kann in den zugehörigen Anleitungen nachgeschlagen werden. Es ist nur wichtig, dass die entsprechenden Shell-Variablen `PETSC_DIR` und `PETSC_ARCH` korrekt gesetzt sind. Um die Quellen von libMesh zu übersetzten, muss zuerst das Archiv über:

```
tar xzf libmesh-0.4.1.tar.gz
```

entpackt werden. Nach dem Wechsel in das neu entstandene Verzeichnis muss die Patchdatei `TetGen-libmesh.patch` in dieses kopiert und über den Aufruf von:

[10] Von Thomas Günther zur Verfügung gestellt.

`patch -p1 < TetGen-libmesh.patch`
angewendet werden. Daraufhin kann die libMesh-Bibliothek konfiguriert und übersetzt werden. Mit dem Aufruf:
`./configure -disable-complex CXX=g++-3.3 CC=gcc-3.3 F77=g77-3.3`
wird die Bibliothek für die Nutzung mit PETSc eingerichtet und mit der GNU Compiler Collection in der Version 3.3 übersetzt. Falls es notwendig ist andere Optionen zu verwenden, hilft folgender Aufruf:
`./configure -help`.
Nach dem Befehl:
`make all`
ist die Bibliothek bereit für die Verwendung.
Zum Übersetzen der Quellen von *dcfempar* muss der Pfad zur libMesh-Bibliothek über:
`export LIBMESH_DIR=<PFAD>`
angegeben werden. Nach dem Wechsel in das Verzeichnis `DCFEMPar` ist nur noch ein Aufruf des Befehls:
`make dcfempar`
notwendig, um die ausführbare Datei *dcfempar* im Verzeichnis vorliegen zu haben.
Für die weitere Nutzung empfiehlt es sich, für *tetgen* und *dcfempar* einen symbolischen Link mit dem Kommando:
`ln -s <Quelle> <Ziel>`
anzulegen.

B.2 Erzeugen der Tetraedergitter

Im Ordner `Tetraedergitter` sind die Eingabedateien für TetGen zu finden. Für die entsprechende `poly`-Datei kann das unstrukturierte Tetraedergitter über:
`tetgen -pAezICVq1.2T1e-12 HomoHR.poly`
erzeugt werden. Falls es zu Fehlern kommt, sollte der Wert für die interne Null `-T1e-12` oder der Wert für den Qualitätsparameter `-q1.2` verändert werden. Im Anschluss werden für die Vorwärtsmodellierung die Dateien mit der Endung `ele` und `node` benötigt. In der `node`-Datei sind die Koordinaten aller Gitterknoten gespeichert. Bei der Implementierung der Einleseroutinen in libMesh war es notwendig, alles möglichst portabel zu gestalten, sodass eine kleinere Änderung im Kopf der `node`-Datei erforderlich ist. Die erste Zeile hat folgende Struktur `61461 3 0 1`. Dies muss geändert werden in `61461 3 1 0`. Dabei gibt der erste Wert die Knotenanzahl, der Zweite die Anzahl der Koordinaten, der Dritte die Anzahl an Parametern an. Der vierte Wert gibt an, ob ein sogenanntes "boundary flag"gesetzt ist oder nicht. Die `ele`-Datei mit der Elementzusammenhangstabelle kann unverändert übernommen werden.

B.3 Programmaufruf

Der Aufruf:

`dcfempar -help`

zeigt eine kurze Hilfe zu den Optionen des Programmes an. Über die im Modell des homogenen Halbraums enthaltene Elektrodenkette kann durch:

`dcfempar -i HomoHR.node -a HomoHR.rho -o HomoHR -b -2 -M t -v`

eine komplette Pol-Pol-Messung simuliert werden. Die beiden Ausgabedateien `HomoHR.dat` und `HomoHR.collect` enthalten alle notwendigen Informationen um aus den simulierten Daten beliebige Pseudosektionen zu generieren. In der `dat`-Datei ist eine Liste der Elektrodenpositionen mit den dazugehörigen Koordinaten gespeichert. Die `collect`-Datei beinhaltet eine $n \times n$ Matrix. Jede Zeile charakterisiert dabei eine Quellposition. In der 15. Zeile sind demnach alle Potentialwerte an den Elektroden für die 15. Quellposition der Elektrodenkette gespeichert.

B.4 Darstellung

Die Datei `PolPol.dat` im Ordner `Darstellung` zeigt am Beispiel für das Modell des homogenen Halbraumes wie der Inhalt einer `dat`-Datei erweitert werden muss, um eine Pol-Pol-Messung mit dem Einspeisepunkt $(0, 0, 0)$ zu simulieren. Mit dieser und der `collect`-Datei können die Ergebnisse in MATLAB über folgende Befehle:

```
1  MEA=load('HomoHR.collect');
2  N=readinv3dfile('PolPol.dat');
3  Rhos=collectrhoa(N,MEA);
4  plot ( N.elec(N.m,2), Rhos);
```

dargestellt werden. In Zeile 1 wird die `collect`-Datei geladen. Die folgende Zeile liest die Lage der einzelnen Elektroden und die zu simulierende Messanordnung ein. Über die Routine `collectrhoa` in Zeile 3 werden die scheinbaren spezifischen Widerstände berechnet und in der Variablen `Rhos` gespeichert. Der `plot`-Befehl stellt diese dann über dem Abstand vom Einspeisepol dar. Zur Darstellung kompletter Pseudosektionen sind in dem Verzeichnis spezielle Routinen vorhanden. Mehr über die einzelnen MATLAB-Funktionen ist in der entsprechenden `m`-Datei zu finden.

Literatur

Balay, S., Buschelman, K., Gropp, W. D., Kaushik, D., Knepley, M., McInnes, L. C., Smith, B. F. und Zhang, H. (2001). *PETSc home page.* (`http://www.mcs.anl.gov/petsc`)

Balay, S., Gropp, W. D., McInnes, L. C. und Smith, B. F. (1997). Efficient management of parallelism in object oriented numerical software libraries. In E. Arge, A. M. Bruaset und H. P. Langtangen (Hrsg.), *Modern software tools in scientific computing* (S. 163–202). Birkhauser Press.

Barrett, R., Berry, M., Chan, T. F., Demmel, J., Donato, J., Dongarra, J., Eijkhout, V., Pozo, R., Romine, C. und van der Vorst, H. (1994). *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Edition.* Philadelphia, PA: SIAM.

Bing, Z. und Greenhalgh, S. A. (2001). Finite element three-dimensional direct current resistivity modelling: accuracy and efficiency considerations. *Geophys. J. Int.*, *145*, 679–688.

Braess, D. (Hrsg.). (1997). *Finite Elemente : Theorie, schnelle Löser und Anwendung in der Elastizitätstheorie.* Springer-Verlag Berlin, Heidelberg, New York.

Coggon, J. H. (1971). Electromagnetic and electrical modeling by the finite element method. *Geophysics*, *36*, 131–155.

Demmel, J., Heath, M. und van der Vorst, H. (1993). Parallel numerical linear algebra. In *Acta numerica 1993* (S. 111–198). Cambridge, UK: Cambridge University Press. (`citeseer.nj.nec.com/article/demmel93parallel.html`)

Dey, A. und Morrison, H. F. (1979). Resistivity modeling for arbitrarily shaped three-dimensional structures. *Geophysics*, *44*, 753–780.

Gropp, W. D. und Lusk, E. (1996). *User's guide for* `mpich`*, a portable implementation of MPI.* (ANL-96/6)

Günther, T. (2000). *Vergleichende Analyse zum Auflösungsvermögen geoelektrischer und elektromagnetischer Verfahren anhand von Sensitivitätsstudien.* Diplomarbeit, Institut für Geophysik, Technische Universiät Bergakademie Freiberg.

Huber, W. (Hrsg.). (1997). *Paralleles Rechnen: Eine Einführung von Walter Huber.* München: R. Oldenbourg Verlag.

INRIA-Rocquencourt. (2003). *Medit: A user-friendly mesh viewer. Gamma project at INRIA-Rocquencourt.* (`http://www-rocq1.inria.fr/gamma/medit/medit.html`)

Jung, M. und Langer, U. (Hrsg.). (2001). *Methode der finiten Elemente für Ingenieure.* B. G. Teubner Stuttgart, Leipzig, Wiesbaden.

Kemna, A. (2000). *Tomographic Inversion of Complex Resistivity: Theory and Application.* Dissertation, Institut für Geophysik, Ruhr-Universität Bochum.

Kirk, B. S., Peterson, J. W., Anderson, M. L., Barth, W. L., Petersen, S., Dreyer, D., Heijden, H. van der und Prill, F. (2004). *libMesh home page.* (`http://libmesh.sourceforge.net`)

Knödel, K., Krummel, H. und Lange, G. (Hrsg.). (1997). *Handbuch zur Erkundung des Untergrundes von Deponien und Altlasten – Geophysik* (Bd. 3). Springer Verlag Berlin, Heidelberg.

Li, Y. und Spitzer, K. (2002). Three-dimensional DC resistivity forward modelling using finite elements in comparison with finite-difference solutions. *Geophys. J. Int., 151,* 924–934.

Lowry, T., Allen, M. B. und Shive, P. N. (1989). Singularity removal: A refinement of resistivity modeling techniques. *Geophysics, 54*(6), 766–774.

Militzer, H. und Weber, F. (Hrsg.). (1985). *Angewandte Geophysik* (Bd. 2). Springer Verlag Wien/Akademie-Verlag Berlin.

Oeser, J. (2002). *Lösung großer linearer Gleichungssysteme in Linux-Clustern mittels MPI und PETSc.* Unveröffentl. Studienarbeit, Institut für Geophysik, Technische Universität Bergakademie Freiberg.

Owen, S. J. (1998). A Survey of Unstructured Mesh Generation Technology. In *Proceedings 7th International Meshing Roundtable.* Dearborn, MI.

Pridmore, D. F., Hohmann, G. W., Ward, S. H. und Sill, W. R. (1981). An investigation of finite-element modeling for electrical and electromagnetic data in three dimensions. *Geophysics, 46,* 1009–1024.

Rücker, C. (1999). *Die Methode der Finiten Elemente mit adaptiver Gittererzeugung und Integration eines Mehrgitterverfahrens zum Lösen des gleichstromgeoelektrischen Vorwärtsproblems.* Diplomarbeit, Institut für Geophysik und Geologie, Universität Leipzig.

Rücker, C. (2003). 3D-FEM Widerstandsmodellierung unter Verwendung von unstrukturierten Tetraederdiskretisierungen. In K. Bahr und A. Junge (Hrsg.), *Protokoll 20. Kolloquium "Elektromagnetische Tiefenforschung", in Druck.* Neustadt an der Weinstraße.

Saad, Y. (Hrsg.). (1996). *Iterative Methods for sparse linear systems.* Boston, MA: PWS Publishing Company.

Sasaki, Y. (1994). 3-D resistivity inversion using the finite-element method. *Geophysics, 59,* 1839–1848.

Si, H. (2003). TETGEN-*1.2: A quality tetrahedral mesh generator.* (`http://tetgen.berlios.de`)

Spitzer, K. (1995). A 3-D finite-difference algorithm for DC resistivity modelling using conjugate gradient method. *Geophys. J. Int., 123,* 903–914.

Van der Vorst, H. A. (1993). *Lecture notes on iterative methods.* (`www.math.ruu.nl/people/vorst/cgnotes.ps.gz`)