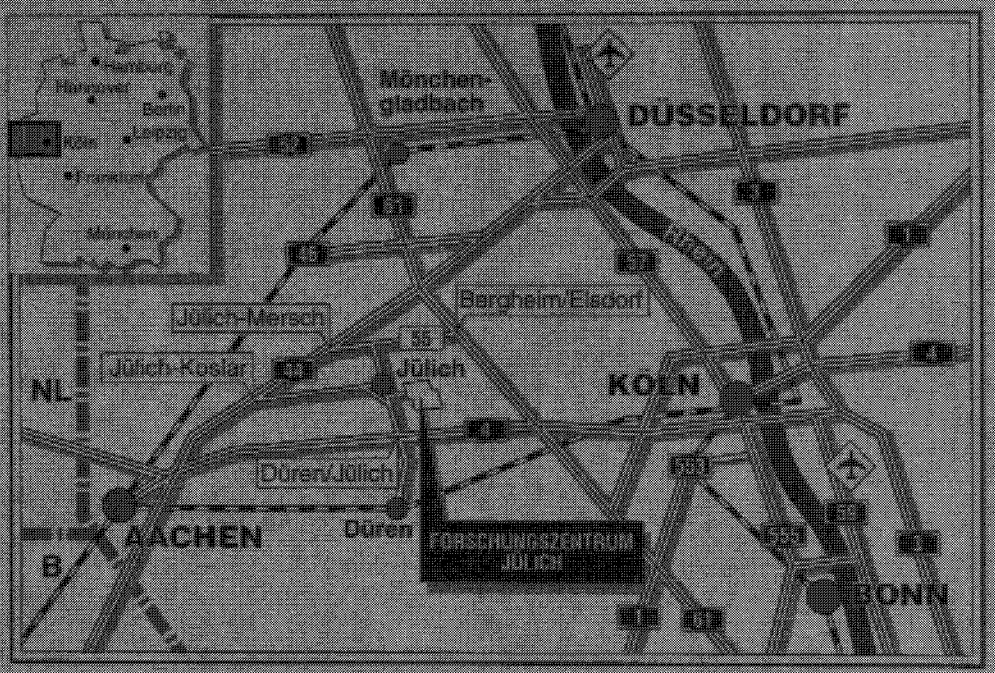
FORSCHUNGSZENTRUM JÜLICH GmbH
KFA

Zentralinstitut für Angewandte Mathematik

Entwicklung und Implementierung einer dreidimensionalen Partitionierungsstrategie für das Programm TRACE auf einem massiv-parallelen Rechner

Ralf Wimmershoff



Berichte des Forschungszentrums Jülich ; 3157
ISSN 0944-2952
Zentralinstitut für Angewandte Mathematik Jül-3157

Zu beziehen durch: Forschungszentrum Jülich GmbH · Zentralbibliothek
D-52425 Jülich · Bundesrepublik Deutschland
Telefon: 02461/61-6102 · Telefax: 02461/61-6103 · Telex: 833556-70 kfa d

Entwicklung und Implementierung einer dreidimensionalen Partitionierungsstrategie für das Programm TRACE auf einem massiv-parallelen Rechner

Ralf Wimmershoff

Kurzfassung

Im Rahmen der Arbeit wird eine Strategie zur Gebietszerlegung eines regulären Finiten-Elemente-Gitters entworfen und im Programm TRACE, einem in FORTRAN 77 geschriebenen Simulationsprogramm aus dem Bereich der Hydrogeologie, implementiert. Die Parallelisierung des Programms basiert auf dem Schwarz'schen Verfahren, bei dem das Finite-Elemente-Gitter in einander überlappende Teilgitter zerlegt wird. Durch die Zerlegung können einzelne FE-Knoten in bis zu acht Teilgittern auftreten. Damit eine geeignete, dreidimensionale Zerlegung während der Programmausführung gefunden werden kann, wird eine Funktion definiert, die den durch die Zerlegung hervorgerufenen Mehraufwand bei der Berechnung angibt. Hiermit kann eine Zerlegung automatisch durchgeführt werden, so daß die Zahl der redundant zu berechenden FE-Knoten minimal ist. Während der Berechnungen müssen infolge des Schwarz'schen Verfahrens Informationen zwischen den einzelnen Teilgittern ausgetauscht werden. Hierbei kann ein Teilgitter Informationen von bis zu 26 benachbarten Teilgebieten erhalten. Der Aufbau der auszutauschenden Bereiche wird in der vorliegenden Arbeit erläutert. Die zur parallelen Programmausführung notwendigen Änderungen bzw. Erweiterungen des Quell-Codes von TRACE werden aufgezeigt. Das parallele Programm wird für zwei unterschiedliche Simulationsmodelle, die aus bis zu 256000 FE-Knoten bestehen, auf einer Intel Paragon ausgeführt, wobei der Einfluß unterschiedlicher Prozessorzahlen bzw. eines variierenden Überlapps untersucht wird.

Rheinisch-Westfälische Technische Hochschule Aachen
Lehrstuhl für Technische Informatik und Computerwissenschaften
Prof. Dr. F. Hoßfeld

Entwicklung und Implementierung einer dreidimensionalen Partitionierungsstrategie für das Programm TRACE auf einem massiv-parallelen Rechner

Diplomarbeit im Fach Elektrotechnik

vorgelegt von

Ralf Wimmershoff
Matr.-Nr.: 164546

Aachen, im Februar 1995

Betreuung durch:

Dr. Michael Weber
Zentralinstitut für Angewandte Mathematik
Forschungszentrum Jülich

Πάντα ῥεῖ – **alles fließt.**
Heraklit (um 500 v. Chr.)

Inhaltsverzeichnis

Einleitung

Auf die fortschreitende Industrialisierung, das enorme Bevölkerungswachstum und die intensive landwirtschaftliche Nutzung großer Gebiete lassen sich eine Vielzahl von Umweltproblemen zurückführen. Insbesondere die Qualität des Grundwassers ist in den letzten Jahrzehnten erheblich beeinträchtigt worden. Das Grundwasser ist Bestandteil einer auf der Erde zirkulierenden Wassermenge von ungefähr $1,38 \cdot 10^{18}\ m^3$, wobei der Anteil des Grundwassers nur circa $7 \cdot 10^{15}\ m^3$ beträgt [HOE89]. Da die Fließgeschwindigkeit des Grundwassers sowohl in vertikaler als auch in horizontaler Richtung sehr gering ist, machen sich die Auswirkungen von Schadstoffen, die heute in dieses System gelangen, häufig erst nach vielen Jahrzehnten bemerkbar. Die Qualität und die Verfügbarkeit des Grundwassers sind von großer Wichtigkeit, da Wasser eine existentielle Grundlage für das Leben auf der Erde darstellt. Deshalb ist es von allgemeiner Bedeutung, den Verlauf des Grundwasserflusses und den daran gekoppelten Stofftransport, z.B. von Schadstoffpartikeln, zu untersuchen. Durch ein besseres Verständnis der Transportprozesse können geeignete Gegenmaßnahmen bei Unfällen eingeleitet oder bereits stark belastete Gebiete saniert werden.

Es wurde schon früh begonnen, diese Wasserkreisläufe durch geeignete Simulationsrechnungen zu untersuchen und zu quantifizieren [SPO86]. Die verwendeten mathematische Verfahren beruhen dabei entweder auf deterministischen oder auf stochastischen Lösungsansätzen. Bei einem deterministischen Modell sind exakte Aussagen über ein zu simulierendes Gebiet möglich, wenn alle erforderlichen Parameter bekannt sind. Durch die Vielzahl der Parameter ist es aber in der Regel nicht möglich, diese vollständig anzugeben. Im Gegensatz hierzu werden die Parameter bei einem stochastischen Ansatz [DAG86], [GEL83] durch statistische Verteilungsfunktionen beschrieben. Es werden verschiedene Simulationsrechnungen durchgeführt, bei denen die Parameterwerte aus vorgegebenen Verteilungen gezogen werden. Durch Vergleich der Ergebnisse sind dann Aussagen über die Wahrscheinlichkeitsverteilungen der gesuchten Größen möglich.

Mathematisch kann die Bewegung von Wasser und der Transport von Schadstoffen im Boden durch partielle nichtlineare Differentialgleichungen beschrieben werden [HUY86]. Da eine analytische Lösung dieser Gleichungen im allgemeinen nur für einfache Fälle bekannt ist, werden numerische Verfahren herangezogen [PIN77]. Die Darstellung der räumlichen Abhängigkeit kann beispielsweise mit einem Finiten-Elemente-Modell und die zeitliche Abhängigkeit der Differentialgleichungen mit einem Finiten-Differenzen-Ansatz gelöst werden. Das zu simulierende Gebiet wird hierbei räumlich und zeitlich diskretisiert. Die einzelnen Gleichungen brauchen dann

nur noch an bestimmten Orten und zu bestimmten Zeitpunkten gelöst werden.

Da bei der Problembeschreibung sehr große Datenmengen anfallen, die die Eigenschaften des Bodens, die Art der Niederschläge und weitere für die Berechnungen wichtige Parameter charakterisieren, und da durch die numerischen Verfahren umfangreiche Gleichungssysteme entstehen, ist es bisher nur möglich, relativ kleine Gebiete mit Hilfe von Computern zu berechnen. Beschränkende Faktoren bei den Programmläufen waren die zu geringen Speicherkapazitäten und die nicht genügend hohen Geschwindigkeiten der verwendeten Maschinen. Durch die fortschreitende Entwicklung von neuen Computerarchitekturen und die weiter fallenden Preise für einzelne Hardware-Komponenten stehen nun leistungsfähigere Computer zur Verfügung. Diese Maschinen ermöglichen es, Rechnungen mit einer viel größeren Zahl von diskreten Stützstellen durchzuführen, wobei entweder eine höhere räumliche Auflösung oder eine Vergrößerung des Simulationsgebietes erlangt werden kann. Es müssen jedoch neue Verfahren angewendet werden, um die zur Verfügung stehende Rechenleistung auch nutzbar zu machen.

In der vorliegenden Arbeit wird eine Partitionierungsstrategie erarbeitet und implementiert, mit der das Programm TRACE [VER94], ein umfangreiches Simulationsprogramm aus der Hydrogeologie, auf einem massiv-parallelen Rechnersystem ausgeführt werden kann. Um möglichst viele Prozessoren effektiv einsetzen zu können, wird das Schwarz'sche Verfahren [SCH90], [BJO89], [WID92] zur Erweiterung des bisherigen sequentiellen Programms [VNL94] eingesetzt. Dieses Verfahren gestattet eine Parallelisierung von TRACE, ohne daß die grundlegende Programmstruktur verändert werden muß. Um das Programm TRACE für unterschiedliche Simulationsrechnungen nutzen zu können, wird eine automatische Gebietszerlegung in Abhängigkeit der Zahl der einzusetzenden Prozessoren und der Größe des Überlappungsbereiches durchgeführt. Der Austausch von geeigneten Informationen während der Berechnungen, hervorgerufen durch das Schwarz'sche Verfahren, ist ein weiterer wesentlicher Bestandteil dieser Arbeit.

In Kapitel 1 soll zunächst ein Überblick über parallele Rechnersysteme gegeben werden. Daran anschließend werden in Kapitel 2 die zum Verständnis der mathematischen Zusammenhänge wichtigen Verfahren vorgestellt. Eine Beschreibung der sequentiellen Programmversion von TRACE findet sich in Kapitel 3. Kapitel 4 und 5 erläutern die Strategien zur Gebietszerlegung und zur Kommunikation während der Berechnungen. Die Implementierung der Strategien soll in Kapitel 6 dargelegt werden. Die ausgewählten Simulationsrechnungen sowie die Ergebnisse der parallelen Programmausführung werden in Kapitel 7 betrachtet. Im letzten Kapitel werden die Erfahrungen dieser Arbeit zusammengefaßt und Ansatzpunkte für Erweiterungen gegeben.

Kapitel 1

Parallele Rechnersysteme

Bereits Ende der 40er Jahre ist von John von Neumann ein Modell für einen Universalrechenautomaten geschaffen worden, das bis heute hinsichtlich seiner Einfachheit und allgemeinen Anwendbarkeit nicht übertroffen worden ist [SCH81]. Diese Rechnerarchitektur besteht im wesentlichen aus nur drei Elementen. Zu nennen sind die Zentraleinheit, die ein Steuerwerk und ein Rechenwerk beinhaltet, der Arbeitsspeicher und ein E/A-System. In Abbildung 1.1 ist der Aufbau dieses Rechners schematisch dargestellt. Der von-Neumann-Rechner kann in die Klasse der sequentiellen Rechner eingeordnet werden. Der Ablauf eines Programms besteht aus Eingabe, Verarbeitung und Ausgabe (EVA-Prinzip), wobei zu beachten ist, daß der Programm-Code und die Daten gleichrangig im Arbeitsspeicher vorliegen. Historisch ist diese Architektur aus der Harvard-Architektur hervorgegangen, die über getrennte Befehls- und Datenspeicher verfügte [JUN92].

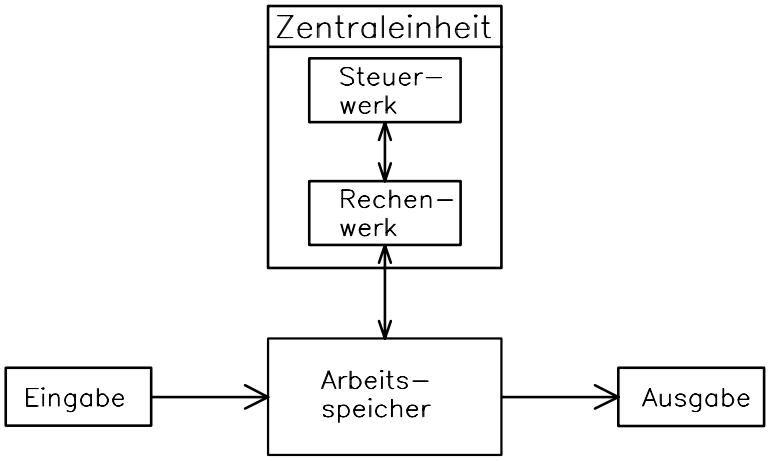


Abbildung 1.1: Aufbau eines von-Neumann-Rechners

Die von-Neumann-Architektur ist jedoch mit einem wesentlichen Nachteil verbunden. Da Daten und Befehle jeweils über einen gemeinsamen Bus zwischen Arbeitsspeicher und Zentraleinheit übertragen werden müssen, können Verzögerungen bei der Programmabarbeitung auftreten. Es kann beispielsweise nicht der nächste Befehl in die Zentraleinheit geladen werden, bevor die Daten aus der vorausgegan-

genen Rechenoperation im Arbeitsspeicher abgelegt worden sind. Diese Engstelle der rein sequentiellen Programmausführung wird als von-Neumann-Flaschenhals bezeichnet und begrenzt die Möglichkeiten einer Leistungssteigerung, da die Übertragungsraten des Bussystems immer niedriger sind als die Rechengeschwindigkeit der Zentraleinheit. Obwohl durch die fortschreitende Hardware-Entwicklung immer schnellere Prozessoren und Bussysteme geschaffen werden, wird es nicht möglich sein, diesen Engpaß zu beheben.

Durch die Einführung von Cache-Speichern können die Zugriffszeiten der Zentraleinheit auf den Arbeitsspeicher deutlich verkürzt werden. Ebenso ist durch eine überlappende Befehlsabarbeitung oder durch den Einsatz von Koprozessoren für Spezialaufgaben eine Steigerung der Leistungsfähigkeit möglich. Hierbei handelt es sich jedoch nur um Verbesserungen einzelner Systemkomponenten und nicht um eine wesentliche Änderung des klassischen Rechnermodells, so daß die Struktur der ausführbaren Programme immer noch auf einem sequentiellen Ansatz basiert. Sehr rechenintensive Aufgabenstellungen oder neue Berechnungsmethoden, beispielsweise aus der numerischen Mathematik, sind für diese konventionellen Rechner häufig zu komplex oder nicht in vernünftiger Zeit ausführbar. Um eine höhere Rechenleistung zur Verfügung stellen zu können, wurden unterschiedliche Konzepte verwirklicht. So wurden durch die konsequente Weiterentwicklung des sequentiellen Rechners unter anderem Systeme für sehr spezielle, auf den entsprechenden Rechnertyp angepaßte Programme geschaffen.

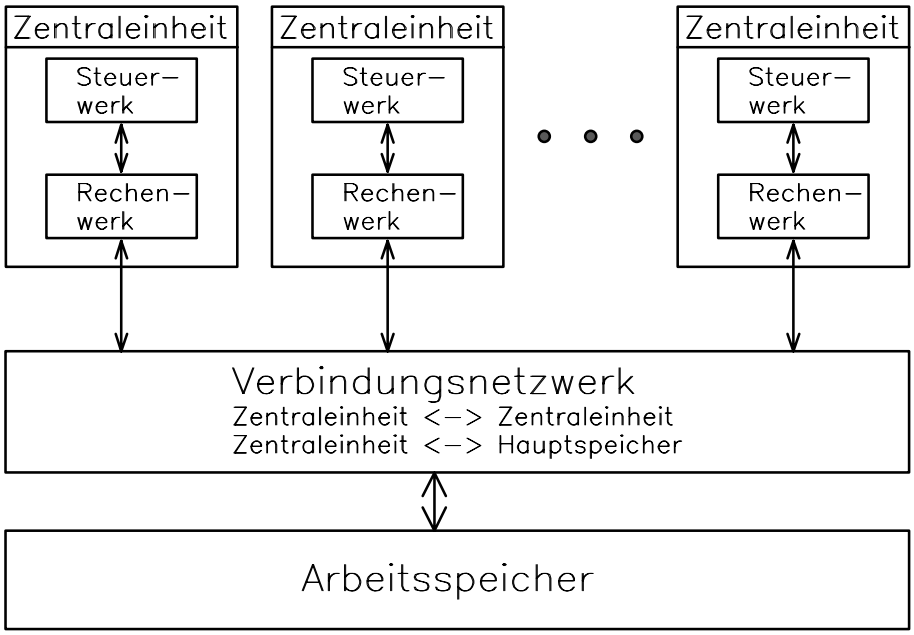


Abbildung 1.2: Prinzipieller Aufbau eines Parallelrechners

Eine andere Möglichkeit ist die Vervielfachung der Prozessorelemente, durch die der von-Neumann-Flaschenhals aufgebrochen werden kann. Eine schematische Darstellung dieser Rechnerarchitektur ist in Abbildung 1.2 aufgezeigt. Die einzelnen Prozessorelemente können den sequentiellen Rechnern entsprechen und sind deshalb

auch in großen Stückzahlen bei geringen Kosten verfügbar. Jeder Prozessor kann entweder mit lokalem oder aber mit globalem, für alle Prozessoren gemeinsamen, Arbeitsspeicher ausgestattet sein. Über ein Netzwerk sind die Prozessorelemente, die jeweils wie beim von-Neumann-Rechner aus Steuerwerk und Rechenwerk bestehen, sowohl miteinander als auch mit dem Arbeitsspeicher verbunden.

Bei einer geeigneten Aufteilung der zu lösenden Aufgabe auf die einzelnen Prozessorelemente kann diese parallel bearbeitet werden. Die mögliche Leistungssteigerung wird jedoch durch erheblich kompliziertere Rechnerarchitekturen und einen größeren Aufwand bei der Programmerstellung erschwert. Ein weiteres Problem ist je nach Art der Anwendung die Zusammenfassung und Darstellung der Einzelergebnisse der Prozessoren.

Die Entwicklung von Theorien und Algorithmen für parallele Anwendungen wird zusätzlich dadurch erschwert, daß sich bis heute kein Standardmodell für einen Parallelrechner durchgesetzt hat. Es sind jedoch verschiedene Programmiermodelle verfügbar, wie beispielsweise das Message-Passing- oder das Shared-Memory-Modell.

In vielen Bereichen der modernen Naturwissenschaften stellen Parallelrechner ein wichtiges Hilfsmittel dar. Die Lösung von umfangreichen mathematischen Gleichungssystemen, wie etwa bei der Simulation von nichtlinearen Systemen, ist mit sequentiellen Rechnersystemen kaum noch zu erreichen. Erst durch eine parallele Bearbeitung kann die Ausführungszeit so reduziert werden, daß auch rechenintensive Anwendungen gelöst werden können.

1.1 Rechnerklassifikationen

Um einen Überblick über die verschiedenen Rechnersysteme zu erlangen, werden im folgenden einige grundlegende Kriterien für deren Unterteilung betrachtet.

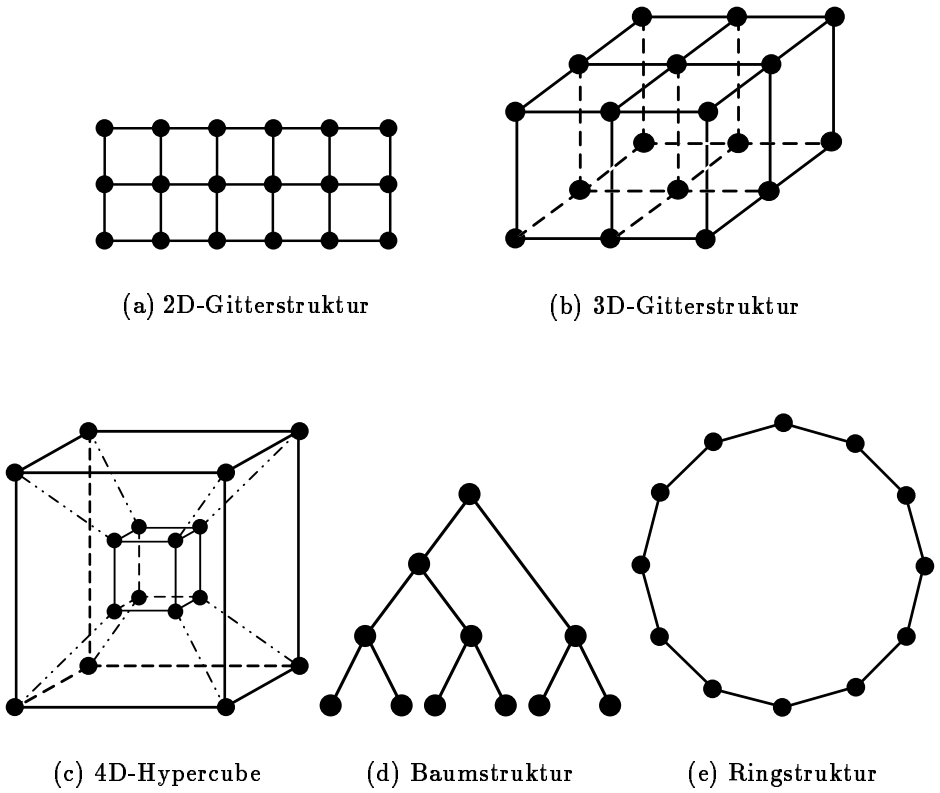
Flynn [FLY66] hat eine Klassifikation angegeben, der die Art der Daten- und Befehlsbehandlung zugrunde liegt. Hiernach lassen sich die verschiedenen Rechnerarchitekturen in die vier Klassen SISD, MISD, SIMD und MIMD unterteilen. Bei einer SISD-Struktur (Single Instruction Stream-Single Data Stream) handelt es sich um einen klassischen von-Neumann-Rechner mit einem Prozessor, bei dem ein Befehlsstrom einen Operandenstrom erzeugt. Zur Klasse der MISD-Rechner (Multiple Instruction Stream-Single Data Stream), bei der viele Prozessoren mit unterschiedlichen Befehlen dieselben Daten bearbeiten, zählen Rechner, wie sie z.B. in der Bildverarbeitung eingesetzt werden. Diese Rechner sind jedoch Spezialrechner. Interessanter sind die Klassen SIMD (Single Instruction Stream-Multiple Data Stream) und MIMD (Multiple Instruction Stream-Multiple Data Stream), da diese Klassen die parallelen beziehungsweise massiv-parallelen Rechner klassifizieren.

Grundlegend für die Struktur eines SIMD-Rechners ist ein zentraler Steuerrechner, der die Ausführung eines Programms synchron auf den einzelnen Prozessorelementen steuert. Jedes Prozessorelement führt entweder die gleiche Instruktion wie alle anderen Prozessorelemente auf seinen lokalen Daten aus oder es ist inaktiv. Zu den SIMD-Rechnern gehören Arrayrechner und Vektorrechner [BRA93]. Der Unterschied zwischen beiden Rechnertypen liegt unter anderem im Verbindungsnetzwerk

zwischen den einzelnen Prozessoren.

Rechner der MIMD-Klasse haben eine allgemeinere Architektur als SIMD-Maschinen. Jeder Prozessor besitzt seinen eigenen Befehls- und Datenfluß, den er unabhängig von den anderen Prozessoren abarbeiten kann. MIMD-Rechner arbeiten stets asynchron, über spezielle Befehle ist aber eine Synchronisation der einzelnen Prozessoren möglich.

Ein Kriterium zur Charakterisierung unterschiedlicher MIMD-Rechner ist die Topologie des Netzwerkes, welches die Prozessoren untereinander und gegebenenfalls mit dem Hauptspeicher verbindet. Die einzelnen Verbindungen können entweder statisch ("fest verdrahtet") oder dynamisch angelegt sein. In Abbildung 1.3 sind verschiedene Netzwerktopologien dargestellt, wobei die einzelnen Punkte jeweils die Prozessoren symbolisieren [GAT93], [GOL93], [MON93], [STE93].



(a) 2D-Gitterstruktur (b) 3D-Gitterstruktur

(c) 4D-Hypercube (d) Baumstruktur (e) Ringstruktur

Abbildung 1.3: Topologie unterschiedlicher Netzwerke zur Charakterisierung eines MIMD-Rechners

Grundlegend können Bus-Netzwerke, Netzwerke mit Schaltelementen und Punkt-zu-Punkt Verbindungsstrukturen unterschieden werden. Im weiteren werden jedoch nur die statischen Netzwerke in Form von Punkt-zu-Punkt Verbindungen betrachtet, da diese für das Verständis des Aufbaus eines MIMD-Rechners entscheidend sind. Ein vollständig vernetztes Netzwerk liegt dann vor, wenn jeder Prozessor mit

jedem anderen Prozessor direkt verbunden ist. Diese Netzwerke sind jedoch sehr kompliziert aufgebaut und lassen sich schlecht erweitern. Häufiger werden nicht vollständig vernetzte Netzwerke verwendet. Beispiele hierfür sind Netzwerke mit Ring- oder Torusstrukturen, Baumstrukturen, Pyramiden- oder Würfelstrukturen sowie Gitternetzwerke, bei denen die einzelnen Prozessoren entweder zwei- oder dreidimensional in einer rechtwinkligen Gitterstruktur angeordnet sind.

Welche Struktur sich am besten für den Aufbau eines Parallelrechners eignet, läßt sich nicht allgemein beantworten. Ein Vergleich zwischen Netzwerken oder eine Eignungsanalyse für ein bestimmtes Netzwerk kann eigentlich nur in Verbindung mit einer parallelen Anwendung erfolgen. Eine Anforderung, die an ein Netzwerk gestellt werden sollte, ist unter anderem, daß die Zahl der Verbindungen, die gleichzeitig zwischen mehreren Prozessorpaaren aufgebaut werden können, möglichst hoch ist, damit die Parallelverarbeitung der Prozessoren nicht durch das Verbindungsnetzwerk eingeschränkt wird. Eine Verbindungsstruktur sollte erweiterbar (skalierbar) sein, um die Rechenleistung eines Parallelrechners steigern zu können.

Ein weiteres Kriterium zur Unterteilung paralleler Rechnersysteme ist durch den Aufbau des Hauptspeichers gegeben. Es gibt Systeme mit gemeinsamem Speicher (shared memory) und mit verteiltem Speicher (distributed memory). Bei "shared-memory"-Rechnern haben alle Prozessoren Zugriff auf einen gemeinsamen Speicherbereich. Jeder Prozessor kann zusätzlich auch einen eigenen lokalen Speicher für seinen Programm-Code oder Zwischenergebnisse besitzen [GOL93]. Synchronisation und Datenaustausch erfolgen über Speicherbereiche, auf die koordiniert zugegriffen werden kann. Rechner mit verteiltem Speicher haben keinen gemeinsam von allen Prozessoren nutzbaren Speicherbereich. Jeder Prozessor verfügt über seinen eigenen, lokalen Speicher. Ein direkter Zugriff auf die Daten eines anderen Prozessors ist nicht möglich. Die Prozessoren in einem "distributed-memory"-Rechner entsprechen logisch einem Verbund unabhängiger Rechner, die über ein Netzwerk miteinander gekoppelt sind. Kommunikation und Synchronisation zwischen den Prozessoren erfolgen durch den Austausch von Nachrichten, die über das verbindende Netzwerk verschickt werden müssen. Ein Vorteil dieser Architektur ist die zumindest theoretisch unbegrenzte Erweiterbarkeit solcher Systeme, falls ein Verbindungsnetzwerk mit einer Punkt-zu-Punkt Struktur verwendet wird.

Ein umfangreicher Überblick über die zur Zeit verfügbaren parallelen Rechnersysteme und ein Vergleich der Performance dieser Systeme ist in [DON94], [GAT93] und [STE93] angegeben.

1.2 Programmierung eines MIMD-Rechners

Gegenüber einem System mit nur einem einzigen Prozessor besitzt ein n-Prozessorsystem eine potentiell n-fach höhere Leistung. Um diese Leistung nutzen zu können, müssen jedoch bei der Programmierung einige grundlegende Regeln beachtet werden. Alle Prozessoren, die zur Ausführung des Programms eingesetzt werden, sollten auch über die gesamte Programmlaufzeit genutzt werden. Dies läßt sich schon nicht mehr gewährleisten, wenn ein Prozessor auf die Ergebnisse eines anderen Prozessors

wartet, dieser aber mit seinen Berechnungen noch nicht fertig ist. Die Bereitstellungszeiten für Daten, die nicht lokal verfügbar sind, müssen ebenfalls berücksichtigt werden. Der zusätzliche Aufwand zur Synchronisation der einzelnen Prozessoren sollte möglichst gering sein. Hieraus ergibt sich, daß die Teilaufgaben, die die einzelnen Prozessoren bearbeiten, weitgehend unabhängig voneinander sein sollten. Der Aufwand, hervorgerufen durch die Datenbereitstellung für die Prozessoren, sollte ebenfalls minimal sein. Wenn beispielsweise für den weiteren Programmablauf wichtige Felder berechnet werden müssen, kann es vorteilhaft sein, daß diese Berechnungen von allen Prozessoren lokal durchgeführt werden, anstatt daß ein Prozessor die Arbeit allein ausführt und seine Ergebnisse den anderen Prozessoren durch Kommunikation mitteilt. Durch geeignete Überlappung von Rechnungs- und Kommunikationsteilen in einem Programm kann der Kommunikationsaufwand verborgen und die damit verbundenen Wartezeiten minimiert werden. An dieser Stelle soll aber darauf hingewiesen werden, daß es in den meisten Fällen nicht möglich ist, die bei einem Parallelrechner potentiell zur Verfügung stehende Leistung vollständig auszunutzen.

Ein paralleles Programm sollte ohne Änderungen des Codes auf einer unterschiedlichen Zahl von Prozessoren ausgeführt werden können. Um dies zu ermöglichen, muß die Verteilung der Aufgaben auf die einzelnen Prozessoren so ausgelegt sein, daß sie nicht statisch vorgegeben, sondern erst während des Programmstarts flexibel durchgeführt wird. Der Ansatz der Datenparallelität ist für eine Programmausführung bei unterschiedlichen Prozessorzahlen geeignet, da hierbei gleiche Programmabschnitte auf unterschiedlichen Datensegmenten parallel ausgeführt werden können und die Datenaufteilung in derartige Segmente leicht beschrieben werden kann. Dieser Ansatz unterstützt auch die asynchrone Parallelität eines MIMD-Rechners, bei dem, wie in Abschnitt 1.1 beschrieben, die einzelnen Prozessoren unabhängig voneinander Teilaufgaben lösen können. Die Prozesse, die dann auf den Prozessoren bearbeitet werden, sind "sequentielle" Programme, die jedoch zur Kooperation bestimmte, architekturabhängige Befehle wie beispielsweise Sende- oder Empfangsbefehle benötigen. Sind die Teilaufgaben nicht völlig unabhängig voneinander, dann müssen die Prozesse untereinander Daten austauschen.

Grundsätzlich lassen sich verschiedene Programmiermodelle unterscheiden, mit deren Hilfe Programme für parallele Rechnerarchitekturen geschrieben werden können. Zu nennen sind unter anderem die Modelle *Shared Virtual Memory* (SVM) [BER93], [LI86], *High Performance Fortran* (HPF) [KOE94] und *Message Passing*. Hier soll jedoch nur die Struktur von Programmen nach dem Message-Passing-Ansatz betrachtet werden. Wird bei einem parallelen Programm eine Kommunikation zwischen verschiedenen Prozessoren erforderlich, so wird diese direkt durch den Austausch von Nachrichten durchgeführt. Hierbei werden die auszutauschenden Daten in einem "Paket" zusammengefaßt, das Paket wird mit einer Adresse, der Nummer des Zielprozessors, und einer eindeutigen Identifikationsnummer versehen, und dann über das die Prozessoren verbindende Netzwerk verschickt. Dieser Austausch kann im allgemeinen sowohl synchron als auch asynchron durchgeführt werden. Bei einer synchronen (blockierenden) Kommunikation müssen der sendende und der empfangende Prozeß gemeinsam einen Synchronisationspunkt erreichen, um

danach ausschließlich zu kommunizieren. Bei der asynchronen (nicht-blockierenden) Kommunikation kann der sendende Prozeß seine Nachricht verschicken, ohne daß der empfangende Prozeß diese Nachricht sofort entgegennehmen muß. Damit ein Programm effizient ausgeführt werden kann, sollten die Teilaufgaben, die jeder Prozessor zu lösen hat, weitgehend unabhängig voneinander bearbeitet werden können. Das Aufteilen zu kleiner Aufgaben, wie etwa arithmetischer Ausdrücke, erfordert durch die sehr oft auftretende Kommunikation hohe Synchronisationskosten und würde den durch die parallele Programmausführung hervorgerufenen Gewinn zunichte machen.

Bezüglich der Größe der ohne Kommunikationsanforderung ausführbaren Programmabschnitte wird qualitativ zwischen fein- und grobgranularen Prozessen unterschieden. Zur Bestimmung der Granularität eines Prozesses wird das Verhältnis der Menge der Berechnungen zur Zahl der Kommunikationsereignisse verwendet. Ein kleines Verhältnis entspricht einem feingranularen und ein großes Verhältnis einem grobgranularen Prozeß [IBB89]. Bei feingranularen Prozessen werden sehr häufig Synchronisationen durchgeführt, während bei grobgranularen Prozessen eine signifikante Zahl von Berechnungen zwischen den Kommunikationsereignissen auftreten kann. Die Granularität ermöglicht jedoch keine Aussagen über die Menge des Programm-Codes oder die zu erwartende Programmlaufzeit, da die relative Häufigkeit von Kommunikationsereignissen nicht notwendigerweise von ihr abhängig ist. Da der Zeitbedarf für eine Kommunikation zwischen verschiedenen Prozessoren in der Regel höher ist als die Dauer der Ausführung von arithmetischen oder logischen Operationen und lokalen Speicherzugriffen, wird auf Mehrprozessorsystemen im allgemeinen die Verwendung von grobgranularen Prozessen angestrebt.

Zur Charakterisierung eines parallelen Programms können verschiedene Kriterien verwendet werden. So sollte zum Beispiel das in [BUR93] erwähnte "Volume-to-Surface"-Verhältnis (Gleichung 1.1) möglichst groß sein, da dieser Quotient ein Maß für den Kommunikationsaufwand bei der Verarbeitung von Daten angibt.

$$V = \frac{Berechnungszeit}{Kommunikationszeit} \tag{1.1}$$

Ein weiteres Kriterium für die Beurteilung eines parallelen Programms ist die Lastverteilung (Load-Balancing). Es wird eine möglichst gute Ausnutzung der Gesamtheit der eingesetzten Prozessoren angestrebt. Die Aufgabe, die jeder Prozessor zu lösen hat, sollte in der Komplexität und in der Bearbeitungsdauer in etwa so groß sein wie die Aufgaben der anderen Prozessoren. Häufig kann der Arbeitsaufwand einzelner Prozesse jedoch nicht a priori abgeschätzt werden, so daß die Verteilung auf die einzelnen Prozessoren erst während der Programmlaufzeit erfolgen kann. Ebenso kann es vorkommen, daß sich der Aufwand für einzelne Prozesse während der Laufzeit ändert, wobei dann keine ausgeglichene Lastverteilung mehr vorliegen muß. In diesen Fällen ist es vorteilhaft, wenn eine neue Aufgabenverteilung durchgeführt wird. Ein solches Vorgehen erweist sich jedoch als schwierig, obwohl schon einige Ansätze für eine dynamische Lastverteilung existieren (*Simulated Annealing* [FOX88]).

Die Programmierung eines MIMD-Rechners erweist sich als bedeutend schwie-

riger als das Erstellen von sequentiellen Programmen. Wenn beispielsweise bei der Prozeß-Synchronisation Programmierfehler auftreten sollten, sind diese in der Regel nicht sofort zu erkennen. Auch Probleme mit inkonsistenter Datenhaltung oder sogenannte "Deadlock"-Situationen sind möglich. Eine weitere Schwierigkeit ergibt sich dadurch, daß das Verhalten eines parallelen Programms trotz gleicher Daten bei jedem Programmlauf unterschiedlich sein kann. Unterschiede im Zeitverhalten bei der Kommunikation zwischen den Prozessoren können dabei die Reihenfolge nachfolgender Aktionen beeinflußen. Ein weiteres Problem liegt darin, daß es nur wenige Hilfsmittel für die Programmierung gibt und diese nicht auf allen Rechnersystemen verfügbar sind. Durch die Komplexität einer parallelen Programmausführung ist der Einsatz von Debuggern erheblich erschwert.

1.3 Portabilität

Zur Zeit hat sich aus der Vielzahl der möglichen parallelen Rechnerarchitekturen noch keine dominierende Architektur herausgebildet. Deshalb sollte ein paralleles Programm portabel sein, damit es auf unterschiedlichen Systemen ausgeführt werden kann. Zwischen der Effizienz und der Portabilität eines Programms muß häufig ein Kompromiß gesucht werden, da sich beide Anforderungen in der Regel widersprechen. Je maschinennäher ein Programm realisiert ist, desto höher wird die Effizienz bei gleichzeitiger Abnahme der Portabilität sein. Die Ausführbarkeit von Programmen auf unterschiedlichen Rechnern bedeutet aber nicht, daß diese Programme auf allen Rechnern mit der gleichen Effizienz laufen. Eine individuelle Anpassung an bestimmte Besonderheiten eines Systems kann deshalb durchaus sinnvoll sein. Auch die Programmiersprache, in der ein Programm implementiert wurde, ist für dessen Portabilität entscheidend. So sind gerade auf neu entwickelten Rechnern in der Regel nur Compiler für wenige Sprachen verfügbar. Meistens sind dies FORTRAN und C.

Bei der Programmierung eines MIMD-Systems sind neben den schon erwähnten Kriterien auch die maschinenabhängigen Befehle, die in Programme eingebunden werden müssen, zu beachten. Zu nennen sind hier besonders Befehle zur Synchronisation der einzelnen Prozessoren, zum Versenden und Empfangen von Daten und zur Abfrage von bestimmten Systemparametern, wie beispielsweise der Anzahl der eingesetzten Prozessoren, bestimmter Prozessornummern oder Variablen, die Laufzeiten angeben. Da bislang jeder Hersteller seine eigenen Routinen entwickelte, gibt es schon seit längerer Zeit Bemühungen, hier einen Standard zu definieren. So sind herstellerunabhängige Bibliotheken entwickelt worden, die standardisierte Befehle auf die entsprechende Hardware abbilden. Eingesetzt werden zur Zeit unter anderem die Software-Systeme PVM, PARMACS und MPI, die auf einer großen Zahl unterschiedlicher Maschinen, von Workstations bis zu massiv-parallelen Systemen, verfügbar sind. Auch eine Kopplung verschiedener Systeme zu einem heterogenen Rechnersystem auf Basis dieser "Message-Passing"-Bibliotheken ist möglich.

1.4 Begriffe zur Leistungsbewertung

Bei der Verwendung von Mehrprozessorsystemen ist der Performance-Gewinn, der durch eine parallele Programmausführung im Vergleich zur sequentiellen Version erzielt werden kann, eine bedeutende Größe. Im folgenden sollen mehrere Kriterien vorgestellt werden, die eine Beurteilung eines parallelen Programms ermöglichen.

Der Speedup S_p gibt den Geschwindigkeitszuwachs an, den ein Parallelrechner, bei dem p Prozessoren genutzt werden, gegenüber einem sequentiellen Rechner mit nur einem Prozessor für die Lösung eines numerischen Problems liefert. Eine Definition dieser Größe hat Amdahl bereits 1967 bei Leistungsbetrachtungen für Parallelrechner [AMD67] angegeben, auch wenn er noch nicht den Begriff Speedup benutzte. Er hat eine Gleichung aufgestellt, in der nur der sequentielle und der parallele Anteil eines Programms als Parameter berücksichtigt wird. Ein Einfluß der Problemgröße ist nicht vorgesehen. Durch dieses Gesetz (Gleichung 1.2) wird eine theoretische Grenze für den maximal zu erwartenden Geschwindigkeitszuwachs vorgegeben.

$$S_p = \frac{T_1}{T_p} = \frac{1}{(1-f) + f/p} \tag{1.2}$$

S_p gibt den maximal zu erwartenden Speedup an. T_1 entspricht der Laufzeit des Programms für einen Prozessor, während T_p die Laufzeit für p Prozessoren angibt. Der Faktor f entspricht dem Anteil der Laufzeit des sequentiellen Programms, der parallel ausgeführt werden kann. Der Wert für f muß zwischen null und eins liegen, entsprechend einer parallel ausführbaren Programmlaufzeit zwischen null und einhundert Prozent. Der maximal mögliche Speedup entspricht der Zahl der Prozessoren p, was bedeutet, daß die parallele Programmversion nur $1/p$ der Zeit der sequentiellen Version benötigt. Aus dem Amdahl'schen Gesetz folgt, daß ein Programm, welches 99 % parallelisierbaren Code aufweist, unabhängig von der Zahl der eingesetzten Prozessoren maximal um den Faktor 100 beschleunigt werden kann. Danach wäre bei Programmen mit großem sequentiellen Anteil (> 50 %) der Einsatz von Parallelrechnern nicht sinnvoll.

Es gibt eine Vielzahl von Gründen, die einen direkten Vergleich eines sequentiellen mit einem parallelen Rechnern erschweren. So kann es sein, daß ein für einen sequentiellen Rechner sehr gut geeigneter Algorithmus nicht effizient auf einen Parallelrechner übertragen werden kann. Auch ein Software-Overhead bei Parallelrechnern, der beispielsweise durch Initialisierungen oder zusätzliche Indexberechnungen hervorgerufen wird, beeinträchtigt einen Vergleich mit einem sequentiellen Rechner. Zudem muß beachtet werden, daß für den Speedup die Gesamtlaufzeit des Programms zugrunde gelegt wird, auch wenn der größte Teil der eingesetzten Prozessoren schon längere Zeit mit seiner Aufgabe fertig ist. Hier läßt sich der Einfluß einer optimalen Lastverteilung auf die Laufzeiten der einzelnen Prozessoren erkennen. Für die Kommunikation muß ebenfalls ein Overhead für die Bereitstellung der entsprechenden Verbindungen eingerechnet werden. Ein weiteres Problem liegt darin, daß die Berechnungsdaten eines parallelen Programms oft nicht mehr auf einem sequentiellen Rechner geladen werden können, so daß eine genaue Bestimmung des Wertes T_1 nicht möglich ist.

Bei Programmläufen auf Parallelrechnern hat sich jedoch gezeigt, daß auch bei wachsender Prozessorzahl noch mit einem akzeptablen Speedup zu rechnen ist. Ein wichtiger Faktor, der im Amdahl'schen Gesetz nicht berücksichtigt wird, ist die Problemgröße, da in der Regel der sequentielle Anteil eines Programms bei wachsender Problemgröße relativ abnimmt. Amdahl hat die Abhängigkeit des Faktors f von der Problemgröße nicht betrachtet, so daß die relative Abnahme des sequentiellen Programmteils nicht berücksichtigt wird. Um den Zuwachs der Problemgröße bei fester Rechenzeit angeben zu können, wird häufig ein Scaleup-Faktor angegeben. Gleichung 1.3 nennt eine Definition für den Scaleup [BRA93]. Hierbei gibt $T_i(j)$ die Ausführungszeit eines Programms mit der Problemgröße j an, wenn es auf i Prozessoren berechnet wurde.

$$S_c = \frac{n}{m} \qquad \text{mit } T_1(m) = T_k(n) \tag{1.3}$$

Der Scaleup S_c beschreibt den Skalierungsgewinn bei der Lösung eines Problems der Größe n, welches mit k Prozessoren berechnet wurde, im Verhältnis zu einem kleineren Problem der Größe m, mit $m < n$, welches auf nur einem Prozessor berechnet wurde. Die Ausführungszeiten T_1 und T_k hängen nur von der Zahl k der eingesetzten Prozessoren und der noch nicht näher definierten Problemgröße ab. Die Problemgröße könnte beispielsweise durch die Zahl der Gitterpunkte einer Diskretisierungsmethode angeben werden, die mit den beiden Programmläufen berechnet werden soll.

Gustafson [GUS88] stellt eine Formel für den skalierten Speedup $ScSP$ vor. Gleichung 1.4 gibt diese Formel an.

$$ScSP = \frac{T_{seq}}{T_{par}} = \frac{T_s + p \cdot T_p}{T_s + T_p} = 1 + f \cdot (p - 1) \tag{1.4}$$

In dieser Gleichung wird das Verhältnis zwischen dem Zeitverhalten auf einem sequentiellen Rechner und einem Parallelrechner mit p Prozessoren bestimmt. Mit T_s wird die Zeit für die Bearbeitung des Anteils des Programms bezeichnet, der nicht parallelisiert werden kann, während T_p die Zeit für den Programmteil angibt, der parallel ausgeführt werden kann. Eine Einschränkung ist sicherlich, daß der sequentielle Anteil bei der Skalierung konstant bleibt und sich nur der parallele Anteil vergrößert. f entspricht wieder dem Anteil des Programms, der parallel ausgeführt werden kann (siehe Gleichung 1.2).

Eine weitere Definition für den Speedup wird von Moler [PAT89] angegeben. Dieses Gesetz ist eine Erweiterung des Amdahl'schen Gesetzes in der Beziehung, daß Parallelismus bei hinreichend großen Berechnungen einen vernünftigen Speedup ergeben kann. Gleichung 1.5 gibt die Zeit an, die eine parallele Berechnung mit p Prozessoren benötigt, und Gleichung 1.6 entspricht der Größe des Speedups.

$$T_p = \frac{1 - (f - 1) \cdot T_1}{p} + (1 - f) \cdot T_1 \tag{1.5}$$

$$S_p = \frac{p}{1 + (p - 1) \cdot (1 - f)} \tag{1.6}$$

T_1 gibt die Laufzeit des Teils des Programms, welcher nicht parallelisiert werden kann. f entspricht der Definition beim Gesetz von Amdahl, jedoch nimmt Moler an, daß f eine Funktion der Problemgröße n ist. Bei einem effektiven parallelen Programm gilt nach Moler, daß für $n \to \infty$ auch $f \to 0$ gilt, was bedeutet, daß der sequentielle Teil eines Programms bei hinreichend großer Problemgröße vernachlässigt werden kann. Für $p > 1$ gilt für den Speedup nach Moler immer $S_p \leq \frac{1}{1-f}$. Der maximal zu erreichende Speedup bei Programmen, die auch einen sequentiellen Anteil besitzen können, ist – im Unterschied zu Amdahl – immer p, falls ein effektiver paralleler Algorithmus gewählt wurde.

Als Maß für den erreichten Speedup einer Anwendung relativ zum maximal möglichen Speedup wird auch häufig der Begriff der Effizienz E benutzt. In Formel 1.7 wird die Effizienz mit Hilfe der Größen S_p für den erreichten Speedup und p für die Zahl der Prozessoren angegeben.

$$E = \frac{S_p}{p} \tag{1.7}$$

Die Aussagekraft von Zahlenwerten für die oben beschriebenen Kriterien ist nur mit gewissen Einschränkungen zu verallgemeinern. Die Werte für Speedup und Scaleup sind immer anwendungsbezogen und können nur sehr bedingt auf andere Problemstellungen übertragen werden. Weitergehende Betrachtungen zur Leistungsanalyse können unter anderem in [FOX88] und [PAT89] nachgelesen werden.

1.5 Die Intel Paragon XP/S 10

Die Intel Paragon ist ein skalierbarer Parallelrechner der Firma Intel Supercomputer Systems Division, der seit September 1992 ausgeliefert wird. Dieser Rechner kann entsprechend Abschnitt 1.1 zur MIMD-Rechnerklasse gezählt werden, wobei er über verteilten Arbeitsspeicher und eine *Message-Passing*-Architektur [BOE94] verfügt. Die Paragon ist eine Weiterentwicklung der Systeme iPSC/860 und Touchstone Delta.

Die Knoten (*nodes*) dieser Maschine basieren auf einem oder mehreren i860 RISC-Prozessoren von Intel. Untereinander sind sie mit einem zweidimensionalen, rechtwinkligen Gitternetzwerk verbunden. Jeder Knoten besitzt mindestens 16 MB Arbeitsspeicher [INT93]. Generell lassen sich die Knoten in drei Gruppen mit unterschiedlichen Aufgaben einteilen: *compute nodes*, *service nodes* und *I/O nodes*. Die *compute nodes* werden zur parallelen Programmausführung benötigt, die *service nodes* stellen dem Anwender das Betriebssystem (OSF/1) und einige Werkzeuge zur Programmentwicklung zur Verfügung, und die *I/O nodes* bilden die Schnittstelle zur Peripherie wie beispielsweise Massenspeichern oder externen Netzwerken. Das die Knoten verbindende Netzwerk besitzt eine Übertragungsrate von bis zu 175 MB/s in jeder Richtung [BOE94]. Die Kommunikation zwischen zwei Prozessoren wird mit der *Wormhole-Routing*-Strategie mit einem deterministischen Algorithmus durchgeführt.

Für die Paragon werden von Intel oder Drittanbietern verschiedene Programmiermodelle angeboten. Neben dem *Message-Passing*-Programmiermodell sind ein *Data-Parallel*-Programmiermodell mit High Performance Fortran (HPF) und ein *Shared-Memory*-Programmiermodell mit Shared Virtual Memory (SVM) verfügbar [ESS93]. Theoretisch können auf jedem Knoten mehrere Prozesse ausgeführt werden, die sich mit einem *Time-Sharing*-Verfahren die Rechenleistung des Prozessors teilen.

Die bei der vorliegenden Arbeit eingesetzte Intel Paragon XP/S 10 ist im Zentralinstitut für Angewandte Mathematik des Forschungszentrum Jülich installiert und verfügt über 138 *compute nodes*, 6 *service nodes* und 7 *I/O nodes*. Eine ausführliche Beschreibung der Intel Paragon ist im Handbuch des Herstellers [INT93] und in [ESS93] gegeben.

Kapitel 2

Mathematische Grundlagen

In diesem Kapitel soll zunächst die im Programm TRACE verwendete Finite-Elemente-Methode beschrieben werden, da durch diese Methode das diskrete Gitter, welches bei der Parallelisierung des Programms zerlegt werden muß, definiert wird. Daran anschließend wird das Schwarz'sche Verfahren vorgestellt. Bei diesem Verfahren können durch eine Zerlegung des Gesamtgitters in mehrere sich überlappende Teilgitter die numerischen Lösungen für die Teilgitter parallel berechnet werden. Durch Kombination der Teillösungen wird dann die Lösung für das Gesamtgitter angegeben.

2.1 Die Finite-Elemente-Methode

Bei der mathematischen Beschreibung physikalischer Zusammenhänge werden sehr oft Differentialgleichungen verwendet, die nicht mehr analytisch gelöst werden können. Durch die verfügbaren Rechnersysteme können die Lösungen dieser Gleichungen jedoch numerisch bestimmt werden. Verfahren, die hierzu verwendet werden, sind die Finite-Elemente-Methode (FEM) und die Finite-Differenzen-Methode (FDM). Der grundlegende Unterschied zwischen FEM und FDM ist, daß bei FEM die Lösungsfunktion direkt und bei FDM die Ableitung der Lösungsfunktion approximiert wird. Daraus ergibt sich, daß FEM räumlich kontinuierliche Lösungen liefert, während FDM nur Lösungen für diskrete Punkte angeben kann [MAR89]. FEM besitzt mehrere Vorteile gegenüber FDM. So können beispielsweise irreguläre Gebietsberandungen ohne zusätzliche Formeln angegeben werden oder in einem Gitter fein- und grobaufgelöste Diskretisierungsgebiete gleichzeitig auftreten. Im Programm TRACE wird die zeitliche Abhängigkeit der Modellgleichungen durch eine Finite-Differenzen-Methode und die räumliche Abhängigkeit durch eine Finite-Elemente-Methode ausgedrückt. Da das Finite-Elemente-Gitter bei der Parallelisierung von TRACE zerlegt werden muß, soll die Finite-Elemente-Methode, die im Programm TRACE mit einem Ansatz von Galerkin Anwendung findet, vorgestellt werden.

Bei der Finiten-Elemente-Methode wird der Definitionsbereich der zu approximierenden Funktion in nicht überlappende Teilbereiche zerlegt, deren Formen weitgehend beliebig gewählt werden können. Diese Teilbereiche werden finite Elemente

genannt. Ein Element kann durch Punkte aus dem Definitionsbereich beschrieben werden. Um beispielsweise ein dreieckiges Element anzugeben, genügen drei Gitterpunkte. Für jedes finite Element wird, entsprechend der dem Element zugeordneten Zahl der Gitterpunkte, eine analytische Näherungslösung des gegebenen Problems mit unbestimmten Koeffizienten angesetzt. Die während der Rechnung zunächst noch unbekannten Werte für die Koeffizienten differieren im allgemeinen von Element zu Element und werden so bestimmt, daß eine möglichst gute Näherungslösung resultiert [MAR89].

Die Finite-Elemente-Methode soll im folgenden exemplarisch für eine einfache lineare Differentialgleichung beschrieben werden. Es wird die Poisson-Gleichung (Gleichung 2.1) betrachtet, die in der Elektrotechnik oft zum Berechnen von elektrischen beziehungsweise magnetischen, statischen Feldern verwendet wird.

$$-\Delta u = f \tag{2.1}$$

In dieser Gleichung sollen u und f jeweils ortsabhängige Funktionen sein mit $u = u(x,y,z)$ und $f = f(x,y,z)$. Weiterhin wird gefordert, daß $u, f \in V$ gilt, also Funktionen entsprechen, die in einem geeignet gewählten Vektorraum definiert sein sollen. Es wird nun eine Lösung dieser Gleichung für ein vorgegebenes dreidimensionales Gebiet $\mathcal{R}$, berandet durch $\partial\mathcal{R}$ gesucht. Eine Multiplikation der Poisson-Gleichung mit einer beliebigen Wichtungsfunktion $v \in V = H_0^1$, also einer Funktion, die auf dem Rand des Gebietes verschwindet und einmal schwach differenzierbar ist, und anschließende Integration führt auf Gleichung 2.2. Diese Gleichung soll für alle Funktionen v gelten und ist äquivalent zu der Ausgangsgleichung 2.1.

$$\int_{\mathcal{R}} -\Delta u \cdot v \, d\Omega = \int_{\mathcal{R}} f \cdot v \, d\Omega \qquad \forall v \in V \tag{2.2}$$

Durch Anwenden der 1. Green'schen Formel (einmalige partielle Integration), angegeben in Gleichung 2.3, auf das linke Integral in Gleichung 2.2 und gleichzeitigem Ausnutzen der Information, daß alle Funktionen v auf dem Rand immer identisch Null sind, folgt Gleichung 2.4.

$$\int_{\mathcal{R}} \Delta U_1 \cdot U_2 \, d\Omega = \int_{\partial\mathcal{R}} U_2 \cdot \nabla U_1 \, d\mathbf{S} - \int_{\mathcal{R}} \nabla U_1 \cdot \nabla U_2 \, d\Omega \tag{2.3}$$

Die Integralgleichung 2.4 wird schwaches Poisson-Problem in bilinearer Form genannt. Wenn eine Funktion h in bilinearer Form vorliegt, dann gilt $h(u_1 + u_2, v) = h(u_1, v) + h(u_2, v)$, beziehungsweise das entsprechende für die Variable v.

$$\int_{\mathcal{R}} \nabla u \cdot \nabla v \, d\Omega = \int_{\mathcal{R}} f \cdot v \, d\Omega \qquad \forall v \in V \tag{2.4}$$

Wenn in Gleichung 2.4 die linke Seite $a(u,v)$ und die rechte Seite $f(v)$ genannt wird, dann läßt sich die Aufgabenstellung umformulieren. Zu lösen ist nun Gleichung 2.5. Diese Gleichung muß wiederum für alle v erfüllt sein; die gesuchte Lösungsfunktion ist erneut u.

$$a(u,v) = f(v) \qquad \forall v \in V \tag{2.5}$$

Die Lösungsfunktion u wird nun durch einen Polygonzug nachgebildet. Hierzu wird das Gebiet $\mathcal{R}$ in n nicht-überlappende Teilgebiete $\mathcal{R}_i$ mit $i = 1 \ldots n$ zerlegt. Falls die Funktion v Unstetigkeiten aufweist, müssen die Grenzen der Teilgebiete so gewählt werden, daß die Unstetigkeiten jeweils auf den Rändern der Teilgebiete liegen. In Abbildung 2.1 ist diese Zerlegung und die Nachbildung durch einen Polygonzug für eine eindimensionale Diskretisierung aufgezeigt. Die exakte Lösung der Differentialgleichung soll durch die Funktion u und die Näherungslösung durch die Funktion u_h angedeutet sein. Mathematisch betrachtet wird die Aufgabenstellung in einen anderen Raum transformiert. Dieser Raum ist ein n-dimensionaler Vektorraum V_h, der durch sogenannte Basisfunktionen φ_i mit $i = 1, \ldots, n$ aufgespannt wird. Jedem der n Teilgebiete, die auch Elemente genannt werden, wird nun eine Basisfunktion zugeordnet. Die Konstruktion der Basisfunktionen ist beliebig, jedoch werden in der Praxis häufig Funktionen benutzt, die entsprechend Abbildung 2.1 aufgebaut sind. Warum diese Form sehr günstig für die numerische Lösung ist, soll später erläutert werden.

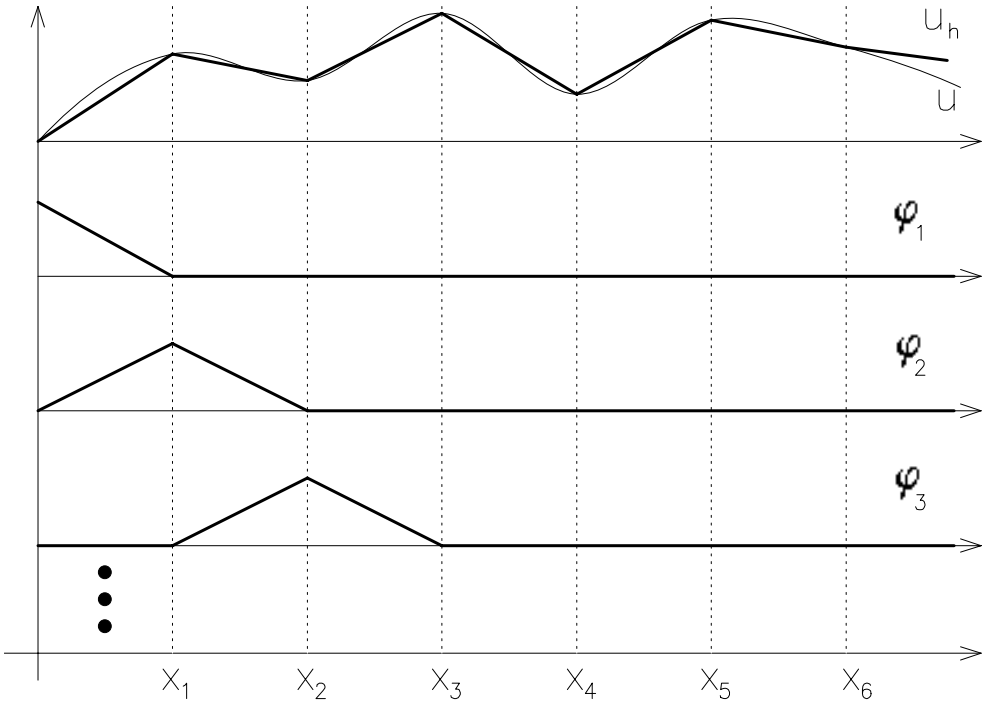


Abbildung 2.1: Lösungsfunktion u_h und Basisfunktionen φ_i bei eindimensionaler Diskretisierung

Die Näherung u_h für die Lösungsfunktion u läßt sich schreiben als Überlagerung der einzelnen, mit einem Koeffizienten α_i multiplizierten, Basisfunktionen φ_i.

$$u_h = \sum_{i=1}^{n} \alpha_i \cdot \varphi_i \tag{2.6}$$

Die Wichtungsfunktionen v können entsprechend Gleichung 2.6 ebenfalls durch eine Summe dargestellt werden.

$$v_h = \sum_{j=1}^{n} \beta_j \cdot \varphi_j \tag{2.7}$$

Die Aufgabenstellung läßt sich erneut umformulieren. Gesucht ist nun ein $u_h \in V_h$, so daß für alle $v_h \in V_h$ gilt:

$$a(u_h, v_h) = f(v_h) \qquad \forall v_h \in V_h. \tag{2.8}$$

Einsetzen der Summe für v_h in Gleichung 2.8 führt zu Gleichung 2.9.

$$a(u_h, \sum_{j=1}^{n} \beta_j \cdot \varphi_j) = f(\sum_{j=1}^{n} \beta_j \cdot \varphi_j) \tag{2.9}$$

Indem die Bilinearität der Funktion a ausgenutzt wird, kann gezeigt werden, daß es genügt, die Gleichung 2.10 für alle Basisfunktionen φ_i, φ_j zu lösen, um die Funktion u_h zu berechnen.

$$a(u_h, \varphi_j) = f(\varphi_j) \qquad \forall j = 1 \ldots n \tag{2.10}$$

Durch Einsetzen der Summe für u_h kann Gleichung 2.11 angegeben werden. Die Aufgabenstellung lautet nun, diese Gleichung nach α_i für alle Funktionen φ aufzulösen.

$$a(\sum_{i=1}^{n} \alpha_i \cdot \varphi_i, \varphi_j) = f(\varphi_j) \qquad \forall j = 1 \ldots n \tag{2.11}$$

Durch Ausnutzen der Bilinearitätsbedingung für a kann die Summe aus der Funktion a herausgezogen werden (Gleichung 2.12).

$$\sum_{i=1}^{n} \alpha_i \cdot a(\varphi_i, \varphi_j) = f(\varphi_j) \qquad \forall j = 1 \ldots n \tag{2.12}$$

Die Funktion $a(\varphi_i, \varphi_j)$ soll im folgenden mit a_{ij} bezeichnet werden. Aus Symmetrieüberlegungen folgt sofort, daß $a_{ij} = a_{ji}$ gelten muß. Die Funktion $f(\varphi_j)$ soll durch f_j angegeben werden.

$$\sum_{i=1}^{n} \alpha_i \cdot a_{ij} = f_j \qquad \forall j = 1 \ldots n \tag{2.13}$$

Gleichung 2.13 kann als Matrixgleichung geschrieben werden.

$$\underline{\mathbf{A}} \cdot \vec{x} = \vec{b} \tag{2.14}$$

In Gleichung 2.14 entspricht $\underline{\mathbf{A}}$ einer [n x n]-Matrix, in der die einzelnen a_{ij} eingetragen sind. Diese Matrix wird oft Steifigkeitsmatrix genannt. Da $a_{ij} = a_{ji}$ gelten soll, ist die Transponierte der Matrix $\underline{\mathbf{A}}$ gleich der Matrix $\underline{\mathbf{A}}$. Der Vektor $\vec{x}$ hat

die Dimension [n x 1] und besitzt die Elemente $\alpha_1 \ldots \alpha_n$, während der Vektor $\vec{b}$ die Dimension [n x 1] hat und die Einträge $f_1 \ldots f_n$ hat.

Die Aufgabenstellung lautet nun, die Matrixgleichung 2.14 nach $\vec{x}$ aufzulösen. Dies entspricht einem linearen Gleichungssystem mit n Gleichungen und n Unbekannten. Für eine numerische Lösung dieser Gleichungen bieten sich verschiedene wohlbekannte Verfahren an, die aber an dieser Stelle nicht weiter betrachtet werden.

Die Wahl der oben beschriebenen Basisfunktionen φ_i hat nach Gleichung 2.15 einen direkten Einfluß auf den Aufbau der Matrix $\underline{\mathbf{A}}$.

$$a_{ij} = a(\varphi_i, \varphi_j) = \int_{\mathcal{R}} \nabla\varphi_i \cdot \nabla\varphi_j \, d\Omega \tag{2.15}$$

Falls die Elemente i und j nicht benachbart sind, ist das Produkt $\nabla\varphi_i \cdot \nabla\varphi_j$ gleich Null, so daß das entsprechende a_{ij} ebenfalls den Wert Null annimmt. Hieraus ergibt sich für die Matrix $\underline{\mathbf{A}}$ eine Bandstruktur und die Matrixgleichung 2.14 wird in Bezug auf die numerische Lösbarkeit erheblich vereinfacht.

Eine Darstellung der Finiten-Elemente- beziehungsweise Finite-Differenzen-Methode erfolgt in [HUY83].

2.2 Das Schwarz'sche Verfahren

Gebietszerlegungsverfahren haben für die Entwicklung numerischer Methoden zur Lösung von partiellen Differentialgleichungen mit Hilfe von Parallelrechnern eine wachsende Bedeutung gewonnen [WID89]. Für die parallele Lösung der oft sehr großen linearen oder nichtlinearen algebraischen Gleichungen, die bei der Lösung von elliptischen Differentialgleichungen durch Anwendung von Finite-Elemente- beziehungsweise Finite-Differenzen-Methoden entstehen, stellen Gebietszerlegungsverfahren einen guten Weg dar. Bei einem Gebietszerlegungsverfahren wird das gesamte Gebiet, in dem die zu berechnenden Gleichungen definiert sind, in eine Vielzahl von Teilgebieten zerlegt, die sich auch überlappen können. Für jedes Teilgebiet kann dann weitgehend unabhängig eine Lösung berechnet werden. Diese Verfahren können als "Divide and Conquer"-Algorithmen verstanden werden, bei denen geeignete Informationen zwischen den einzelnen Teilgebieten in Zusammenhang mit den Iterationen der Lösung ausgetauscht werden müssen [WID89]. Im folgenden soll ein Verfahren mit überlappender Gebietszerlegung betrachtet werden, das von Schwarz vorgestellt worden ist [SCH90].

Dieses Verfahren ist eines der ersten Gebietszerlegungsverfahren zur Lösung elliptischer Differentialgleichungen. Schwarz hat dieses Verfahren bereits 1869 ohne die Kenntnis der heutigen Parallelverarbeitung angegeben. Er wollte mittels des Verfahrens die Existenz von harmonischen Funktionen auf Gebieten mit nichtglatten Grenzen einführen [WID89]. Das Verfahren basiert auf einer alternierenden Berechnung der Lösung für zwei sich überlappende Teilgebiete. Zunächst wird die für das erste Teilgebiet angepaßte partielle Differentialgleichung entweder analytisch oder iterativ berechnet. Dann werden die Werte, die auf dem Rand des zweiten Gebietes liegen, als Dirichlet-Randwerte übernommen, und das zweite Gebiet wird

damit berechnet. In Abhängigkeit der gewünschten Lösungsgenauigkeit wird dieses Verfahren mehrmals wiederholt. Durch Zusammenführen der Teillösungen kann schließlich eine Näherung für die Gesamtlösung erhalten werden. Das Schwarz'sche Verfahren soll mit Hilfe eines Beispiels genauer vorgestellt werden.

Betrachtet werden soll die Poisson-Gleichung, angegeben in Gleichung 2.16, bei der ohne Einschränkung der Allgemeinheit die Randwerte auf dem Gebiet Ω als Dirichlet-Randwerte mit Nullwerten angenommen werden sollen.

$$-\Delta u(x,y) = f(x,y) \qquad \forall (x,y) \in \Omega \tag{2.16}$$

Das Gebiet Ω soll beschränkt sein, entweder zwei- oder dreidimensional aufgebaut sein und einen Lipschitzstetigen Rand haben. Die Berandung des Gebietes sei durch $\partial\Omega$ beschrieben.

$$u(x,y) = 0 \qquad \forall (x,y) \in \partial\Omega \tag{2.17}$$

Das Gesamtgebiet Ω wird nun in zwei sich überlappende Teilgebiete $\Omega^{(1)}$ und $\Omega^{(2)}$ zerlegt.

Entsprechend Abbildung 2.2 soll mit $\Omega^{(1)}$ das linke Teilgebiet, mit $\Omega^{(2)}$ das rechte Teilgebiet und mit $\Omega^{(3)}$ das beiden Teilgebieten gemeinsame Überlappgebiet bezeichnet sein. Die Berandung des Teilgebietes $\Omega^{(2)}$, die im Teilgebiet $\Omega^{(1)}$ liegt, soll mit Γ_4 und die Berandung von $\Omega^{(1)}$, die in $\Omega^{(2)}$ liegt, mit Γ_5 bezeichnet werden.

Durch Anwenden einer Finiten-Elemente-Methode wird die gegebene Differentialgleichung diskretisiert. Hierzu wird ein Finiter-Elemente-Raum V_h eingeführt, der ein Teilraum von $H_0^1(\Omega)$ ist. Entsprechend der im vorhergehenden Abschnitt dargestellten Finite-Elemente-Methode wird die Poisson-Gleichung beidseitig mit einer beliebigen Wichtungsfunktion $v \in H_0^1$ multipliziert. Nach Integration dieser Gleichung und Anwendung der 1. Green'schen Formel ergibt sich Gleichung 2.18. Da diese Gleichung in der Variationsformulierung angegeben ist, wird das Maximumprinzip, welches von Schwarz zum Beweis der Konvergenz seines Verfahrens benötigt wurde, überflüssig. Einen Beweis hierfür hat Sobolev [SOB36] angegeben.

$$a_\Omega(u,v) = \int_\Omega \nabla u \cdot \nabla v \, d\Omega = \int_\Omega f \cdot v \, d\Omega = f(v) \qquad \forall v \in H_0^1(\Omega) \tag{2.18}$$

Die Lösung u ist ebenfalls im Raum $H_0^1(\Omega)$ definiert. Damit kann das diskretisierte Problem mit Gleichung 2.19 beschrieben werden, wobei $u_h \in V_h$ die approximierte Lösung enthalten soll.

$$a_\Omega(u_h, v_h) = f(v_h) \qquad \forall v_h \in V_h \tag{2.19}$$

Die exakte Lösung dieser Gleichung kann angegeben werden durch

$$u_h = P_{V_h} u, \tag{2.20}$$

welches der orthogonalen Projektion der kontinuierlichen Lösung $u \in H_0^1(\Omega)$ von $H_0^1(\Omega)$ nach V_h entspricht [BJO89].

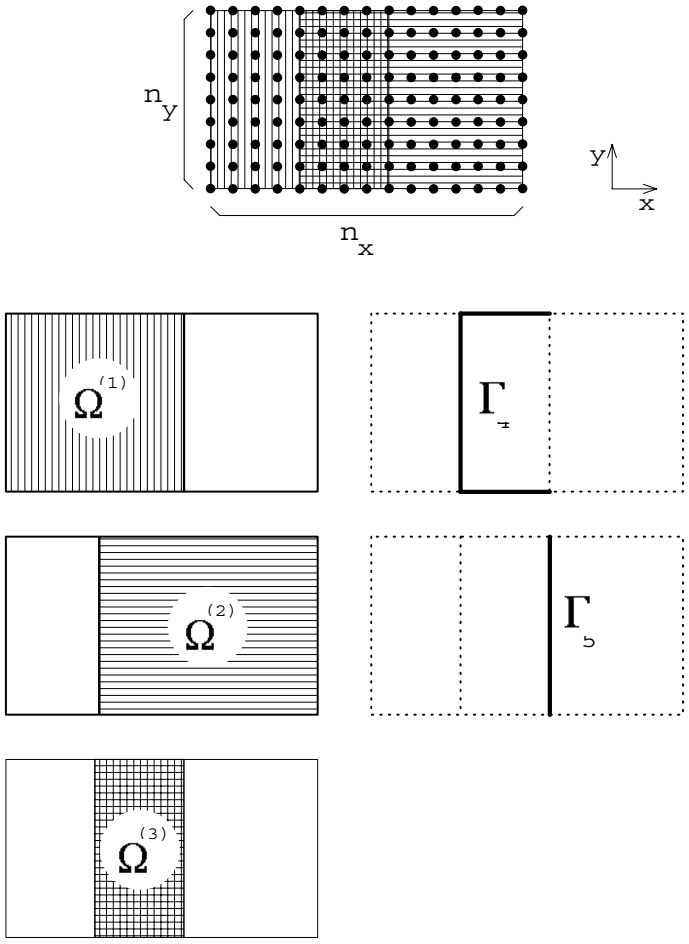


Abbildung 2.2: Bezeichnung der Teilgebiete und der Ränder für das Schwarz'sche Verfahren

Für den kontinuierlichen Fall kann das Schwarz'sche Verfahren in seiner traditionellen Form betrachtet werden. Bei diesem Verfahren gehören zu einem vollen Iterationsschritt jeweils zwei Teilschritte: der erste für das Teilgebiet $\Omega^{(1)}$ und der zweite für das Teilgebiet $\Omega^{(2)}$. u^{n+1} gibt die Lösung für den $(n+1)$-ten vollen Iterationsschritt mit $n = 0, 1, \ldots$ an.

$$\text{für Gebiet 1} \quad \begin{array}{lcll} -\Delta u^{n+1/2} & = & f & \text{in } \Omega^{(1)} \\ u^{n+1/2} & = & 0 & \text{für } \partial\Omega^{(1)} \cap \partial\Omega \\ u^{n+1/2} & = & u^n & \text{auf } \Gamma_5 \end{array}$$

$$\text{für Gebiet 2} \quad \begin{array}{lcll} -\Delta u^{n+1} & = & f & \text{in } \Omega^{(2)} \\ u^{n+1} & = & 0 & \text{für } \partial\Omega^{(2)} \cap \partial\Omega \\ u^{n+1} & = & u^{n+1/2} & \text{auf } \Gamma_4 \end{array}$$

P. L. Lions [LIO88], [LIO89] hat dieses klassische Verfahren genauer untersucht und durch Ausnutzen der Variationsformulierung neu angeben können. Er benutzte

orthogonale Projektionen, die sowohl für den kontinuierlichen als auch für den diskreten Fall angewendet werden können.

Das Schwarz'sche Verfahren kann damit in Variationsformulierung für das Teilgebiet $\Omega^{(1)}$ entsprechend

$$\begin{aligned} a_\Omega(u_h^{n+1/2} - u_h^n, v_h) &= f(v_h) - a_\Omega(u_h^n, v_h) \qquad \forall v_h \in V_h \\ &= a_\Omega(u_h - u_h^n, v_h) \end{aligned}$$

und für das Teilgebiet $\Omega^{(2)}$ entsprechend

$$\begin{aligned} a_\Omega(u_h^{n+1} - u_h^{n+1/2}, v_h) &= f(v_h) - a_\Omega(u_h^{n+1/2}, v_h) \qquad \forall v_h \in V_h \\ &= a_\Omega(u_h - u_h^{n+1/2}, v_h) \end{aligned}$$

angegeben werden.

Werden die Randwerte der Teilgebiete während der Zwischenschritte einer Iteration konstant gehalten, kann durch Ausnutzen der orthogonalen Projektion eine Vereinfachung des Verfahrens gewonnen werden. P_1 und P_2 entsprechen zwei linearen Operatoren, die die orthogonalen Projektionen von V_h auf $V_i = V_h \cap H_0^1(\Omega^{(i)})$ mit $i = 1, 2$ angeben. Der Finite-Elemente-Raum V_h kann damit dargestellt werden als die Summe der beiden Teilräume V_1 und V_2 [DRY90].

$$V_h = V_1 + V_2 \tag{2.21}$$

Die Iterationsvorschrift für die Lösung u_h kann nun mit Hilfe der Operatoren P_1 und P_2 festgelegt werden.

$$\begin{aligned} u_h^{n+1/2} - u_h^n &= P_1(u_h - u_h^n) \\ u_h^{n+1} - u_h^{n+1/2} &= P_2(u_h - u_h^{n+1/2}) \end{aligned} \tag{2.22}$$

Der Fehler, der bei der $(n+1)$-ten Iteration in Bezug auf die tatsächliche Lösung u_h auftritt, kann mit

$$\begin{aligned} e_h^{n+1} &= (I - P_2) \cdot (I - P_1) \cdot e_h^n \\ &= (I - (P_1 + P_2 - P_2 \cdot P_1)) \cdot e_h^n \end{aligned} \tag{2.23}$$

angegeben werden. In dieser Gleichung ist $e_h^n = u_h^n - u_h$ der Fehler bei der n-ten Iteration. Für bestimmte, geeignet gewählte Funktionen kann das Schwarz'sche Verfahren als eine direkte, iterative Methode zum Lösen der Gleichung 2.24 betrachtet werden [BJO89].

$$(P_1 + P_2 - P_2 \cdot P_1) \cdot u_h = g_m \tag{2.24}$$

Das im Vorhergehenden beschriebene Verfahren wird wegen des Produktterms $P_2 \cdot P_1$ in Gleichung 2.24 auch das *multiplikative* Schwarz'sche Verfahren genannt. Die Kombination der Operatoren P_i stellt ein Polynom vom Grad 2 dar. Zur parallelen Berechnung ist dieses Verfahren nicht ideal, da zwei sequentielle Schritte enthalten sind. Wenn mehr als zwei Teilgebiete benutzt werden sollen, dann verstärkt sich

dieser Effekt weiter, auch wenn der Grad des resultierenden Polynoms meistens kleiner ist, als er theoretisch sein könnte [WID92].

Um das Schwarz'sche Verfahren für den Einsatz auf Parallelrechnern verwenden zu können, muß Gleichung 2.24 angepaßt werden. Der einfachste Weg besteht darin, den störenden Produktterm wegzulassen [BJO89]. Zu lösen ist dann eine Gleichung, die durch ein Polynom 1. Grades bestimmt wird. Die rechte Seite der Gleichung 2.25 muß entsprechend berechnet werden, jedoch ist diese Berechnung ohne großen Aufwand möglich.

$$(P_1 + P_2) \cdot u_h = g_a \tag{2.25}$$

Dieses modifizierte Verfahren wird *additives* Schwarz'sches Verfahren genannt, da nur noch additive Verknüpfungen der Operatoren vorliegen. Bei der Anwendung dieses Verfahrens wird das gesamte Gebiet in überlappende Teilgebiete zerlegt. Die Pseudogrenzen, also die Grenzen, die zwischen den Teilgebieten liegen, müssen vor der ersten Iteration geeignet vorbelegt werden. Danach können sofort alle Teilgebiete parallel berechnet werden. Am Ende jedes Iterationsschrittes werden alle Pseudogrenzwerte gleichzeitig ausgetauscht. Der Unterschied zum multiplikativen Verfahren liegt darin, daß beim additiven Verfahren alle Gebiete zeitgleich und unabhängig voneinander berechnet werden können. Beim multiplikativen Verfahren konnten die Teilgebiete nur nacheinander berechnet werden, da der Austausch der Randwerte erst erfolgen konnte, wenn ein Teilgebiet vollständig berechnet war.

Ist der Operator P_1+P_2 symmetrisch und positiv definit in Bezug auf die bilineare Form, so kann zur Lösung der Matrixgleichung das konjugierte Gradientenverfahren (CG) eingesetzt werden. Es kann gezeigt werden, daß die Zahl der CG-Iterationen, die beim additiven Verfahren benötigt werden, maximal das Doppelte der Zahl der CG-Iterationen beim multiplikativen Verfahren bei gleichbleibender Genauigkeit der Ergebnisse ist [BJO89]. Hieraus folgt, daß im schlechtesten Fall (Berechnung von zwei Teilgebieten mit zwei Prozessoren) das additive Verfahren doppelt so viele Iterationen benötigt wie das multiplikative Verfahren bei Einsatz nur eines Prozessors. Für die parallele Berechnung werden aber in der Regel eine sehr große Zahl Teilgebiete benutzt, wodurch sich der Unterschied der beiden Verfahren zugunsten des additiven Verfahrens verbessert.

Eine Verbesserung der Konvergenzrate des Schwarz'schen Verfahrens kann nach G. Rodrigue [ROD89] erreicht werden, wenn die Randwerte, die an den Grenzen der Teilgebiete ausgetauscht werden, nicht als Dirichlet-Werte sondern als gemischte Randwerte gesetzt werden – auf der einen Seite Neumann-Randwerte, auf der anderen Seite Dirichlet-Randwerte. Rodrigue hat weiter angegeben, daß die beste Effizienz bei paralleler Berechnung nur dann erzielt werden kann, wenn die Teilgebiete annähernd gleich groß sind und der Überlapp der Teilgebiete minimal ist.

Das Schwarz'sche Verfahren für mehr als zwei sich überlappende Teilgebiete wird in [DRY90] und [BJO90] beschrieben.

Kapitel 3

Das Programm TRACE

Das Programm TRACE (Transport of Contaminants in Environmental Systems) ist ein Simulationsprogramm, das den Wasserfluß in variabel gesättigten, porösen, heterogenen Medien berechnet und mit dem Stofftransport in diesen Zonen koppelt. TRACE basiert auf einem dreidimensionalen Finiten-Elemente-Modell des Wasserflusses in gesättigten und ungesättigten porösen Medien. Das Modell gestattet es, viele für die Hydrogeologie wichtige Strukturen wie heterogene und anisotrope Medien zu behandeln. Es können sowohl räumlich verteilte als auch punktförmig angeordnete Quellen und Senken simuliert werden.

Partielle Differentialgleichungen bilden die vielfältigen Wirkungszusammenhänge auf ein mathematisches Gerüst ab. Als Randbedingungen können Cauchy'sche, Neumann'sche, Dirichlet'sche und variable Bedingungen angenommen werden. Für die numerische Lösung dieser Gleichungen wird eine Diskretisierung mit einer großen Zahl an Knotenpunkten benötigt. Das resultierende Gleichungssystem ist zeitabhängig und unter bestimmten Bedingungen streng nichtlinear.

Um den Einsatzbereich des Programms TRACE besser verstehen zu können, soll im folgenden ein kurzer hydrogeologischer Überblick über den Wasserfluß im Boden gegeben werden. Danach werden die im Programm verwendeten Modellgleichungen vorgestellt. Eine Beschreibung des Programms TRACE und der Gründe, die eine Erweiterung des Programm-Codes erforderlich machten, schließen sich an. Als Referenzversion des Programms TRACE für die Erklärungen in diesem Kapitel wird die Version betrachtet, wie sie am Anfang dieser Arbeit eingesetzt wurde.

Da in der vorliegenden Arbeit nur der Programmteil zur Berechnung des Wasserflusses bearbeitet wurde, wird im folgenden der Stofftransport weitgehend vernachlässigt. Eine ausführliche Beschreibung findet sich in [VER93] und [VER94].

3.1 Grundlegende hydrogeologische Begriffe

Um einen Einblick in die hydrogeologischen Zusammenhänge gewinnen zu können, auf denen das im Programm TRACE verwendete Modell basiert, sollen in diesem Abschnitt die wichtigsten Begriffe erklärt werden. Eine allgemeine Definition des Begriffs Grundwasser wird in der DIN 4049-3 [DIN49] angegeben:

"Grundwasser ist unterirdisches Wasser, das die Hohlräume der Lithosphäre zusammenhängend ausfüllt und dessen Bewegungsmöglichkeit ausschließlich durch die Schwerkraft bestimmt wird."

Gesteinskörper, die Hohlräume enthalten und damit geeignet sind, Grundwasser weiterzuleiten, werden als Grundwasserleiter oder in Anlehnung an den englischen Ausdruck auch als Aquifer bezeichnet. Unter natürlichen Bedingungen enthält jeder Boden[1] ab einer bestimmten Tiefe stets Wasser. Der Wassergehalt des Bodens wird durch die Sättigung gekennzeichnet. Falls der gesamte Porenraum des Bodens mit Wasser gefüllt ist, liegt eine Sättigung von einhundert Prozent vor, es wird von der gesättigten Zone gesprochen. Wenn einzelne Poren nicht mehr mit Wasser gefüllt sind, fällt der Wassergehalt unter einhundert Prozent und es wird von der ungesättigten Zone gesprochen.

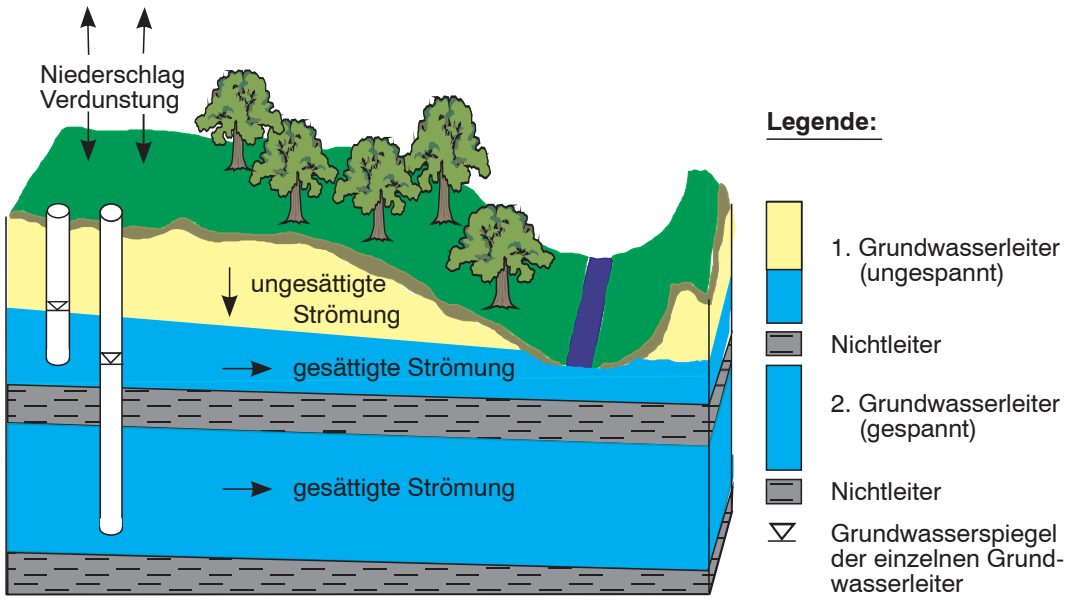


Abbildung 3.1: Profilschnitt durch den Boden zur Darstellung von verschiedenen hydrogeologischen Begriffen

In Abbildung 3.1 werden die wichtigsten hydrogeologischen Begriffe zusammengefaßt dargestellt. Zu erkennen ist ein Profilschnitt durch den Boden, in dem zwei Grundwasserleiter beziehungsweise zwei Nichtleiter eingezeichnet sind.

Der erste Grundwasserleiter kann in zwei Schichten aufgeteilt werden. In der oberen Schicht liegt eine ungesättigte Wasserströmung in vertikaler Richtung vor. Die

[1]Der Begriff Boden wird in dieser Arbeit für Untergrund verwendet. Im geologischen Zusammenhang bezeichnet Boden jedoch nur die oberen Dezimeter bis Meter des Erdreichs. In [SCH69] wird eine Definition des Begriffs Boden angegeben: Boden ist das mit Wasser, Luft und Lebewesen durchsetzte, unter dem Einfluß der Umweltfaktoren an der Erdoberfläche entstandene und im Ablauf der Zeit sich weiterentwickelnde Umwandlungsprodukt mineralischer und organischer Substanzen mit eigener morphologischer Organisation, das in der Lage ist, höheren Pflanzen als Standort zu dienen und die Lebensgrundlage für Tiere und Menschen bildet.

Sättigung beträgt in dieser Schicht weniger als einhundert Prozent. Durch Niederschlag oder Verdunstung kann sich der Wassergehalt in dieser Schicht ändern. In der unteren Schicht ist der Boden zu einhundert Prozent mit Wasser gesättigt, es stellt sich eine gesättigte Wasserströmung in horizontaler Richtung ein. Der Übergang von der oberen zur unteren Schicht kennzeichnet den Grundwasserspiegel. Im ersten Grundwasserleiter wird auch von freiem Grundwasser gesprochen, da sich das Wasser im hydraulischen Gleichgewicht befindet. Die Grundwasseroberfläche fällt mit dem Grundwasserspiegel zusammen und liegt innerhalb des ersten Grundwasserleiters, ein Anstieg des Wasserstandes ist möglich. Grundwasserleiter mit diesen Eigenschaften werden ungespannte Leiter genannt.

Nach unten wird der erste Grundwasserleiter durch eine nur gering wasserdurchlässige Schicht abgeschlossen. Diese Schicht verhindert weitgehend eine vertikale Wasserströmung und wird Nichtleiter genannt.

Der zweite Grundwasserleiter wird oben und unten durch einen Nichtleiter begrenzt. Der Sättigungsgehalt in diesem Leiter beträgt einhundert Prozent, es liegt eine horizontale, gesättigte Wasserströmung vor. Durch die begrenzenden Schichten erhöht sich der Wasserdruck innerhalb des zweiten Grundwasserleiters, so daß sich das Wasser nicht mehr im hydraulischen Gleichgewicht befindet. Die Grundwasseroberfläche entspricht dem oberen Abschluß des Leiters. Der Grundwasserspiegel liegt höher als die Grundwasseroberfläche, da sich das Wasser infolge des erhöhten Druckes ausbreitet. Ein Grundwasserleiter, bei dem die Grundwasseroberfläche nicht mit dem Grundwasserspiegel übereinstimmt, wird als gespannter Leiter bezeichnet.

Die wasserführenden Schichten bestehen oft aus Sanden und Kiesen, während die nur gering wasserdurchlässigen Schichten aus Ton bestehen.

Die Dynamik eines Grundwasserleiters hängt entscheidend von den hydraulischen Druckverhältnissen ab. Auch bei freiem Grundwasser herrschen innerhalb des Grundwasserleiters unterschiedliche Drücke. An der freien Grundwasseroberfläche ist der Druck gleich dem der überlagernden Luft (Luftdruck), mit zunehmender Tiefe addiert sich der Druck, der aus dem Gewicht der überlagernden Wassersäule resultiert.

Der Grundwasserspiegel ist in der Regel natürlichen zeitlichen Schwankungen unterworfen, die besonders durch die Niederschlagsmengen verursacht werden. Ebenso haben der Grundwasserzufluß, der Grundwasserabfluß und die künstliche Förderung von Grundwasser, z. B. durch Brunnen, Einfluß auf den Verlauf des Grundwassers. Etwa 80% der Grundwasserneubildung erfolgt im Winterhalbjahr zwischen November und April. In diesem Zeitraum ist die Wasseraufnahme durch die Pflanzen und die Verdunstungsrate des Wassers im Boden geringer als in den Sommermonaten.

Zeitlich kann das Wasser unter Umständen mehrere Jahre benötigen, um von der Geländeoberfläche in den gesättigten Bereich des ersten Grundwasserleiters zu gelangen. Diese Tatsache hängt jedoch sehr stark von den Eigenschaften des zu durchdringenden Bodens und dessen Bewuchs ab. Die Geschwindigkeit des vorwiegend horizontalen Wasserflusses in einem Grundwasserleiter liegt zwischen einigen Millimetern bis Metern pro Tag. Je tiefer ein Grundwasserleiter liegt, desto älteres Wasser führt er im allgemeinen. In tieferen Grundwasserleitern kann das Wasser

schon seit mehreren zehntausend Jahren transportiert worden sein. Aus diesen Zahlenwerten wird ersichtlich, daß sich der Einfluß von Schadstoffen im Grundwasser gegebenenfalls erst zu einem sehr viel späteren Zeitpunkt bemerkbar macht oder über einen langen Zeitraum von Bedeutung sein kann.

Der oberste Grundwasserleiter ist in vielen Gebieten der wichtigste zur Wassergewinnung. Durch den Einfluß des Menschen kann dieser Leiter regional schon derart stark belastet sein, daß die Qualität des Wassers nicht mehr den Anforderungen für Trinkwasser genügt. In diesem Fall wird bei der Wassergewinnung Wasser aus tiefer liegenden Leitern zugemischt, so daß die erforderlichen Grenzwerte eingehalten werden können.

Weitere wichtige Effekte, die vor allem den obersten Grundwasserleiter betreffen, sind die Transpiration (Aufnahme von Wasser durch Pflanzenwurzeln und Abgabe an die Atmosphäre) und die Evaporation (Verdunstung von Wasser im Boden oder von freien Wasserflächen).

Schadstoffe können entweder über die Bodenoberfläche (z.B. Landwirtschaft) oder aber direkt im Boden (z.B. Mülldeponien) in einen Grundwasserleiter eindringen. Die Stoffe können im Wasser gelöst oder gebunden, z. B. an Kolloide oder Bakterien, vorliegen. Abhängig von der Art des Schadstoffes kann sich dieser konservativ oder reaktiv verhalten. Konservative Schadstoffe unterliegen während ihres Transportes keinen Reaktionen. Reaktive Schadstoffe können mit dem Boden, der Biomasse oder anderen Substanzen reagieren.

Den Transport von Schadstoffen im Grundwasser bestimmen vor allem Dispersions-, Diffusions- und Sorptionsprozesse sowie chemische Reaktionen. Mit Dispersion wird die Tatsache beschrieben, daß sich eine Schadstoffwolke mit zunehmender Entfernung beziehungsweise Zeit in allen Raumrichtungen ausbreitet. Unter der Diffusion wird die Ausbreitung der Schadstoffmoleküle im Wasser aufgrund der Brown'schen Bewegung verstanden, wobei der hydrogeologische Begriff der Diffusion dem physikalischen Begriff der thermischen Diffusion entspricht. Unter Sorption fällt der Effekt, daß sich Teile des im Wasser gelösten Schadstoffes an eine Oberfläche binden und sich erst verzögert weiterbewegen.

Da dieser Abschnitt nur einen kurzen Überblick geben sollte, sind nur wenige, zum allgemeinen Verständis nötige Zusammenhänge aufgezeigt worden. Weiterführende Betrachtungen können in [FET93], [FRE79], [MAT83] und [SCH84] gefunden werden.

3.2 Herleitung der Modellgleichungen

Das erste grundlegende Gesetz für den Wasserfluß im Boden hat Buckingham (1907) aufgestellt [FET93]. Er erkannte, daß der hydrostatische Druck Ψ, hervorgerufen durch die überlagernde Wassersäule, im ungesättigten Boden eine Funktion des Wassergehalts Θ, der Temperatur T und der Gesamtdichte des Bodens ist. Der Druck Ψ, welcher auch als Potential verstanden werden kann, wird definiert als Arbeit, die notwendig ist, um eine Einheitsmenge Wasser von einem gegebenen Punkt eines Kraftfeldes zu einem Bezugspunkt zu transportieren. Das Potential des Wassers

kann mit Gleichung 3.1 angegeben werden.

$$\Psi = m \cdot g \cdot h \tag{3.1}$$

Hierbei ist m die Masse des Wassers, g die Erdbeschleunigung und h die Höhe über einer freien Wasserfläche als Bezugsniveau. Eine Normierung des Potentials auf als konstant angenommene Größen ist üblich. Sehr oft wird das Gewicht des Wassers als Bezugsgröße gewählt, so daß Ψ die Dimension einer Länge, angegeben in *cm* Wassersäule, annimmt.

Das Wasser bewegt sich immer von Stellen höheren Potentials (höhere potentielle Energie) zu Stellen mit niedrigerem Potential. Diese Bewegung hält so lange an, bis an allen Stellen das Gesamtpotential den gleichen Wert erreicht hat. Nach Sposito [SPO86] hatte Buckingham keine Kenntnis der Arbeiten von Darcy, der bereits 1856 erste experimentelle Versuche zur Untersuchung des Grundwasserflusses in mit Sand gefüllten Röhren durchgeführt hatte und die gewonnenen Erkenntnisse in einer Formel angeben konnte [FRE79]. Der Wasserfluß kann über eine Funktion der Potentialdifferenzen in der ungesättigten (nach Buckingham) beziehungsweise in der gesättigten Zone (nach Darcy) dargestellt werden, welches dem Gradienten des totalen Potentials Φ entspricht. Das totale Potential Φ eines Volumenelementes setzt sich zusammen aus dem hydrostatischen Potential Ψ und dem Gravitationspotential Ψ_z. Das Gravitationspotential gibt den Einfluß des Gravitationsfeldes der Erde auf das im Boden befindliche Wasser an. Wird das Gewicht des Wassers als Bezugseinheit verwendet, dann entspricht das Gravitationspotential der Ortshöhe z. Das Bezugsniveau für das totale Potential wird so gewählt, daß das Gravitationspotential ein positives Vorzeichen und somit von der freien Wasseroberfläche nach oben gerichtet zunehmende Beträge erhält. Gleichung 3.2 gibt den Zusammenhang zwischen dem totalen Potential Φ, dem hydrostatischen Potential Ψ und dem Gravitationspotential Ψ_z an, wenn das Gewicht als Bezugseinheit verwendet wird.

$$\begin{aligned} \Phi &= \Psi + \Psi_z \\ &= \Psi + z \end{aligned} \tag{3.2}$$

Für das totale Potential wird häufig auch der Begriff Standrohrspiegelhöhe und für das hydraulische Potential der Begriff Druckhöhe verwendet. Der Einfluß des atmosphärischen Drucks wird durch eine Korrektur der gemessenen Pegelstände berücksichtigt, so daß er in der gesamten Rechnung vernachlässigt werden kann.

Um die Zusammenhänge zwischen dem Wasserfluß und dem Gradienten des totalen Potentials auszudrücken, wird noch eine Proportionalitätsmatrix $\underline{\mathbf{K}}$ benötigt, die im allgemeinen eine Funktion der Sättigung des Bodens mit Wasser ist. Der Sättigungsgehalt ist in der ungesättigten Zone eine Funktion des hydrostatischen Potentials und wird oft durch Auswerten von sogenannten Wasserspannungskurven ermittelt. In gesättigten Zonen ist der Wert des Sättigungsgehalts immer konstant einhundert Prozent. Die Proportionalitätsmatrix $\underline{\mathbf{K}}$ wird auch hydraulische Leitfähigkeit beziehungsweise Permeabilität genannt.

Die Formel 3.3 gibt das Flußgesetz nach Buckingham beziehungsweise Darcy an, in dem $\vec{Q}$ den Wasserfluß in Volumeneinheiten pro Zeit, $\underline{\mathbf{K}}(\mathbf{\Psi})$ die hydraulische Leitfähigkeit in Längeneinheiten pro Zeit, $\nabla\Phi$ den dimensionslosen Gradienten des totalen Potentials und A die durchströmte Fläche in Flächeneinheiten bezeichnet.

$$\vec{Q} = -\underline{\mathbf{K}}(\mathbf{\Psi}) \cdot \nabla\Phi \cdot A \tag{3.3}$$

Eine weitere Größe, die im Modell für die Kopplung des Wasserflusses mit dem Stofftransport benötigt wird, ist die Fließgeschwindigkeit des Wassers im Boden. Da durch Gleichung 3.3 die gesamte Wassermenge, die durch eine gegebene Fläche fließt, berechnet werden kann, muß diese Gleichung nur durch die betrachtete Fläche dividiert werden, um die Geschwindigkeit zu erhalten. In Gleichung 3.4 soll A diese Fläche bezeichnen.

$$\vec{q} = \frac{\vec{Q}}{A} = -\underline{\mathbf{K}}(\mathbf{\Psi}) \cdot \nabla\Phi \tag{3.4}$$

Die Kontinuitätsgleichung für den Wasserfluß durch ein Volumenelement in der ungesättigten Zone kann angegeben werden durch die Änderung des Wassergehalts Θ mit der Zeit. Dies entspricht der Summe des zu- und abfließenden Wassers in dem betrachteten Volumenelement. Gleichung 3.5 gibt die Kontinuitätsgleichung an, wobei q_x, q_y, q_z den Wasserfluß in den einzelnen Raumrichtungen kennzeichnet.

$$\frac{\partial\Theta}{\partial t} = -(\frac{\partial q_x}{\partial x} + \frac{\partial q_y}{\partial y} + \frac{\partial q_z}{\partial z}) \tag{3.5}$$

In Vektornotation ergibt sich hieraus

$$\frac{\partial\Theta}{\partial t} = -\nabla \cdot \vec{q} \tag{3.6}$$

Durch Kombination der Gleichungen 3.6 und 3.3 ergibt sich die Richards-Gleichung, angegeben in Gleichung 3.7, die Richards 1931 [RIC31] angegeben hat.

$$\frac{\partial\Theta}{\partial t} = \nabla \cdot [\underline{\mathbf{K}}(\mathbf{\Psi}) \cdot \nabla\Phi] \tag{3.7}$$

Durch eine Umformung des Differentialoperators auf der linken Seite von Gleichung 3.7 gelangt man zur Größe des Speicherkoeffizienten F, der mit der Einheit Eins pro Länge angegeben werden kann.

$$\frac{\partial\Theta}{\partial t} = \frac{\partial\Theta}{\partial\Psi} \cdot \frac{\partial\Psi}{\partial t} \tag{3.8}$$

$$F = \frac{\partial\Theta}{\partial\Psi} \tag{3.9}$$

Damit läßt sich die Richards-Gleichung in der folgenden Form angeben.

$$F \cdot \frac{\partial\Psi}{\partial t} = \nabla \cdot [\underline{\mathbf{K}}(\mathbf{\Psi}) \cdot \nabla\Phi] \tag{3.10}$$

Eine Erweiterung der Richards-Gleichung um einen Quellen-/Senkenterm S führt auf Gleichung 3.11, die die allgemeine Form dieser Gleichung angibt.

$$F \cdot \frac{\partial \Psi}{\partial t} = \nabla \cdot [\underline{\mathbf{K}}(\mathbf{\Psi}) \cdot \nabla \Phi] + S \tag{3.11}$$

Die Richards-Gleichung beschreibt den Wasserfluß in gesättigten und ungesättigten Bodenzonen. Bedingungen, die für die Gültigkeit der Richards-Gleichung erfüllt sein müssen, sind im einzelnen, daß die Temperatur, die Dichte des Wassers und der Luftdruck konstant sind und daß die Permeabilitätsmatrix $\underline{\mathbf{K}}$ nicht deformierbar ist, woraus folgt, daß die Eigenschaften des Bodens bei Änderung des Wasserdrucks erhalten bleiben. Die Richards-Gleichung ist eine partielle Differentialgleichung vom Typ der parabolischen Differentialgleichungen, die durch zeitliche Ableitungen erster Ordnung und zweiten Ableitungen nach dem Ort gekennzeichnet werden. Falls die zeitliche Ableitung des Wassergehaltes Θ Null ergibt, ist die linke Seite der Gleichung 3.11 identisch Null und der Typ der Richards-Gleichung ändert sich in eine elliptische Differentialgleichung.

In der ungesättigten Zone kann der Wassergehalt Θ durch verschiedene empirisch hergeleitete Gleichungen angegeben werden. Im Modell wird die van-Genuchten-Gleichung benutzt, die van Genuchten 1980 [VGE80] empirisch aus dem Zusammenhang zwischen dem hydrostatischen Druck und dem Wassergehalt hergeleitet hat. Formel 3.12 gibt diese Gleichung an, wobei die Parameter n und α entsprechend Gleichung 3.13 und 3.14 berechnet werden können.

$$\Theta = \Theta_r + \frac{\Theta_s - \Theta_r}{[1 + (\alpha \cdot \Psi)^n]^m} \tag{3.12}$$

$$n = \frac{1}{1 - m} \tag{3.13}$$

$$\alpha = \frac{1}{h_b} \cdot (2^{\frac{1}{m}} - 1)^{1-m} \tag{3.14}$$

Die einzelnen Größen in Gleichung 3.12 werden aus einer experimentell ermittelten Wasserspannungskurve bestimmt, die die Zusammenhänge zwischen Ψ und Θ darstellt. Hierbei entspricht Θ_s dem Wassergehalt für sehr große Ψ – Wassergehalt bei Sättigung ($\Theta = \Theta_s$) für $\Psi = 0$ – und Θ_r dem Wassergehalt für sehr kleine Ψ – Wassergehalt bei Restsättigung ($\Theta = \Theta_r$) für $\Psi \ll -1$. Der Wert für den Parameter m kann ebenfalls aus dieser Kurve bestimmt werden. h_b gibt den Druck an, bei dem der Boden entwässert wird (*bubbling pressure*). Auch dieser Wert kann mit Hilfe der Wasserspannungskurve bestimmt werden.

Die hydraulische Leitfähigkeit $\underline{\mathbf{K}}$ kann mit der Mualem-van-Genuchten-Gleichung berechnet werden. In dieser Gleichung wird die hydraulische Leitfähigkeit K_{sat} im gesättigten Leiter benutzt, welche ebenso wie der funktionale Zusammenhang zwischen Θ und Ψ räumlich variabel ist. Die Werte dieser Größen können durch Zufallsfunktionen angegeben werden, die aus dem ersten und dem zweiten statistischen Moment entstehen.

Die Richards-Gleichung bildet den Kern des Programms TRACE. Um diese Gleichung vollständig angeben zu können, müssen Anfangs- und Randwerte vorgegeben werden. Die Randwertbedingungen können Dirichlet'sche, Neumann'sche, Cauchy'sche oder variable Bedingungen sein, je nachdem, welche Fließbedingungen an den Rändern des Gebietes, in Abhängigkeit von den verschiedenen Zonen, gelten sollen.

Unter einer Dirichlet'schen Randbedingung versteht man eine Randbedingung erster Art, bei der auf dem Rand des Gebietes ein konstanter Funktionswert vorgegeben wird. Diese Bedingungen sollten bei Boden-Wasser-Grenzflächen, wie beispielsweise Seen und Flüssen, verwendet werden. Eine Neumann'sche beziehungsweise Cauchy'sche Randbedingung, auch Randbedingungen zweiter Art genannt, liegt vor, wenn auf dem Rand des Gebietes für die Normalenableitung der Funktion ein konstanter Wert vorgegeben wird. Bei Neumann'schen Bedingungen wird der Gradient von Φ, dem totalen Potential, und bei Cauchy'schen Bedingungen der Gradient von Ψ, dem hydraulischen Potential, vorgegeben. Bei einer variablen Randbedingung kann sich die Art der Bedingung während der Berechnung ändern. Es können hierbei sowohl Dirichlet-Randwerte als auch Cauchy'sche Randbedingungen gewählt werden. Die variablen Randbedingungen sollten bei Boden-Luft-Grenzflächen benutzt werden, um Niederschläge, Evaporation und Transpiration angeben zu können.

Dirichlet'sche Randbedingung

$$\Phi = \Phi_d \tag{3.15}$$

Neumann'sche Randbedingung

$$-\vec{n} \cdot \underline{\mathbf{K}} \cdot \nabla\Phi = \vec{v}_n \tag{3.16}$$

Cauchy'sche Randbedingung

$$\vec{n} \cdot \underline{\mathbf{K}} \cdot \nabla\Psi = \vec{v}_c \tag{3.17}$$

Variable Randbedingung

$$\begin{aligned} \Phi &= \Phi_p \\ -\vec{n} \cdot \underline{\mathbf{K}} \cdot \nabla\Psi &= \vec{q}_{net} \\ \Phi &= \Phi_{min} \end{aligned} \tag{3.18}$$

Die Anfangsbedingungen werden für das totale Potential Φ für jeden Punkt des Definitionsbereiches der Richards-Gleichung angegeben.

Anfangsbedingungen

$$\Phi\,|_{t=0} = \Phi_0(\vec{x}) \tag{3.19}$$

Die Bodenbeschaffenheit kann eine Veränderung des Aufbaus der Modellgleichungen bewirken. Wenn der Wasserfluß im gesättigten Grundwasserleiter berechnet

werden soll, ist die Permeabilität $\underline{\mathbf{K}}$ beispielsweise keine Funktion des hydrostatischen Druckes Ψ, so daß sich die Gleichungen vereinfachen lassen.

Für die Umsetzung des mathematischen Modells in von Computern lösbare Formeln wird die zeitliche Abhängigkeit der Gleichungen nach der Methode der finiten Differenzen und die räumliche Abhängigkeit nach der Methode der finiten Elemente diskretisiert. Die Finite-Elemente-Methode basiert auf einem Ansatz von Bubnov-Galerkin, bei dem lineare hexaedrische Elemente benutzt werden. Eine Erklärung der verwendeten Finite-Elemente-Methode kann in Abschnitt 2.1 nachgelesen werden.

Die räumliche und zeitliche Diskretisierung führt auf eine nichtlineare Matrix-Gleichung (siehe Gleichung 3.20), bei der $\underline{\mathbf{A}}$ eine Koeffizientenmatrix in Abhängigkeit vom Lösungsvektor $\vec{h}$ und $\vec{y}$ ein bekannter Vektor, bestehend aus Informationen aus den Randwerten des Problems, ist.

$$\underline{\mathbf{A}}(\vec{\mathbf{h}}) \cdot \vec{h} = \vec{y} \tag{3.20}$$

Die wesentliche Aufgabe für das Programm TRACE liegt darin, diese Matrixgleichung numerisch zu lösen. Da eine direkte Lösung nicht angegeben werden kann, wird die Gleichung, nachdem sie linearisiert wurde, iterativ durch ein Konjugiertes-Gradienten-Verfahren [GOL93], [BAS95] gelöst.

Im Programm-Code von TRACE ist auch ein mathematisches Modell für den Stofftransport im Grundwasser enthalten. Die grundlegende mathematische Gleichung ist die Konvektions-Dispersions-Gleichung, bei der c die Stoffkonzentration im Wasser, $\vec{u}$ die Porengeschwindigkeit, $\underline{\mathbf{D}}$ den Dispersions-/Diffusionstensor und S einen Quellen-/Senkenterm bezeichnet.

$$\nabla \cdot (\underline{\mathbf{D}} \cdot \nabla c - c \cdot \vec{u}) = \frac{\partial c}{\partial t} + S(c,t) \tag{3.21}$$

Auf die Beschreibung des Modells wird an dieser Stelle jedoch verzichtet, da es zum einen den Rahmen dieser Arbeit überschreiten würde und zum anderen in der momentanen parallelen Programmversion keine Anwendung findet.

Eine ausführliche Beschreibung des im Programm TRACE benutzten Modells findet sich in [VER94].

3.3 Entwicklungsgeschichte von TRACE

Das Programm TRACE basiert auf zwei Programmen, die von Yeh (Oak Ridge National Laboratory (ORNL)) entwickelt wurden. Zum einen ist dies das Programm 3DFEMWATER [YEH87], geschrieben 1987, welches zur Berechnung des Grundwasserflusses in gesättigten und ungesättigten, porösen, heterogenen Medien verwendet wird. Das zweite Programm ist ein von Yeh nicht veröffentlichter Code, der den Stofftransport im Grundwasser simulieren kann.

Durch die Kombination der beiden Teile im Institut für Chemie und Dynamik der Geosphäre, Institut für Erdöl und Organische Geochemie (ICG-4), Forschungszentrum Jülich, ist ein umfangreiches und leistungsstarkes Simulationsprogramm für

hydrogeologische Fragestellungen entstanden [VER94]. Dieses Programm, TRACE genannt, besteht aus ca. 12.000 Zeilen FORTRAN 77-Code und ist auf unterschiedlichsten Rechnersystemen, von PC's über Workstations bis hin zu parallelen Rechnersystemen, einsetzbar.

Im Forschungszentrum Jülich ist das Programm weiterentwickelt worden. Unter anderem wurden numerische Verfahren durch effizientere und modernere Verfahren ersetzt und Anpassungen des Programms für spezielle Rechnerarchitekturen durchgeführt. Systeme, auf denen TRACE ausgeführt wurde, sind beispielsweise die CRAY X-MP/416, die CRAY Y-MP8/832 und der iPSC/860 von Intel [VER93].

Die Entwicklung des Programms TRACE ist noch nicht abgeschlossen, die zur Zeit existierenden Produktionsversionen gestatten jeweils nur in der Funktionalität eingeschränkte Simulationsläufe. Eine graphische Darstellung der Berechnungsergebnisse ist noch nicht möglich.

In jüngster Zeit sind mit dem Programm TRACE erste Probeläufe für den Einsatz auf massiv-parallelen Rechnersystemen durchgeführt worden. Es wurden zwei verschiedene Parallelisierungsstrategien verfolgt. Zum einen ist ein Gebietszerlegungsverfahren, aufbauend auf dem Schwarz'schen Verfahren, und zum anderen ein Ansatz zur Verteilung der Matrizen (*data partitioning method*) implementiert und getestet worden. In [GER94] und [VNL94] können die Ergebnisse dieser Probeläufe nachgelesen werden. Da für diese Vorversionen nur die nötigsten Programmänderungen durchgeführt wurden und dadurch die benutzten Verfahren nur für eingeschränkte Testfälle lauffähig waren, ist es das Ziel dieser Arbeit, den Gebietszerlegungsansatz für eine automatische dreidimensionale Partitionierungsstrategie zu verallgemeinern.

3.4 Struktur von TRACE

In diesem Abschnitt soll die Struktur des Programms TRACE vorgestellt werden. Als Grundlage für diese Darstellungen wird die Programmversion herangezogen, wie sie am Anfang dieser Arbeit eingesetzt wurde. Zunächst soll die Umsetzung des in Abschnitt 3.2 betrachteten Modells aufgezeigt werden. Danach werden die Datenstrukturen und die Dateien mit den Eingabeparametern erläutert, da diese bei der Parallelisierung von TRACE angepaßt werden müssen.

3.4.1 Umsetzung des Modells

Durch Anwendung der Finiten-Elemente-Methode auf die verallgemeinerte Richards-Gleichung (siehe Gleichung 3.11) ergibt sich eine nichtlineare Matrixgleichung entsprechend Gleichung 3.22.

$$\underline{\mathbf{F}} \cdot \frac{d}{dt}\vec{h} + \underline{\mathbf{K}} \cdot \vec{h} - \vec{R} + \vec{D} - \vec{S} = 0 \tag{3.22}$$

In dieser Gleichung steht $\underline{\mathbf{F}}$ für die Gewichtsmatrix, $\underline{\mathbf{K}}$ für die Leitfähigkeitsmatrix, $\vec{R}$ und $\vec{D}$ für Lastvektoren und $\vec{S}$ für einen Quellen-/Senkenvektor. Der Vektor $\vec{h}$ entspricht dem hydrostatischen Potential für alle diskreten Gitterpunkte und stellt

den Lösungsvektor dieser Matrixgleichung dar. Alle Vektoren haben die Dimension n, die Matrizen besitzen die Gestalt $[n \times n]$. Der Parameter n entspricht der Zahl der FE-Knoten des durch die Finite-Elemente-Methode erzeugten Gitters. Um die weitere Beschreibung übersichtlicher zu gestalten, soll der Vektor $\vec{r}$ eingeführt werden.

$$\vec{r} = \vec{R} - \vec{D} + \vec{S} \tag{3.23}$$

Hiermit ergibt sich für Gleichung 3.22 die folgende Form.

$$\underline{\mathbf{F}} \cdot \frac{d}{dt}\vec{h} + \underline{\mathbf{K}} \cdot \vec{h} = \vec{r} \tag{3.24}$$

Um die zeitliche Ableitung des Vektors $\vec{h}$ numerisch bestimmen zu können, wird eine Finite-Differenzen-Methode benutzt. Dazu wird die kontinuierliche Zeitvariable t in eine Zahl gleichgroßer Zeitschritte Δt zerlegt.

$$t_n = n \cdot \Delta t \qquad n = 0, 1, \ldots \tag{3.25}$$

Damit kann der $(n+1)$-te Zeitschritt durch den n-ten Zeitschritt angegeben werden.

$$t_{n+1} = t_n + \Delta t \tag{3.26}$$

Durch Anwendung eines Differenzenschemas, dessen Struktur durch den Parameter ω mit $0 \leq \omega \leq 1$ verändert werden kann, wird Gleichung 3.24 zeitlich diskretisiert. Die Hochindizes sollen im folgenden den Zeitschritt angeben, zu dem die entsprechende Größe betrachtet werden soll.

$$\underline{\mathbf{F}} \cdot \frac{\vec{h}^{n+1} - \vec{h}^{n}}{\Delta t} + \underline{\mathbf{K}} \cdot (\omega \cdot \vec{h}^{n+1} + (1-\omega) \cdot \vec{h}^{n}) = \omega \cdot \vec{r}^{n+1} + (1-\omega) \cdot \vec{r}^{n} \tag{3.27}$$

Falls $\omega = 0$ gesetzt wird, ist das benutzte Differenzenschema ein explizites Schema mit Vorwärtsdifferenzen. Für $\omega = 1$ ergibt sich ein implizites Differenzenschema mit Rückwärtsdifferenzen. Für den Fall $\omega = 0{,}5$ wird von einem Crank-Nicolson- oder zentralen Differenzenschema gesprochen.

Gleichung 3.27 kann nun sukzessive für alle Zeitschritte gelöst werden, wenn für $t = 0$ die Anfangsbedingungen eingesetzt werden. Zu jedem Zeitschritt kann der Vektor $\vec{h}^{n+1}$ berechnet werden, in dem der schon berechnete Vektor $\vec{h}^{n}$ benutzt wird. Eine Umformung von Gleichung 3.27 liefert die folgende Gleichung, die wieder eine Matrixgleichung mit dem Lösungsvektor $\vec{h}^{n+1}$ darstellt.

$$\underline{\mathbf{A}} \cdot \vec{h}^{n+1} = \vec{g}^{n} \tag{3.28}$$

In Gleichung 3.28 entspricht $\underline{\mathbf{A}}$ einer $[n \times n]$ Matrix und $\vec{g}^{n}$ einem Vektor der Dimension n.

$$\underline{\mathbf{A}} = \frac{1}{\Delta t} \cdot \underline{\mathbf{F}} + \omega \cdot \underline{\mathbf{K}} \tag{3.29}$$

$$\vec{g}^{n} = (\frac{1}{\Delta t} \cdot \underline{\mathbf{F}} + (1-\omega) \cdot \underline{\mathbf{K}}) \cdot \vec{h}^{n} + \omega \cdot \vec{r}^{n+1} + (1-\omega) \cdot \vec{r}^{n} \tag{3.30}$$

Die Lösung der Matrixgleichung 3.28 entspricht der Lösung eines linearen Gleichungssystems mit n Gleichungen und n Unbekannten. Die Schwierigkeit dieser Gleichung liegt jedoch darin, daß die Matrizen $\underline{\mathbf{F}}$ und $\underline{\mathbf{K}}$ indirekt zeitabhängig sind und aktualisiert werden müssen. Der Grund hierfür liegt darin, daß die Bodeneigenschaften im allgemeinen eine Funktion des hydrostatischen Potentials, also des Lösungsvektors $\vec{h}^n$, sind. Es wäre vorstellbar, die Matrizen immer am Ende eines Zeitschrittes zu aktualisieren. Das Problem bei diesem Vorgehen ist aber, daß das Gleichungssystem für einige Simulationen mit bestimmten Eingangsparametern instabil werden kann. Damit diese Instabilitäten vermieden werden können, müssen die beiden Matrizen immer mit dem Lösungsvektor des aktuellen Zeitschrittes berechnet werden. Es ergibt sich damit eine implizite Matrixgleichung, die jedoch immer stabil ist.

$$\underline{\mathbf{A}}^{n+1} \cdot \vec{h}^{n+1} = \vec{g}^n \tag{3.31}$$

Um die implizite Matrixgleichung numerisch lösen zu können, wird im Programm TRACE die Picard-Methode (Methode der sukzessiven Approximationen) verwendet. Mit dieser Methode kann iterativ zu jedem Zeitschritt $n+1$ Gleichung 3.31 gelöst werden. Hierzu wird in jedem Iterationsschritt ein lineares Gleichungssystem aufgestellt, das dann mit einem Lösungsverfahren, im Programm TRACE ist dies das Verfahren der konjugierten Gradienten, gelöst werden kann. In Gleichung 3.32 sollen die Hochindizes n den zu berechnenden Zeitschritt und p den Iterationsschritt des Picard-Verfahrens angeben.

$$\underline{\mathbf{A}}^{n+1,p} \cdot \vec{h}^{n+1,p+1} = \vec{g}^n \tag{3.32}$$

Die Matrix $\underline{\mathbf{A}}$ kann im $(p+1)$-ten Iterationsschritt bezüglich des Picard-Verfahrens mit dem schon berechneten Lösungsvektor $\vec{h}^{n+1,p}$ für den $(n+1)$-ten Zeitschritt aufgestellt werden. Eine Beschreibung des Picard-Verfahrens findet sich in [HUY83]. Die Picard'sche Iteration wird abgebrochen, wenn die Bedingung entsprechend Gleichung 3.33 erfüllt ist.

$$2 \cdot \frac{\left|\vec{h}^{n+1,p+1} - \vec{h}^{n+1,p}\right|}{\left|\vec{h}^{n+1,p+1}\right| + \left|\vec{h}^{n+1,p}\right|} \leq \alpha \tag{3.33}$$

Im folgenden soll die Implementierung der beschriebenen numerischen Verfahren im Programm TRACE betrachtet werden. Hierzu zeigt Abbildung 3.2 die sequentielle Programmstruktur von TRACE. Es soll an dieser Stelle noch einmal darauf hingewiesen werden, daß nur der Programmteil betrachtet wird, der zur Berechnung des Grundwasserflusses benötigt wird.

Nachdem die Dateien mit den Eingangsparametern eingelesen worden sind, werden zunächst vom Programm TRACE die Bodeneigenschaften berechnet. Grundlegend lassen sich dann zur Berechnung des Grundwasserflusses drei ineinanderliegende Schleifen in der Abbildung 3.2 erkennen. Die äußere Schleife ist die Zeitschleife. Hier wird mit Hilfe der Finiten-Differenzen-Methode der insgesamt zu simulierende Zeitbereich in gleichgroße Zeitintervalle zerlegt, die einzeln berechnet

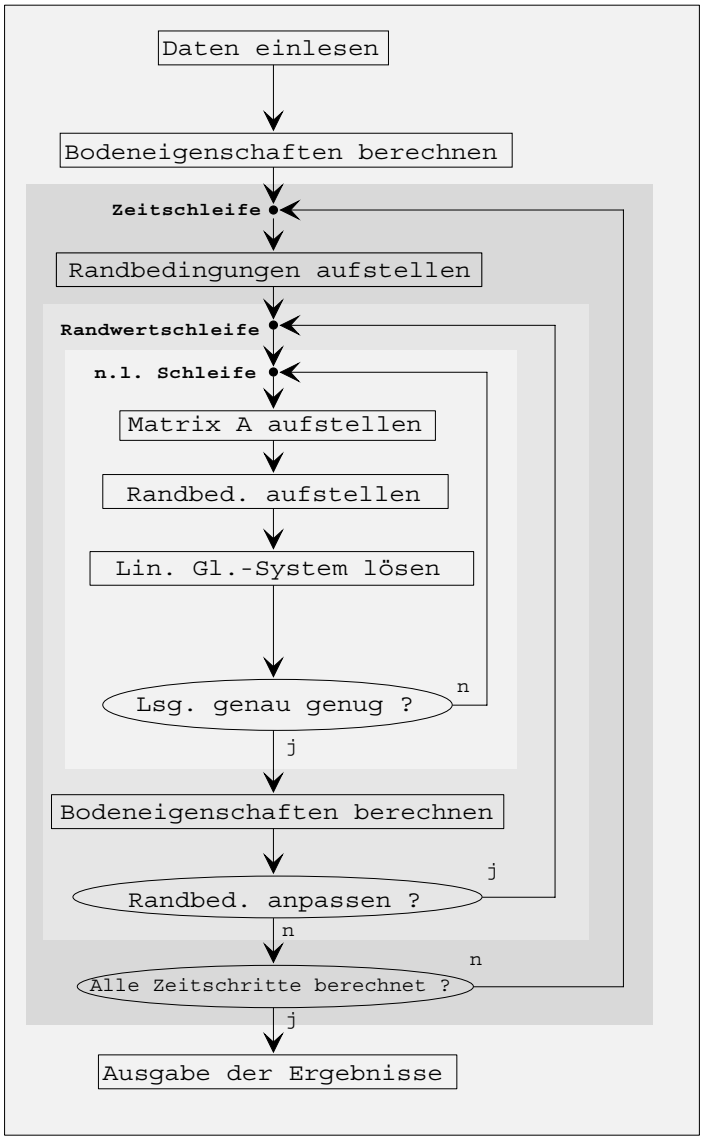


Abbildung 3.2: Sequentielle Struktur des Programms TRACE

werden können. Für jeden Zeitschritt werden zunächst die Randbedingungen neu berechnet.

Die mittlere Schleife ist die Randwertschleife. Mit dieser Schleife können die Randbedingungen angepaßt werden, die sich entweder in Abhängigkeit der Zeitschritte oder in Abhängigkeit der berechneten Lösung ändern können.

Die innere Schleife ist die nichtlineare (n. l.) Schleife, die durch Anwendung des Picard-Verfahrens die implizite Matrixgleichung entsprechend Gleichung 3.31 linearisiert. Hierzu muß zunächst für jeden Iterationsschritt die Matrix $\underline{\mathbf{A}}$ zusammengesetzt werden. Danach werden die Randbedingungen aufgestellt, damit die rechte Seite der Matrixgleichung (siehe Gleichung 3.32) berechnet werden kann. Das resultierende lineare Gleichungssystem wird dann mit dem Verfahren der konjugierten Gradienten gelöst. Wenn das Abbruchkriterium, entsprechend Gleichung

3.33, für den Lösungsvektor erfüllt ist, wird die innere Schleife verlassen. Da sich die Bodeneigenschaften in Abhängigkeit des Lösungsvektors geändert haben können, müssen diese erneut berechnet werden.

Falls sich eine Änderung der Randbedingungen ergeben hat, muß die innere Schleife erneut ausgeführt werden. Wenn alle Zeitschritte berechnet worden sind, wird die äußere Schleife verlassen und die Berechnungsergebnisse in eine Datei ausgegeben.

Die in diesem Abschnitt vorgestellte Struktur des Programms TRACE konnte nur schematisch dargestellt werden. Eine ausführliche Beschreibung der numerischen Verfahren zur Implementierung der Modellgleichungen im Programm TRACE findet sich in [VER94].

3.4.2 Datenstrukturen

Im Zentrum der Variablendeklarationen für das Programm TRACE stehen eine Vielzahl von Feldern, die die zu berechnenden Variablen und die hierfür erforderlichen Parameter beinhalten. Die Größe der Felder entspricht sehr oft der Gesamtzahl der FE-Knoten oder FE-Elemente, die das zu simulierende Gebiet beschreiben. Alle für die Berechnung wichtigen Felder sind in COMMON-Blöcken gespeichert, so daß diese Felder in allen Routinen als globale Variablen zur Verfügung stehen können. Da FORTRAN 77 keine dynamische Speicherverwaltung zuläßt, müssen die Felder schon zur Übersetzungszeit dimensioniert werden. Ein Nachteil dieses Vorgehens liegt darin, daß bei einer Änderung der Größe des Simulationsgebietes die Felderdimensionierungen im Quell-Code geändert werden müssen. Hierdurch wird eine erneute Übersetzung des Programms erforderlich.

Durch die gewachsene Programmstruktur sind die Variablendefinitionen unübersichtlich strukturiert, so daß die Anpassung von TRACE an ein neues Modellgebiet fehleranfällig ist. Zudem müssen Änderungen an unterschiedlichen Stellen im Quell-Code vorgenommen werden, so daß eine Programmanpassung für einen Benutzer, der mit dem Quell-Code nicht vertraut ist, nur schwer durchzuführen ist.

Auffallend an der Datenstruktur von TRACE ist, daß sehr viele Variablen nur ein einziges Mal gesetzt werden und dann über die gesamte Programmlaufzeit nur noch für Berechnungen oder Abfragen ausgewertet werden. Eine Kapselung von zusammengehörenden Daten in RECORD-Strukturen ist nicht vorgesehen, da solche Datentypen nicht von FORTRAN 77 unterstützt werden. Auch bei der Namensgebung der Variablen wird weitgehend der Standard von FORTRAN 77 eingehalten, der nur sechs Zeichen für Namen vorsieht.

3.4.3 Dateien mit Eingangsparametern

Die für den Programmteil zur Berechnung des Wasserflusses wichtigste INPUT-Datei ist eine umfangreiche ASCII-Datei mit Namen *WATDAT.hom* für homogene beziehungsweise *WATDAT.het* für heterogene Bodeneigenschaften. Der interne Aufbau der *WAT(er)-DAT(a)*-Datei besteht aus maximal 17 Datensätzen, die je nach Komplexität der Simulation verwendet werden können. In den Datensätzen sind

jeweils logisch zusammengehörende Daten gespeichert. Es existieren beispielsweise Datensätze für die Koordinaten der diskreten Gitterpunkte, für die Bodeneigenschaften, für die Anfangswerte, für die Randwerte und für die Quellen- und Senkenterme. Mehrere Datensätze enthalten auch Steuerparameter, mit denen die Programmausführung von TRACE beeinflußt werden kann. Die Zahl der maximal durchzuführenden Iterationen, die Ausgabe von Zwischenergebnissen und vergleichbare Parameter können über diese Datensätze vorgegeben werden. Die *WATDAT*-Datei kann weitgehend automatisch mit dem Programm INPGEN [KUH94] generiert werden. Die Größe der Datei kann je nach Simulation mehrere tausend bis zehntausend Zeilen erreichen. Obwohl das Editieren der Datei umständlich ist, bleibt die Möglichkeit, individuelle Anpassungen der Eingangsdaten vorzunehmen. Im Anhang A ist ein verkürzter Ausschnitt einer *WATDAT*-Datei angegeben. Eine ausführliche Beschreibung der Struktur und der für das Einlesen einzuhaltenden Syntax der *WATDAT*-Datei findet sich in [YEH87].

Neben der *WATDAT*-Datei benötigt das Programm TRACE vier weitere INPUT-Dateien. Mit Hilfe der Datei *CLIMAT* können variable Randbedingungen für die Grenzfläche Boden-Atmosphäre definiert werden. Die Datei *CROPDAT* liefert Parameter zur Berechnung des Einflusses von Pflanzenwachstum auf die Vorgänge im Boden. Die Datei *HYST.DAT* beschreibt die Materialeigenschaften des Bodens und des Grundwasserleiters. Die Datei *SOLDAT* beinhaltet alle für die Berechnung des Stofftransportes benötigten Parameter.

3.5 Grenzen der sequentiellen Programmversion

Da das Programm TRACE in FORTRAN 77 geschrieben ist und deshalb keine dynamische Speicherverwaltung möglich ist, ist die Zahl der FE-Knoten, die maximal berechnet werden können, durch die Größe des verfügbaren Arbeitsspeichers, beziehungsweise bei Systemen mit virtuellem Speicher durch die Programmlaufzeiten begrenzt. Einen Einfluß auf die Zahl der FE-Knoten in Abhängigkeit von dem zu simulierenden Boden hat die Korrelationslänge. Unter dem Begriff der Korrelationslänge versteht man den räumlichen Abstand, in dem sich einzelne Parameter, wie zum Beispiel die Permeabilität, beeinflussen können. Eine Vorgabe für die zu verwendende Knotenzahl ist, daß pro Korrelationslänge mindestens vier FE-Knoten angeordnet sein sollten [GEL86]. Für geologische Fragestellungen ist die Korrelationslänge in vertikaler Richtung sehr viel kleiner als in horizontaler Richtung. Dies liegt daran, daß sich in vertikaler Richtung häufiger die Bodeneigenschaften ändern als in horizontaler Richtung (vergleiche die einzelnen Bodenschichten in Abbildung 3.1).

Das ICG-4 betreibt ein Versuchsgelände zur Untersuchung der Ausbreitung von Schadstoffen im Grundwasser. Die Größe dieses Geländes beträgt $150\,m \times 50\,m \times 10\,m$ (L x B x T). Die horizontale Korrelationslänge kann mit 4 m, die vertikale Korrelationslänge mit 0,4 m abgeschätzt werden. Die bei den Versuchen experimentell ermittelten Ergebnisse sollen mit den Berechnungen des Programms TRACE verglichen werden. Um dieses Gelände zu simulieren, werden für eine horizontale

Gitterebene ca. 7500 FE-Knoten und für die vertikale Richtung ungefähr 100 FE-Knoten bei den vorgegebenen Korrelationslängen benötigt. Die Größe des Simulationsgebietes würde bei dieser Rechnung 750000 FE-Knoten betragen.

Bei bisherigen Programmläufen auf dem iPSC/860 konnten 10000 FE-Knoten, beziehungsweise auf der CRAY X-MP/416 50000 FE-Knoten pro eingesetztem Prozessor berechnet werden. Durch den Einsatz eines massiv-parallelen Rechnersystems mit verteiltem Arbeitsspeicher kann, bei geeigneter Anpassung von TRACE auf diese Rechnerstruktur, durch Ausnutzen einer bestimmten Zahl von Prozessoren der gesamte zur Verfügung stehende Arbeitsspeicher durch Kombination der lokalen Speicher so vergrößert werden, daß das vollständige Simulationsgebiet geladen und berechnet werden kann.

Um realistische Simulationen durchführen zu können, wird eine große Zahl an Zeitschritten benötigt. Dies bedeutet einen hohen Berechnungsaufwand, der auf sequentiellen Rechnern eine lange Programmlaufzeit verursacht. Durch eine Parallelisierung von TRACE läßt sich die Programmlaufzeit verkürzen.

Ein weiteres Problem der sequentiellen Programmversion ist die Schnittstelle zum Anwender. Die Anpassung des Programms an unterschiedliche Simulationsgebiete (Vorgabe der Simulationsgröße und der Daten für die Eingabedateien) ist unübersichtlich, wobei eine Veränderung des Quell-Codes an mehreren, nicht zusammenhängenden Stellen erforderlich wird. Auch die Ausgabe der Berechnungsergebnisse in geeigneter Form ist im Programm TRACE noch nicht implementiert.

Kapitel 4

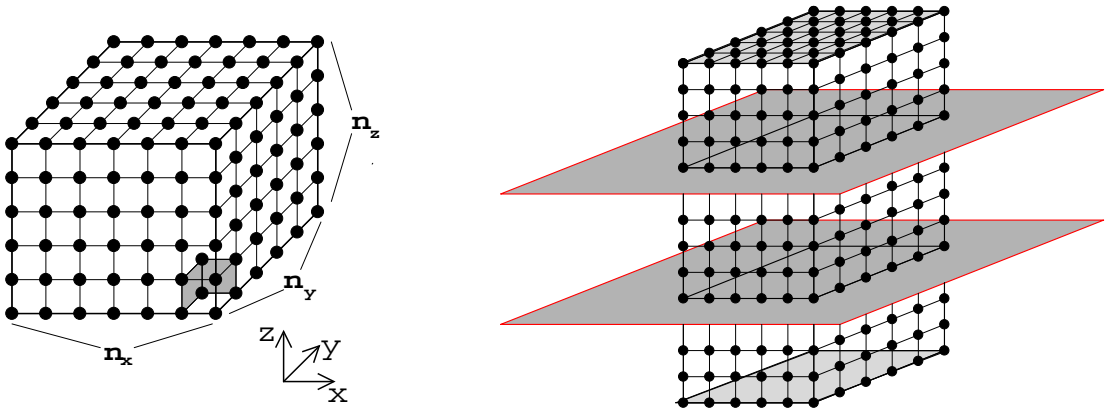
Die Zerlegung des FE-Gitters

4.1 Struktur des FE-Gitters

Um die Finite-Elemente-Methode (FE-Methode) zur Lösung partieller Differentialgleichungen im Ortsraum nutzen zu können, wird der Definitionsbereich der Gleichungen diskretisiert. Es entsteht ein dreidimensionales Gitter, bei dem den einzelnen Gitterpunkten jeweils drei Raumkoordinaten zugeordnet werden. Im folgenden soll die Struktur eines diskreten Gitters betrachtet werden, welches im Programm TRACE zur Berechnung des Wasserflusses in porösen, heterogenen Medien verwendet wird. Der Aufbau des Gitters wird durch die folgenden Regeln charakterisiert:

- Das gesamte Gitter hat die Struktur eines Hexaeders.
- In einer Richtung (x-, y- oder z-Richtung) muß die Anzahl der Gitterpunkte konstant sein.
- In unterschiedlichen Richtungen kann die Zahl der Gitterpunkte (n_x, n_y, n_z) variieren.
- Der räumliche Abstand zwischen den Gitterpunkten in einer Richtung kann unterschiedlich sein.
- Jeder innere Gitterpunkt ist immer mit sechs anderen Gitterpunkten verbunden.
- Gitterpunkte auf den Rändern haben im allgemeinen fünf Verbindungen zu benachbarten Punkten, Eckknoten besitzen drei Verbindungen.
- Durch Stauchen, Dehnen oder Verdrehen des Gitters kann dieses immer in eine hexaedrische Gestalt überführt werden, wobei jedoch die Verbindungen zwischen den Gitterpunkten erhalten bleiben müssen.

Ein so beschriebenes Gitter gehört zur Klasse der strukturierten Gitter. In Abbildung 4.1 (a) ist die einfachste Form eines derartigen Gitters dargestellt. Die Winkel zwischen den Koordinatenachsen können unterschiedlich sein. Sie müssen



(a) allg. Gitterstruktur (b) Schnittebenen

Abbildung 4.1: Das Finite-Elemente-Gitter

nicht, wie in der Abbildung dargestellt, 90 Grad betragen. Durch die FE-Methode werden Elemente erzeugt, die mit bestimmten Eigenschaften, beispielsweise Materialeigenschaften, verknüpft werden können. Beim Programm TRACE definieren immer acht Knoten ein Element. In der Abbildung 4.1 (a) ist ein Element in der vorderen rechten Ecke des Gitters besonders hervorgehoben. Die Elemente haben eine hexaedrische Form. Zwei benachbarte Elemente besitzen immer vier gemeinsame Knoten, die in einer Ebene liegen. Für die in diesem Kapitel folgenden Betrachtungen über die Zerlegung des Gitters reichen zur vollständigen Charakterisierung die Gitterparameter n_x, n_y und n_z, das heißt die Anzahl der FE-Knoten in den drei Raumrichtungen, aus.

$$N_{NP} = n_x \cdot n_y \cdot n_z \tag{4.1}$$

$$N_{EL} = (n_x - 1) \cdot (n_y - 1) \cdot (n_z - 1) \tag{4.2}$$

Mit den Größen N_{NP} (Number of Nodal Points) und N_{EL} (Number of Elements) werden die für dieses Gitter maßgeblichen Werte angegeben.

Um das FE-Gitter programmintern beschreiben zu können, sind die FE-Knoten und die FE-Elemente jeweils mit Nummern versehen. Die Numerierung der FE-Knoten ist in Abbildung 4.2 dargestellt. Es soll ein Gitter betrachtet werden, bei dem in x-Richtung 7, in y-Richtung 3 und in z-Richtung mindestens 3 FE-Knoten auftreten.

Die FE-Knoten werden zunächst immer in Ebenen mit $z = const.$ numeriert. Der Knoten mit der Nummer 1 liegt in der vorderen, linken, unteren Ecke des Gesamtgebietes. Die folgenden Knoten sind in Richtung der x-Achse angeordnet. Wenn der rechte Rand des Gebietes erreicht ist (Knotennummer: 7), dann folgt der nächste Knoten wieder am linken Rand des Gebietes, jedoch um einen Knoten nach

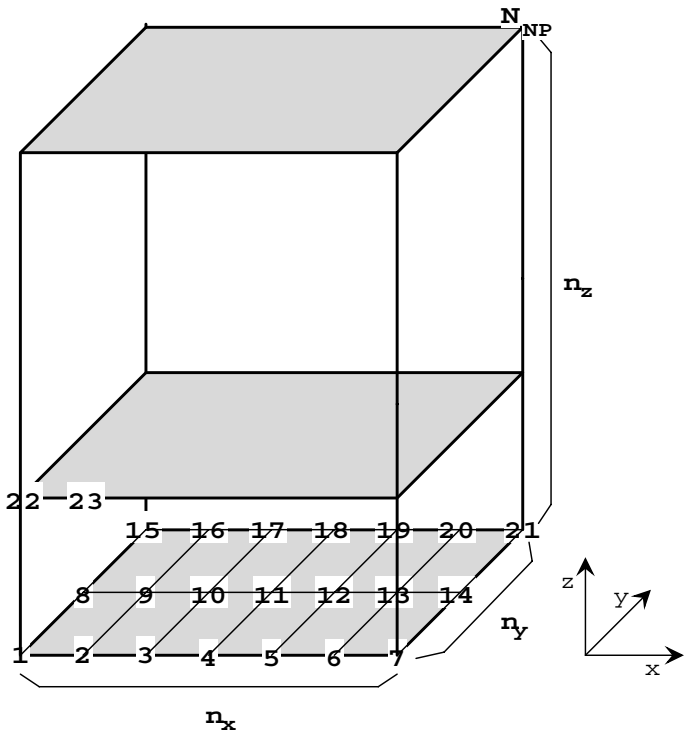


Abbildung 4.2: Numerierung der FE-Knoten

hinten (in y-Richtung) verschoben (Knotennummer: 8). Auf den letzten Knoten der unteren Ebene (Knotennummer: $n_x \cdot n_y = 21$) folgt der erste Knoten der nächsten Ebene in z-Richtung (Knotennummer 22). Dieser Knoten besitzt die gleichen x-, y-Koordinaten wie der Knoten mit der Nummer 1. Der Knoten mit der höchsten Nummer (Knotennummer: N_{NP}) befindet sich in der obersten Ebene, hinten rechts. Die Numerierung der FE-Elemente wird entsprechend durchgeführt.

4.2 Anforderungen an die Zerlegung

Um das Gesamtgitter parallel bearbeiten zu können, wird beim Programm TRACE das Schwarz'sche Verfahren in der additiven Variante eingesetzt. Hierzu muß das Gitter in Teilgitter zerlegt werden, die einzelnen Prozessoren zugeordnet werden können. Das Schwarz'sche Verfahren verlangt, daß die Teilgitter überlappen. Dieser Überlapp soll in den drei Raumrichtungen jeweils gleich groß sein.

An die Zerlegung des gesamten Gitters werden zusätzliche Forderungen gestellt, die die Rechenzeit des Programms und den Aufwand für die Kommunikationen während der Programmlaufzeit beeinflussen. Das Programm TRACE berechnet mit Hilfe des diskretisierten Gitters sowohl den Wasserfluß als auch den Stofftransport. In welchen Teilbereichen des Gitters der Rechenaufwand groß ist, hängt von einer Vielzahl von Eingangsparametern ab und kann bei unterschiedlichen Simulationen variieren. Da eine a priori Abschätzung nicht angegeben werden kann und eine dynamische Lastverteilung zur Zeit nicht vorgesehen ist, wird angenommen, daß

der Rechenaufwand für jeden Gitterpunkt gleich ist. Eine ausgewogene Lastverteilung kann bei den getroffenen Annahmen dann erreicht werden, wenn das gesamte Gitter gleichmäßig auf alle Prozessoren aufgeteilt wird. Die Zahl der Knoten, die zu den einzelnen Teilgittern gehören, soll deshalb annähernd gleich groß sein. Die entstehenden Teilgitter sollen wieder die Form eines Hexaeders besitzen, der bei gleicher Knotenzahl in den einzelnen Raumrichtungen in einen Würfel übergeht. Im einzelnen lassen sich folgende Bedingungen an die Zerlegung stellen:

- Es sollen möglichst viele Prozessoren eingesetzt werden, um die Gesamtrechenzeit zu minimieren.
- Jeder Prozessor soll genau ein Teilgebiet zur Berechnung zugewiesen bekommen.
- Der Arbeitsspeicher der Prozessoren soll möglichst gut ausgelastet werden.
- Das gesamte Simulationsgebiet soll statisch verteilt werden, was bedeutet, daß die Aufteilung beim Programmstart durchgeführt wird und während der Programmabarbeitung erhalten bleibt.
- Die Zahl der Pseudoknoten, d.h. der Knoten, die durch die Gitterzerlegung auf neu entstandenen Rändern der Teilgebiete liegen, soll gering sein, um den Aufwand für die Kommunikation während der Rechnung klein zu halten.
- Die Zahl der FE-Knoten, die in den Überlappgebieten liegen, soll minimal sein, da diese Knoten redundant in jedem Teilgebiet berechnet werden müssen.
- Die Zahl der Gitterpunkte, die jeder Prozessor zur Berechnung bekommt, soll annähernd gleich sein, um eine möglichst gute Lastverteilung zu erhalten.

Um eine Entscheidung über die Art der Zerlegung des FE-Gitters im Zusammenhang mit den oben aufgestellten Bedingungen treffen zu können, stehen neben den Gitterparametern n_x, n_y und n_z auch die Zahl der einzusetzenden Prozessoren C und die Größe des Überlapps L zur Verfügung. Der Überlapp gibt die Zahl der Elemente an, die im Überlappbereich liegen sollen. Eindimensional betrachtet bedeutet dies, daß bei einem Überlapp von einem Element zwei Knoten im Überlappbereich liegen müssen. Im folgenden soll von einer konstant vorgegebenen Prozessoranzahl C ausgegangen werden.

$$C = c_x \cdot c_y \cdot c_z \tag{4.3}$$

Entsprechend Gleichung 4.3 wird die Zahl der Prozessoren in die drei ganzzahligen Faktoren c_x, c_y und c_z zerlegt. Um das Gesamtgebiet in einzelne Teilgebiete aufzuteilen, werden Schnitte durch das Gesamtgebiet gelegt. Ein Schnitt muß entsprechend Abbildung 4.1 (b) durch das vollständige Gebiet gezogen werden. Eine Schnittebene liegt immer in der Ebene, in der die FE-Knoten liegen, was bedeutet, daß die Elemente des Gitters erhalten bleiben. Hieraus ergibt sich, daß die Knoten, die in der Schnittebene liegen, immer zu den Gebieten beiderseits des Schnittes

gehören. Die Folge hiervon ist, daß sich die Zahl der Knoten im Überlappbereich verdoppelt. Die um 1 erniedrigten Faktoren c_i mit $i = x, y, z$ sollen die Zahl der Schnitte in den einzelnen Raumrichtungen angeben. In Abbildung 4.1 (b), in dem das Gebiet in z-Richtung zweimal geschnitten wurde, wäre dementsprechend $c_z = 3$. Bei diesem Vorgehen werden hexaedrische Teilgebiete erzeugt, die zusammenhängend sind. Die Berührungsfläche zweier Teilgitter erstreckt sich über den gesamten Querschnitt der Gitter, es gibt keinen Versatz.

Wie schon erwähnt, läßt sich das FE-Gitter vollständig durch die Gitterparameter n_x, n_y und n_z beschreiben. Der Überlapp L soll für jeden Schnitt unabhängig von der Raumrichtung gelten. Die Bestimmung der Schnittebenen soll nun exemplarisch für die x-Richtung aufgezeigt werden. Für die anderen beiden Richtungen gilt das Entsprechende. In Abhängigkeit von der Zahl der Schnitte $(c_x - 1)$ und dem vorgegebenen Überlapp (L) soll N_{eff} die effektive Knotenanzahl in der x-Richtung angeben.

$$N_{eff} = n_x + (c_x - 1) \cdot (L + 1) \tag{4.4}$$

Diese Zahl, angegeben in Gleichung 4.4, gibt die Gesamtzahl der FE-Knoten in der x-Richtung an; sie entspricht der Summe der FE-Knoten aller Teilgebiete in der x-Richtung. Durch die Summenbildung wird berücksichtigt, daß die in den Überlappgebieten liegenden FE-Knoten doppelt gewertet werden. Um die Länge der Teilgebiete in der x-Richtung angeben zu können, wird eine Integer-Division und eine Restberechnung durchgeführt (siehe Gleichung 4.5).

$$\begin{aligned} G &= N_{eff}\ DIV\ c_x \\ R &= N_{eff}\ MOD\ c_x \end{aligned} \tag{4.5}$$

Jedes Teilgebiet besitzt in x-Richtung G FE-Knoten, falls der Rest R der Division ungleich Null ist, werden die ersten R Gebiete jeweils um einen Knoten erweitert. Bei diesem Vorgehen sind die Längen der einzelnen Teilgebiete um maximal einen Knoten unterschiedlich. Dreidimensional betrachtet bedeutet dies, daß sich die Teilgebiete um höchstens eine Knotenebene pro Raumrichtung unterscheiden.

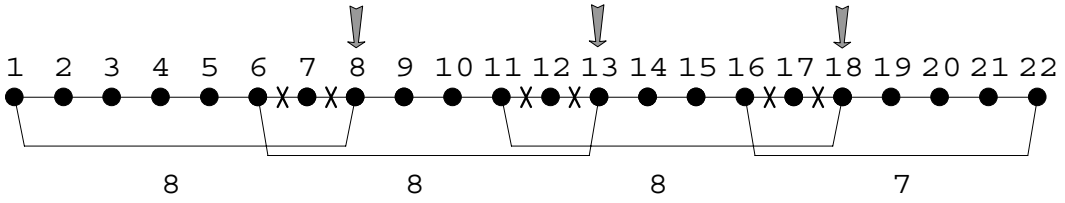


Abbildung 4.3: 1D-Betrachtung der Zerlegung

Abbildung 4.3 zeigt ein eindimensionales Gitter, das aus $n_x = 22$ FE-Knoten besteht. Diese Knoten sollen bei einem vorgegebenem Überlapp von 2 Elementen $(L = 2)$ in 4 zusammenhängende Teilgitter $(c_x = 4)$ zerlegt werden. Die effektive Knotenanzahl für diese Konfiguration beträgt $N_{eff} = 22 + 3 \cdot 3 = 31$. Aus Gleichung

4.5 folgt $G = 7$ und $R = 3$, so daß dieses Gitter in die Teilgitter mit den Größen 8-8-8-7 zerlegt wird. Um die Zerlegung durchzuführen, werden zunächst von links die ersten acht Knoten abgezählt. Der achte Knoten gibt die Ebene an, in der ein Schnitt durchgeführt werden muß. Die Schnitte sind in der Abbildung durch die Pfeile über der Knotennumerierung angedeutet. In Abhängigkeit vom Überlapp wird dann um L Elemente, entsprechend der Kreuze in Abbildung 4.3, zurückgegangen. Von Knoten 6 an werden wieder acht Knoten abgezählt, so daß Knoten 13 wieder eine Schnittebene angibt. Dieser Vorgang wird bis zum Ende des Gebietes fortgesetzt.

Für die Verwaltung der Teilgebiete wird dieselbe Numerierung verwendet, die bei der Bezeichnung der FE-Knoten beziehungsweise der FE-Elemente eingeführt wurde. Es ist hierbei jedoch der Unterschied zwischen der lokalen und der globalen Numerierung zu beachten. Bei jedem Teilgebiet beginnt die lokale Knoten- beziehungsweise Elementnumerierung mit dem Wert Eins, so daß aus der lokalen Numerierung ohne weitere Informationen nicht mehr auf die globale Numerierung geschlossen werden kann.

4.3 Die K-Funktion

Für die Faktorisierung der Prozessorzahl C gibt es eine endliche Zahl möglicher Kombinationen c_x, c_y und c_z. Im folgenden soll der Einfluß der Wahl einer bestimmten Kombination auf das Laufzeitverhalten des Programms TRACE genauer untersucht werden. Dazu wird ein zweidimensionaler Ausschnitt eines Gitters betrachtet, in dem jeweils ein Schnitt in jeder Richtung ausgeführt wurde. Der Überlappbereich entspricht zwei Elementen. In Abbildung 4.4 sind vier Teilgebiete Ω_1 bis Ω_4 zu erkennen, die in der Mitte einen gemeinsamen Überlappbereich besitzen. Dieser Bereich ist im Vordergrund der Abbildung vergrößert dargestellt, wobei die Kreise jeweils den Knoten des Gitters entsprechen. In den Gitterpunkten ist die Zahl der Zugehörigkeiten dieser Punkte zu den einzelnen Teilgebieten eingetragen. Es lassen sich Gruppen von Gitterpunkten zusammenfassen, die entweder zu einem, zu zwei oder zu vier Teilgebieten gehören. Dies wird in der Abbildung durch verschiedene Grauschattierungen angedeutet. Bei einem in drei Dimensionen geschnittenen Gesamtgitter können bis zu acht Teilgebiete einen gemeinsamen Überlappbereich besitzen, so daß Gitterpunkte, die in diesem Bereich liegen, achtfach vorkommen und berechnet werden müssen.

Ein Vergleich der verschiedenen Kombinationen für die Faktorisierung der Prozessorzahl ergibt, daß die Zahl der Knoten, die in den gemeinsamen Überlappbereichen liegen, stark variieren kann. Knoten, die in mehreren Teilgebieten liegen und dementsprechend auf verschiedenen Prozessoren mehrfach vorhanden sind und berechnet werden müssen, besitzen für den Fortgang der Berechnung keine weiterführenden Informationen, da sie jeweils im Rahmen der numerischen Genauigkeit zu denselben Berechnungsergebnissen führen. Um eine Entscheidung über die optimale Kombination in Zusammenhang mit dem Mehraufwand, der durch die überlappende Zerlegung des Gitters hervorgerufen wird, treffen zu können, wird ein Kriterium definiert, welches aus dem Quotienten der Zahl der tatsächlich zu

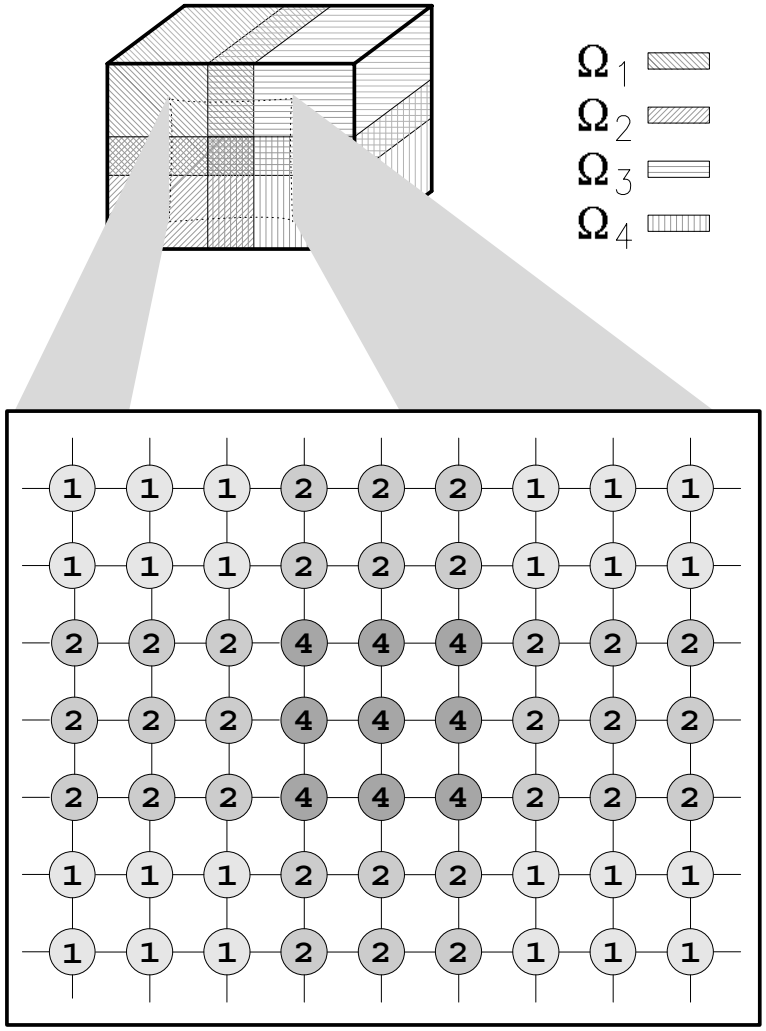


Abbildung 4.4: Häufigkeit des Auftretens von einzelnen FE-Knoten in einem Überlappungsbereich von vier Teilgebieten

berechnenden FE-Knoten und der ursprünglichen Gesamtzahl der FE-Knoten besteht. Diese Funktion, im folgenden auch K-Funktion genannt, wird in Gleichung 4.6 angegeben.

$$K = \frac{\textit{Zahl tatsächlich zu berechnender FE-Knoten}}{\textit{Gesamtzahl FE-Knoten}} \tag{4.6}$$

Wenn das Gesamtgebiet nur mit einem Prozessor berechnet werden soll, also keine Gebietszerlegung durchgeführt wird, ist die Zahl der tatsächlich zu berechnenden FE-Knoten gleich der Gesamtzahl der FE-Knoten. In diesem Fall gibt die K-Funktion den Wert 1 an, was bedeutet, daß der Rechenaufwand gleich dem ursprünglichen Aufwand ist. Falls mehrere Prozessoren eingesetzt werden sollen, ist der Wert der K-Funktion immer größer als 1, da durch die Gebietszerlegung Überlappbereiche entstehen, die zu einem erhöhten Rechenaufwand führen. Eine theoretische Grenze für den Maximalwert, den die K-Funktion bei einer dreidimensionalen

Zerlegung annehmen kann, ist 8. Dies ist jedoch nur ein theoretischer Wert, da dies bedeuten würde, daß alle FE-Knoten jeweils im gemeinsamen Überlappbereich von acht Teilgebieten liegen müßten.

Um die K-Funktion berechnen zu können, werden zunächst drei neue Größen eingeführt. Diese sind die Zahl der effektiven Überlappknoten (U_{eff}), die Zahl der inneren Knoten (I) und die Gesamtzahl (G) der FE-Knoten. Gleichung 4.7 gibt die K-Funktion, bestehend aus den neuen Größen, an.

$$K = \frac{U_{eff} + I}{G} \tag{4.7}$$

Die Zahl der effektiven Überlappknoten setzt sich aus der Zahl der Knoten zusammen, die in den Überlappbereichen liegen, jeweils multipliziert mit der Zahl der Zugehörigkeit der Knoten zu unterschiedlichen Teilgebieten. Die Zahl der inneren Knoten gibt die Knoten an, die nicht in den Überlappbereichen liegen. Die Gesamtzahl der FE-Knoten entspricht der Zahl der Knoten des ursprünglichen Gesamtgebietes. Um die Zahl der inneren Knoten berechnen zu können, wird die Größe U_{einf}, die Zahl der einfach gewerteten Überlappknoten, eingeführt. Damit kann für I der Zusammenhang nach Gleichung 4.8 angegeben werden.

$$I = G - U_{einf} \tag{4.8}$$

Zur Verdeutlichung sollen diese Größen für den ein-, den zwei- und den dreidimensionalen Fall berechnet werden.

1D-Betrachtung

Betrachtet werden soll Abbildung 4.5. In dieser Abbildung ist ein eindimensionales Gebiet dargestellt, welches aus $n_x = 10$ FE-Knoten besteht. Dieses Gebiet wird in drei Teilgebiete der Größen 5-5-4 mit einem Überlapp von $L = 1$ Element, entsprechend zwei FE-Knoten, zerlegt. Bei dieser Zerlegung werden zwei Schnitte durch das Gebiet gelegt, so daß $c = c_x = 3$ gilt. Über den einzelnen FE-Knoten ist jeweils die Zahl der Zugehörigkeiten dieser Knoten zu verschiedenen Teilgebieten angegeben.

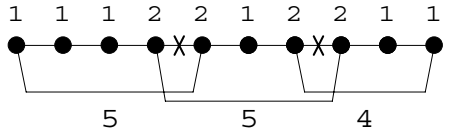


Abbildung 4.5: 1D-Gitterbetrachtung

Für diese einfache Anordnung ergeben sich für die gesuchten Größen die folgenden Werte, wie leicht durch Abzählen nachvollzogen werden kann:

$$G = 10 \quad U_{eff} = 8$$
$$U_{einf} = 4 \quad I = 6$$

Die Parameter lassen sich gemäß den Gleichungen 4.9 bis 4.12 angeben.

$$
\begin{aligned}
G &= n_x && (4.9)\\
U_{einf} &= (c_x - 1) \cdot (L + 1) && (4.10)\\
U_{eff} &= 2 \cdot (c_x - 1) \cdot (L + 1) \\
&= 2 \cdot U_{einf} && (4.11)\\
I &= G - U_{einf} \\
&= n_x - (c_x - 1) \cdot (L + 1) && (4.12)
\end{aligned}
$$

Hiermit ergibt sich für die gesuchte K-Funktion eine Funktion, wie sie in Gleichung 4.13 aufgestellt ist.

$$
\begin{aligned}
K &= \frac{U_{eff} + I}{G} \\
&= 1 + \frac{(c_x - 1) \cdot (L + 1)}{n_x} && (4.13)
\end{aligned}
$$

Der kleinste Wert, den die K-Funktion annehmen kann, ist 1. Dies entspricht, wie bereits erwähnt, dem Rechenaufwand für das Gebiet ohne überlappende Zerlegung ($c_x = 1$).

2D-Betrachtung

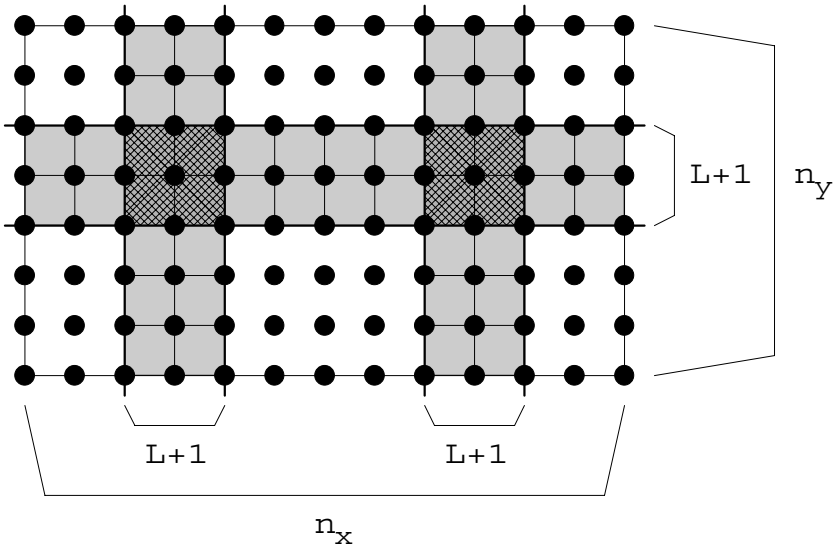


Abbildung 4.6: 2D-Gitterbetrachtung

In Abbildung 4.6 ist ein zweidimensionales Gitter dargestellt, welches in x-Richtung aus $n_x = 13$ und in y-Richtung aus $n_y = 8$ FE-Knoten besteht. Dieses Gitter wird nun in x-Richtung zweimal ($c_x = 3$) und in y-Richtung einmal ($c_y = 2$) geschnitten. Der Überlapp wird mit $L = 2$ Elementen, entsprechend 3 FE-Knoten,

gewählt. Die für die Berechnung der K-Funktion relevanten Größen können aus der Abbildung entnommen werden:

$$\begin{array}{rclrcl} G & = & 104 & U_{eff} & = & 174 \\ U_{einf} & = & 69 & I & = & 35 \end{array}$$

Bei diesem einfachen Beispiel fällt auf, daß sich der Berechnungsaufwand, hervorgerufen durch die überlappende Zerlegung, mehr als verdoppelt hat. Es müssen insgesamt $U_{eff} + I = 174 + 35$ FE-Knoten berechnet werden. Das Ausgangsproblem bestand jedoch nur aus $G = 104$ FE-Knoten.

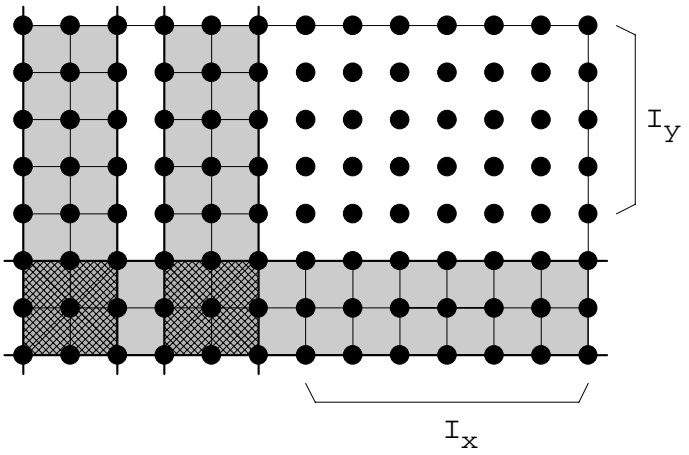


Abbildung 4.7: 2D-Gitterbetrachtung mit verschobenen Überlappungsbereichen

Für eine allgemeine Analyse der zweidimensionalen Gitterzerlegung soll Abbildung 4.7 betrachtet werden. In dieser Abbildung sind die Überlappbereiche jeweils in Richtung der Ränder des Gebietes verschoben worden. Die gitterspezifischen Parameter n_x und n_y, die Größe der Überlappbereiche und die Zahl der Knoten, die in den Schnittmengen der Überlappbereiche liegen, bleiben bei dieser Operation erhalten. Für die Bestimmung der Zahl der Knoten, die nicht in den Überlappbereichen liegen, ergibt sich jedoch nach der Verschiebung, daß diese Knoten in einem rechteckigen, zusammenhängenden Bereich liegen und deren Anzahl einfach berechnet werden kann. Die Zahl der Knoten, die in dem Rechteck liegen, kann durch das Produkt von I_x und I_y angegeben werden, wobei I_x und I_y entsprechend Abbildung 4.7 die Zahlen der Knoten angeben, die auf den Rändern des Rechtecks liegen.

$$I = I_x \cdot I_y \tag{4.14}$$

In x-Richtung betrachtet ist I_x die Differenz der Gesamtknoten n_x in x-Richtung und der Knoten in den Überlappbereichen. Gleichung 4.15 gibt die Zahl der inneren Knoten in x-Richtung an.

$$I_x = n_x - (L+1) \cdot (c_x - 1) \tag{4.15}$$

Für die y-Richtung kann derselbe Zusammenhang angegeben werden.

$$I_y = n_y - (L+1) \cdot (c_y - 1) \tag{4.16}$$

Entsprechend Gleichung 4.14 kann nun die Zahl I der inneren Knoten angegeben werden.

$$\begin{aligned} I &= n_x \cdot n_y \\ &\quad - (L+1) \cdot \Big\{ (c_x - 1) \cdot n_y + (c_y - 1) \cdot n_x \Big\} \\ &\quad + (L+1)^2 \cdot (c_x - 1) \cdot (c_y - 1) \end{aligned} \tag{4.17}$$

Die Gesamtzahl G der FE-Knoten des Gitters kann sofort berechnet werden.

$$G = n_x \cdot n_y \tag{4.18}$$

Zur Bestimmung der Zahl U_{eff} der effektiven Überlappknoten müssen die Knoten betrachtet werden, die in den Überlappbereichen liegen. Hierzu werden diese Bereiche getrennt voneinander in der x- bzw. in der y-Richtung untersucht. In der x-Richtung existieren $c_x - 1$ Überlappbereiche mit jeweils $(L+1) \cdot n_y$ FE-Knoten. Jeder Knoten, der in einem Überlappbereich liegt, gehört zu genau zwei Teilgebieten, so daß die Zahl der effektiven Überlappknoten in Bezug auf Überlappbereiche in der x-Richtung angegeben werden kann.

$$2 \cdot (c_x - 1) \cdot (L+1) \cdot n_y$$

Für die y-Richtung kann genau dieselbe Betrachtung durchgeführt werden, wobei die Knoten in den Überlappbereichen ebenfalls genau zu zwei Teilgebieten gehören.

$$2 \cdot (c_y - 1) \cdot (L+1) \cdot n_x$$

Bei einer gemeinsamen Betrachtung beider Richtungen gehören die Knoten, die in der Schnittmenge der Überlappbereiche liegen, zu vier Teilgebieten, während die anderen Knoten der Überlappbereiche genau zu zwei Teilgebieten gehören. Da die Zahl der Knoten der Überlappbereiche bei der nach Richtungen getrennten Betrachtung schon mit dem Faktor 2, das heißt der Zahl der Zugehörigkeit der Knoten zu Teilgebieten, gewichtet wurde, ergibt sich für die Gesamtzahl U_{eff} der effektiven Überlappknoten eine einfache Addition der Teilergebnisse. Gleichung 4.19 gibt diese Größe an.

$$U_{eff} = 2 \cdot (L+1) \cdot \Big\{ (c_x - 1) \cdot n_y + (c_y - 1) \cdot n_x \Big\} \tag{4.19}$$

Einsetzen der berechneten Parameter in die mit Gleichung 4.7 beschriebene K-Funktion und Umformen dieses Ausdrucks führt zu Gleichung 4.20.

$$\begin{aligned} K &= 1 + \frac{(L+1) \cdot \Big\{ (c_x - 1) \cdot n_y + (c_y - 1) \cdot n_x \Big\}}{n_x \cdot n_y} \\ &\quad + \frac{(c_x - 1) \cdot (c_y - 1) \cdot (L+1)^2}{n_x \cdot n_y} \end{aligned} \tag{4.20}$$

Da die Zahl C der Prozessoren fest vorgegeben wird und $C = c_x \cdot c_y$ gelten muß, kann $c_y = C/c_x$ in die K-Funktion eingesetzt werden. Die K-Funktion besteht dann aus den bekannten Größen n_x, n_y, L, C und der Zahl c_x, die die optimale Aufteilung des Gebietes bestimmt.

In Abbildung 4.8 ist die K-Funktion über der Zahl der Prozessoren, die in x-Richtung eingesetzt werden sollen, aufgetragen. Den Berechnungen liegt das Beispiel aus Abbildung 4.6 zugrunde. Die Zahl der insgesamt einzusetzenden Prozessoren wird konstant auf $C = 6$ gehalten, so daß für die Faktorisierung von C nur die Kombinationen $1 \cdot 6$, $2 \cdot 3$, $3 \cdot 2$ und $6 \cdot 1$ auftreten können.

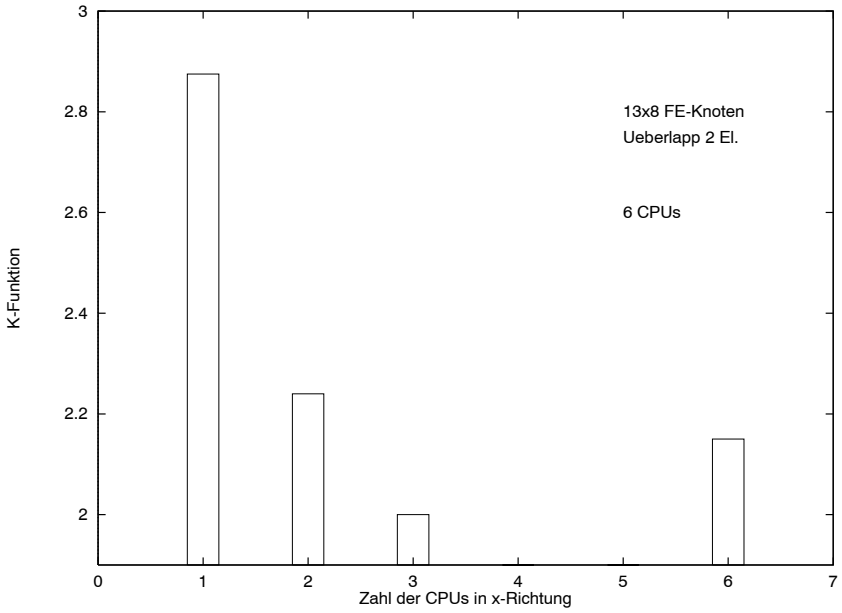


Abbildung 4.8: 2D-Beispiel: K-Funktion

Aus der Abbildung kann entnommen werden, daß es bei diesem speziellen Beispiel nicht möglich ist, eine Gebietszerlegung durchzuführen, bei der sich die Vergrößerung des Rechenaufwands, hervorgerufen durch die überlappenden Zerlegung, in Bezug auf die Zahl der zu berechnenden FE-Knoten nicht mindestens verdoppelt. Die Zerlegung mit 3 Prozessoren in x-Richtung und dementsprechend 2 Prozessoren in y-Richtung liefert die in Hinsicht auf den Berechnungsaufwand optimale Lösung.

3D-Betrachtung

Im folgenden soll ein Weg zur Bestimmung der Parameter G, U_{eff} und I für eine dreidimensionale Gebietszerlegung aufgezeigt werden. Hierzu wird ein Gebiet entsprechend Abbildung 4.9 betrachtet.

Das Gebiet soll durch n_x Knoten in der x-Richtung, n_y Knoten in der y-Richtung und n_z Knoten in der z-Richtung definiert sein. Unter der Annahme, daß 12 Prozessoren ($C = 12$) zur Verfügung stehen, wird das Gesamtgebiet in 12 Teilgebiete mit einem Überlapp von $L = 2$ Elementen zerlegt. Hierzu wird das Gebiet in x-Richtung zweimal und in y-, bzw. z-Richtung jeweils einmal geschnitten. In diesem Beispiel ergeben sich damit für die ganzzahligen Faktoren der Prozessorzahl C die Werte

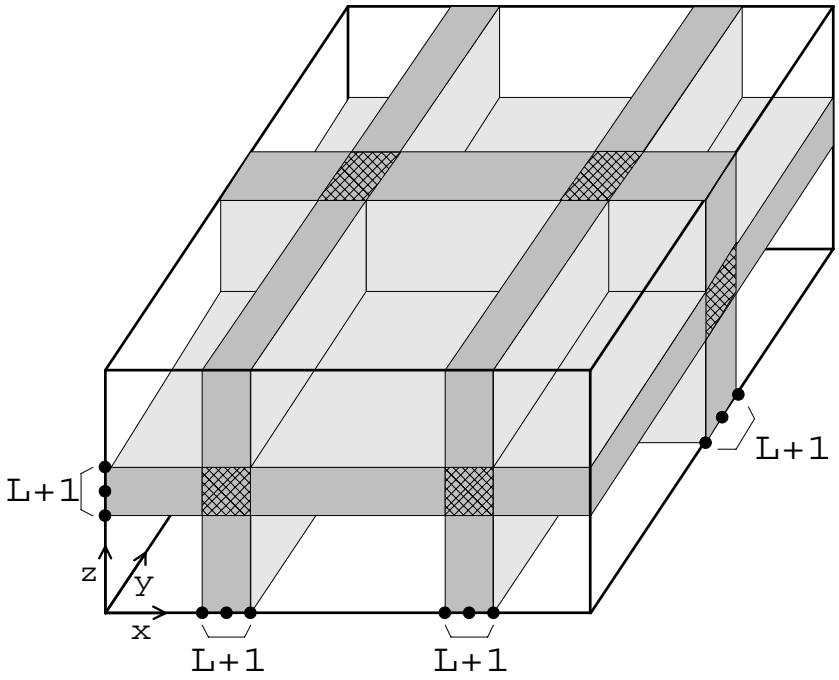


Abbildung 4.9: 3D-Gitterbetrachtung

$c_x = 3$, $c_y = 2$ und $c_z = 2$. In der Abbildung sind die gemeinsamen Überlappbereiche der Teilgebiete als Scheiben der Dicke $(L + 1)$ FE-Knoten zu erkennen.

Die Gesamtzahl G der FE-Knoten kann sofort mit den Parametern n_x, n_y und n_z des Gesamtgitters angegeben werden.

$$G = n_x \cdot n_y \cdot n_z \tag{4.21}$$

Um die Zahl I der inneren Knoten berechnen zu können, werden die Überlappscheiben, entsprechend der zweidimensionalen Betrachtung, zu den Rändern des Gebietes hin verschoben. Es entsteht ein quaderförmiger, zusammenhängender Bereich von Knoten. Die Zahl dieser Knoten kann mit Hilfe von Gleichung 4.22 angegeben werden, wobei die Zahlen I_x, I_y und I_z die Knoten angeben, die auf den Kanten des Quaders liegen.

$$I = I_x \cdot I_y \cdot I_z \tag{4.22}$$

Die Berechnung der Parameter I_i mit $i = x, y, z$ erfolgt entsprechend der zweidimensionalen Gitterbetrachtung.

$$I_i = n_i - (c_i - 1) \cdot (L + 1) \qquad i = x, y, z \tag{4.23}$$

Durch Einfügen der Parameter I_i in Gleichung 4.22 kann die Zahl I der inneren Knoten angegeben werden.

$$\begin{aligned} I &= (n_x - (c_x - 1) \cdot (L + 1)) \\ &\quad \cdot (n_y - (c_y - 1) \cdot (L + 1)) \\ &\quad \cdot (n_z - (c_z - 1) \cdot (L + 1)) \end{aligned} \tag{4.24}$$

Das Aufstellen der Formel für die effektive Überlappknotenzahl wird ebenfalls entsprechend der zweidimensionalen Betrachtung durchgeführt. Zunächst werden die Überlappbereiche wieder getrennt nach Raumrichtungen betrachtet. In der x-Richtung ergeben sich $c_x - 1$ Scheiben der Dicke $L + 1$ mit einer Grundfläche von $n_y \cdot n_z$ FE-Knoten. Jeder Knoten, der in einer solchen Überlappscheibe liegt, gehört wieder zu genau zwei Teilgebieten. Die effektive Überlappknotenzahl für Schnitte in x-Richtung kann damit angegeben werden: $2 \cdot (L + 1) \cdot (c_x - 1) \cdot n_y \cdot n_z$. Für die y-Richtung ergibt sich $2 \cdot (L + 1) \cdot (c_y - 1) \cdot n_x \cdot n_z$ und für die z-Richtung folgt $2 \cdot (L + 1) \cdot (c_z - 1) \cdot n_x \cdot n_y$.

Aus der gleichzeitigen Betrachtung aller drei Raumrichtungen folgt, daß die Knoten, die in der Schnittmenge jeweils zweier Überlappscheiben liegen, zu genau vier Teilgebieten gehören. Knoten, die in der Schnittmenge von drei Überlappbereichen liegen, gehören zu acht ineinandergreifenden Teilgebieten. Knoten, die nicht innerhalb der Schnittmengen von zwei oder mehr Überlappscheiben liegen, aber zu den Überlappbereichen gehören, treten in genau zwei Teilgebieten auf. Um die Größe der effektiven Überlappknotenzahl angeben zu können, werden wieder die Teilergebnisse der nach Richtungen getrennten Betrachtung summiert. Für den Fall, daß zwei Bereiche eine gemeinsame Schnittmenge besitzen, ist die effektive Knotenzahl analog zum zweidimensionalen Fall direkt durch die Addition gegeben. Falls drei Bereiche eine gemeinsame Schnittmenge besitzen, werden die Knoten bei der Addition nur sechsfach gewichtet, so daß für diesen Fall ein zusätzlicher Term auftritt. Die Schnittmenge von drei Überlappbereichen besteht aus genau $(L + 1)^3$ FE-Knoten. Insgesamt treten diese Bereiche $(c_x - 1) \cdot (c_y - 1) \cdot (c_z - 1)$ mal auf. Damit die Knoten in diesen Bereichen achtfach gewichtet werden, muß der folgende Term zu der Zahl der effektiven Überlappknoten addiert werden.

$$2 \cdot (L + 1)^3 \cdot (c_x - 1) \cdot (c_y - 1) \cdot (c_z - 1)$$

Damit ergibt sich für U_{eff} der Zusammenhang nach Gleichung 4.25.

$$\begin{aligned} U_{eff} &= 2 \cdot (L + 1) \cdot \Big\{ (c_x - 1) \cdot n_y \cdot n_z \\ &\quad + (c_y - 1) \cdot n_x \cdot n_z \\ &\quad + (c_z - 1) \cdot n_x \cdot n_y \\ &\quad + (c_x - 1) \cdot (c_y - 1) \cdot (c_z - 1) \cdot (L + 1)^2 \Big\} \end{aligned} \tag{4.25}$$

Auf die Herleitung der K-Funktion für den dreidimensionalen Fall entsprechend Gleichung 4.7 soll an dieser Stelle verzichtet werden, da das Einsetzen der Formelausdrücke für U_{eff}, G und I zu einer sehr unübersichtlichen Form führt, aus der

keine weiteren Aussagen abgeleitet werden können. Da die Prozessorzahl C konstant vorgegeben wird, kann aus der Faktorisierung entsprechend Gleichung 4.3 eine Variable eliminiert werden. Die Beschreibung der Gebietszerlegung kann dann durch die beiden unabhängigen Größen c_x und c_y angegeben werden.

$$c_z = \frac{C}{c_x \cdot c_y} \tag{4.26}$$

Wenn die zusätzliche Annahme aus Gleichung 4.26 in die K-Funktion eingesetzt wird, besteht diese Funktion nur noch aus den bekannten Größen n_x, n_y, n_z, L, C und den zu optimierenden Parametern c_x und c_y.

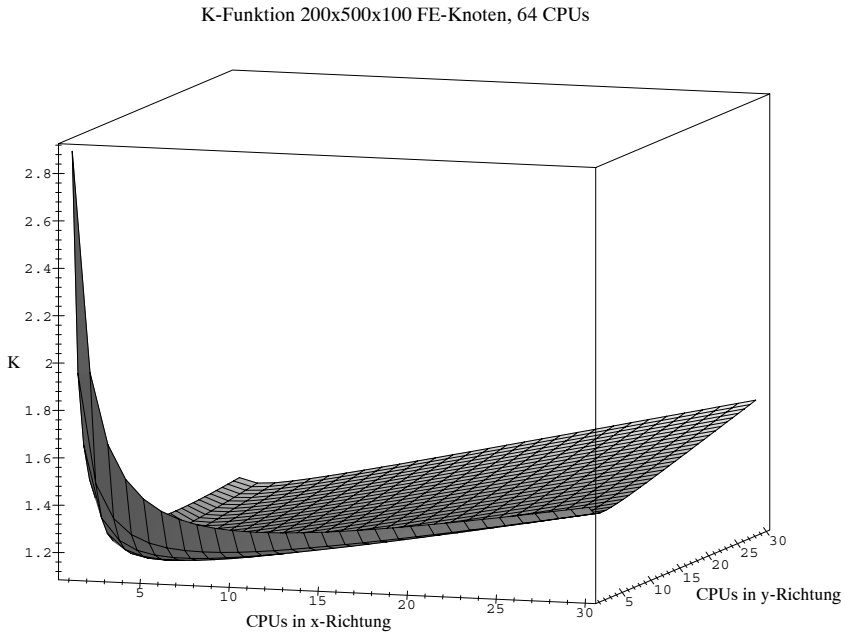


Abbildung 4.10: 3D-Beispiel: K-Funktion

In Abbildung 4.10 ist ein Beispiel für eine dreidimensionale Gebietszerlegung dargestellt. Es soll ein Gebiet, bestehend aus $n_x = 200$, $n_y = 500$ und $n_z = 100$ FE-Knoten, auf $C = 64$ Prozessoren aufgeteilt werden. Für diese Zerlegung soll ein Überlapp von $L = 2$ Elementen vorgegeben werden. Auf der x- bzw. y-Achse ist jeweils die Prozessorzahl aufgetragen, die in der jeweiligen Richtung eingesetzt werden soll. Die z-Achse gibt den Wert der K-Funktion für die entsprechende Prozessorkombination an. Da die Werte für c_x, c_y und c_z natürliche Zahlen sein müssen und die Multiplikation der drei Größen die Gesamtprozessorzahl C ergeben muß, sind nicht alle Kombinationen, die in der Abbildung dargestellt sind, wählbar. Die optimale Aufteilung wird erreicht, wenn in x-Richtung vier, in y-Richtung acht und in z-Richtung zwei Prozessoren eingesetzt werden. Die K-Funktion ergibt den Wert

1, 12 für diese Kombination, was einem sehr geringen Mehraufwand für die parallele Berechnung entspricht.

Eine Erweiterung der in diesem Kapitel vorgestellten Partitionierungsstrategie für n-dimensionale Gitterstrukturen erscheint möglich, soll aber im Rahmen dieser Arbeit nicht weiter betrachtet werden.

4.4 Beispiele für Gitterzerlegungen

Für das Programm TRACE wird die Zahl der Prozessoren, die jeweils eingesetzt werden sollen, beim Programmstart vorgegeben. Für diese Zahl soll dann eine in Hinsicht auf den Berechnungsaufwand optimale Aufteilung des Gesamtgebietes durchgeführt werden. Hierzu werden zunächst alle Kombinationen für die Faktorisierung der Prozessorzahl berechnet, und danach wird für jede Kombination die K-Funktion aufgestellt. Die optimale Kombination ist diejenige, bei der die K-Funktion ihren kleinsten Wert annimmt. In diesem Kapitel werden einige Beispiele für Gitterzerlegungen mit Schnitten in einer, zwei und drei Raumrichtungen vorgestellt. Es sollen dabei jeweils die Werte der K-Funktion für unterschiedliche Prozessorzahlen und die entsprechende optimale Gebietszerlegung betrachtet werden.

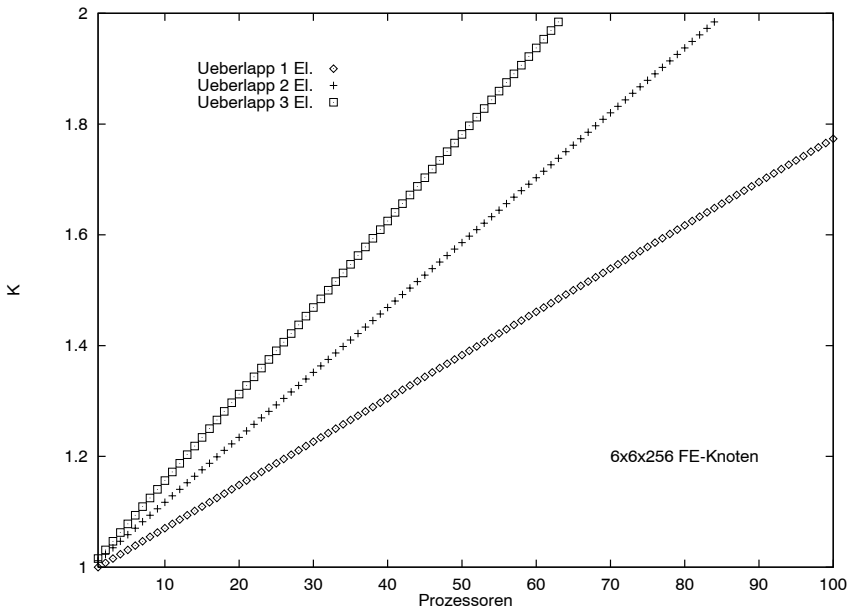


Abbildung 4.11: Beispiel 1: K-Funktionen mit Schnitten nur in z-Richtung

Im *ersten Beispiel* wird ein kleines Gebiet mit 9216 FE-Knoten betrachtet, bei dem die Gitterparameter die Größen $n_x = 6, n_y = 6$ und $n_z = 256$ FE-Knoten besitzen. Dieses Gebiet soll für einen Überlapp von ein, zwei und drei Elementen auf bis zu 100 Prozessoren aufgeteilt werden. In Abbildung 4.11 sind die Werte der K-Funktion für die verschiedenen Zerlegungen über der Zahl der Prozessoren aufgetragen. Die jeweils optimale Zerlegung wird erreicht, indem das Gebiet nur in z-Richtung geschnitten wird. Hieraus ergibt sich für die Prozessorzahlen, daß

$c_x = 1, c_y = 1$ und $c_z = C$ gelten muß. Diese Zerlegung ergibt sich sofort aus der Betrachtung des Gebietes, da die Knotenzahlen in x- und in y-Richtung sehr viel kleiner sind als die Knotenzahl in z-Richtung. Bei sehr großen Prozessorzahlen kann es bei diesem kleinen Gebiet vorkommen, daß die Zahl der inneren Knoten der einzelnen Teilgebiete, also die Zahl der Knoten, die nicht in den Überlappbereichen liegen, Null wird. Im Extremfall könnten sich die Überlappgebiete, die jeweils von oben und von unten in ein Teilgebiet hineinragen, in der Mitte des Teilgebietes überlappen. Dieser Fall ist für eine gültige Zerlegung entsprechend dem Schwarz'schen Verfahren (siehe Abschnitt 2.2) nicht zulässig und muß deshalb vom Programm TRACE bei der Berechnung der Zerlegung erkannt und ausgeschlossen werden. Gleichung 4.27 gibt die Grenze für eine sinnvolle Gebietszerlegung an.

$$\frac{n_i + (c_i - 1) \cdot (L + 1)}{c_i} > 2 \cdot (L + 1) \qquad i = x, y, z \tag{4.27}$$

Die Zahl der FE-Knoten, die in einer Raumrichtung zu einem Teilgebiet gehören, sollte mindestens so groß sein wie die Zahl der Knoten, die im Überlappbereich liegen. Für den Fall, daß zwei Überlappbereiche in der betrachteten Raumrichtung in das Gebiet hineinragen, gewährleistet diese Abschätzung, daß sich die Überlappbereiche nicht überschneiden und daß immer mindestens ein FE-Knoten ein innerer Knoten des Gebietes ist. Bei der Zerlegung entsprechend Abbildung 4.11 ist im weiteren zu erkennen, daß bei einer wachsenden Prozessorzahl ein gleichmäßig steigender Berechnungsaufwand, hervorgerufen durch die Gebietszerlegung, auftritt. Da das Gebiet nur in z-Richtung geschnitten wird und dementsprechend die Faktorisierung der Prozessorzahl immer nach der Regel $1 : 1 : C$ durchgeführt wird, können alle Prozessorzahlen für die Programmausführung gewählt werden. Die Zahl der FE-Knoten in z-Richtung, die jeweils zu den Teilgebieten gehören, kann durch Verwendung von Gleichung 4.5 angegeben werden.

Als *zweites Beispiel* soll ein Gebiet mit 480000 FE-Knoten betrachtet werden. In Abbildung 4.12 ist die K-Funktion für unterschiedliche Prozessorzahlen für dieses Gebiet aufgetragen. Das Gebiet setzt sich aus $n_x = 200, n_y = 30$ und $n_z = 80$ FE-Knoten zusammen. Die Zerlegung soll jeweils für einen Überlapp von 2 Elementen durchgeführt werden. Bei diesem Gebiet kann der Einfluß der Vorgabe einer bestimmten Prozessorzahl erkannt werden. Falls C sich aus dem Produkt von drei Primzahlen zusammensetzt, kann die optimale Kombination nur durch Permutation der Faktoren bestimmt werden. Bei dem in der Abbildung dargestellten Beispiel liegen die Werte der K-Funktion für diese Primzahlenkombinationen jeweils auf einer Geraden. Diese Geraden sind mit $C : 1 : 1, C/2 : 1 : 2$ usw. beschriftet. Für einige Prozessorzahlen konnten keine sinnvollen Gebietszerlegungen gefunden werden, da das Kriterium 4.27 nicht erfüllbar war. Diese Prozessorzahlen (z.B. 67, 103, 197) sind nicht in der Abbildung dargestellt. Eine untere Grenze für den Rechenaufwand, der bei optimaler Gebietszerlegung mindestens durchgeführt werden muß, wird durch eine konvexe, monoton steigende Funktion beschrieben.

In Tabelle 4.1 sind einige optimale Zerlegungen des Gebietes für vorgegebene Prozessorzahlen angegeben. Falls 27 Prozessoren eingesetzt werden sollen, ergibt

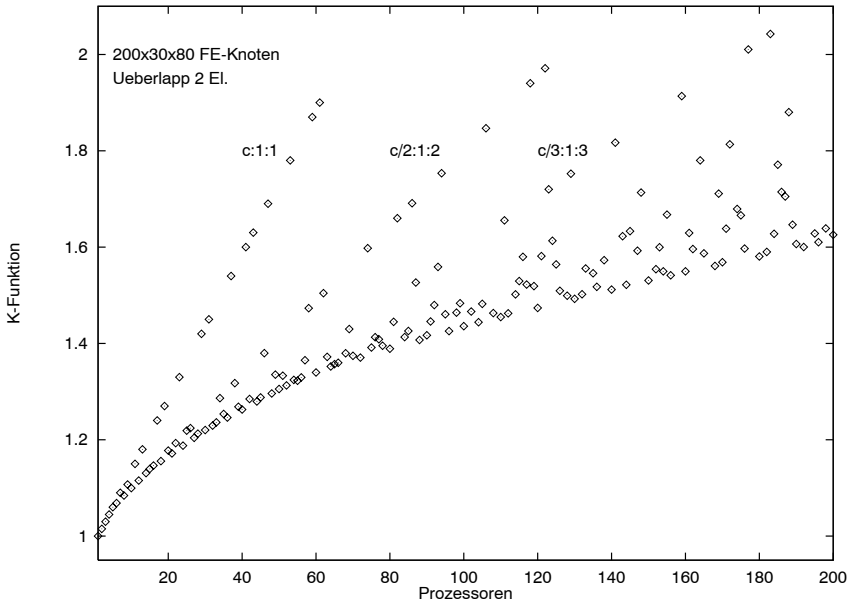


Abbildung 4.12: Beispiel 2: K-Funktion für unterschiedliche Gebietszerlegungen

CPUs	27	60	61	152
K	1,20	1,34	1,90	1,55
opt. Auft.	9/1/3	12/1/5	61/1/1	19/2/4

Tabelle 4.1: Gebietszerlegungen zum Beispiel 2

sich die optimale Gebietszerlegung für die Kombination 9:1:3. Bei dieser Kombination beträgt der Berechnungsaufwand nur 20 Prozent mehr als er bei sequentieller Lösung betragen würde. Wenn 61 Prozessoren eingesetzt werden sollen, ergibt sich für die optimale Zerlegung die Kombination 61:1:1, da die vorgegebene Zahl eine Primzahl ist. Die K-Funktion besitzt den Wert 1,9, was einem hohen Mehraufwand, hervorgerufen durch die überlappende Zerlegung entspricht. In diesem Fall wäre der Einsatz von 60 Prozessoren effizienter, da der zusätzliche Aufwand für die Berechnungen nur 34 Prozent betragen würde.

Aus den beiden in diesem Kapitel vorgestellten Beispielen läßt sich erkennen, daß eine allgemeine Angabe einer sinnvollen Zahl von Prozessoren für die Programmausführung nicht möglich ist. Falls die Werte der Gitterparameter n_x, n_y und n_z nahe beieinander liegen, sollte die Prozessorzahl möglichst nicht aus drei Primzahlen zusammengesetzt sein, während bei weit auseinander liegenden Gitterparametern auch Primzahlen für C gewählt werden können.

Kapitel 5

Der Austausch von FE-Knoten

Mit der additiven Variante des Schwarz'schen Verfahrens (Abschnitt 2.2) ist eine gleichzeitige parallele Berechnung von sich überlappenden Teilgebieten eines Finiten-Elemente-Gitters möglich. Durch die Zerlegung des Gesamtgitters entstehen neue Ränder, die mit geeignet gewählten Dirichlet-Randwerten belegt werden müssen. Während der Berechnungen müssen diese Randwerte durch entsprechende Werte aus den angrenzenden Teilgebieten aktualisiert werden. In diesem Kapitel sollen zunächst die Nachbarschaftsbeziehungen der Teilgebiete betrachtet werden. Daran schließt sich eine Definition an, die die auszutauschenden Bereiche beschreibt. Zur Verdeutlichung wird danach die Struktur dieser Bereiche für bestimmte Nachbarschaftsbeziehungen aufgezeigt.

5.1 Nachbarschaftsbeziehungen

Wie in Kapitel 4 beschrieben, sollen die bei der Zerlegung des Gesamtgitters entstehenden Teilgitter eine hexaedrische Gestalt besitzen, die durch die Zahl der FE-Knoten in der x-, y- und z-Richtung angegeben werden kann. Die Schnitte sind immer senkrecht zu einer Koordinatenachse durchzuführen und sollen durch das gesamte Gebiet gelegt werden, so daß kein Versatz der Teilgebiete beiderseits der Schnittfläche auftreten kann. Da die Schnitte in allen drei Raumrichtungen gezogen werden können und die bei einem Schnitt entstehenden Teilgebiete jeweils um die Zahl der Überlappknoten ineinander geschoben werden, erhöht sich die Komplexität der Geometrie.

In diesem Abschnitt soll eine dreidimensionale Zerlegung des Gesamtgebietes betrachtet werden. In Abbildung 5.1 ist diese Zerlegung dargestellt. In jeder Raumrichtung wurden jeweils zwei Schnitte gezogen, so daß sich insgesamt 27 Teilgebiete ergeben. Jeder Würfel in der Abbildung soll ein Teilgebiet charakterisieren, wobei vereinfachend angenommen wurde, daß alle Teilgebiete dieselbe Gestalt besitzen. Bei einer dreidimensionalen Zerlegung kann ein Teilgebiet maximal 26 Nachbarn besitzen. Jedes Teilgebiet entspricht jeweils einem dreidimensionalen Gitter aus FE-Knoten, das jedoch nicht in der Abbildung dargestellt ist. Da die Abbildung nur die Nachbarschaftsbeziehungen aufzeigen soll, wurden die überlappenden Teilgebiete auseinandergezogen. Alle Teilgebiete sind ursprünglich jeweils in das Gebiet

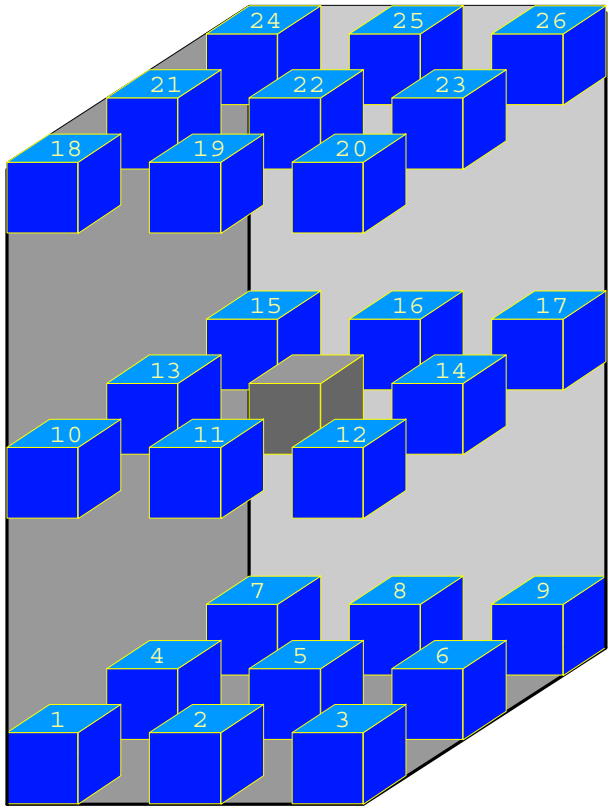


Abbildung 5.1: Interne Numerierung der direkten Nachbarprozessoren

des heller dargestellten Würfels um die Zahl der Überlappknoten hineinverschoben. Hieraus ergibt sich, daß einzelne FE-Knoten des Gesamtgebietes in dieser Abbildung mehrfach vorhanden sind.

Im folgenden soll jeweils der heller dargestellte Würfel in der Mitte der Abbildung betrachtet werden, da dieser Würfel die maximal mögliche Zahl von 26 Nachbarwürfeln besitzt. Generell lassen sich drei Arten von Verschiebungen erkennen. Unter einer *1D-Verschiebung* soll eine Verschiebung in nur einer Raumrichtung verstanden werden. Ein Beispiel hierfür ist das Teilgebiet mit der Nummer 5, welches um genau die Zahl der Überlappknoten in vertikaler Richtung in das Gebiet des helleren Würfels hineinverschoben werden muß. Auch für die Teilgebiete 11, 13, 14, 16 und 22 muß jeweils eine 1D-Verschiebung durchgeführt werden, so daß die 1D-Verschiebung insgesamt sechsmal auftreten kann. Bei einer *2D-Verschiebung* wird ein Teilgebiet zunächst um die Zahl der Überlappknoten in einer Raumrichtung und danach um die Zahl der Überlappknoten in einer hierzu senkrechten zweiten Raumrichtung in den helleren Würfel verschoben. Als Beispiel soll der Würfel mit der Nummer 6 betrachtet werden. Zunächst wird dieser Würfel vertikal nach oben und danach horizontal nach links in den helleren Würfel verschoben. 2D-Verschiebungen werden für insgesamt 12 Teilgebiete (2,4,6,8,10,12,15,17,19,21,23,25) durchgeführt. Bei einer *3D-Verschiebung* wird ein Teilgebiet entsprechend in drei Raumrichtungen jeweils um die Zahl der Überlappknoten verschoben. Ein Beispiel hierfür ist der Würfel mit der Nummer 1. Dieser Würfel wird zunächst horizontal nach rechts,

dann vertikal nach oben und anschließend senkrecht zur Papierebene nach hinten um jeweils die Zahl der Überlappknoten in den helleren Würfel hineinverschoben. 3D-Verschiebungen treten insgesamt achtmal auf (Würfel 1,3,7,9,18,20,24,26).

Es ergibt sich, daß nach den Verschiebungen der 26 Nachbarn des helleren Würfels einige der Nachbarn ebenfalls gemeinsame FE-Knoten besitzen. So wird beispielsweise zwischen den Würfeln mit den Nummern 1 und 2 eine 1D-Verschiebung stattgefunden haben. Dieser Fall ist analog zu der Verschiebung von Würfel 13, so daß die Betrachtung des helleren Würfels zur Darstellung aller Nachbarschaftsbeziehungen ausreicht.

Da für das Programm TRACE die Annahme gelten soll, daß jeder Prozessor nur genau ein Teilgebiet zur Berechnung zugewiesen bekommt, können die Würfel in Abbildung 5.1 auch zur Symbolisierung der Prozessoren verwendet werden. In der Abbildung sind die Nachbarprozessoren jeweils mit einer Nummer versehen, die der Numerierung der FE-Knoten in Abschnitt 4.1 folgt. Jeder der 26 Prozessoren muß während der Berechnungen unter anderem einen Teil der Oberfläche des helleren Prozessors mit neuen Randwerten belegen. Im nächsten Abschnitt soll erläutert werden, welche FE-Knoten ausgetauscht werden müssen.

5.2 Pseudo-Randknoten

Unter einem Pseudo-Randknoten soll allgemein ein FE-Knoten verstanden werden, der während der Berechnungen von Nachbarn aktualisiert werden muß. Da jeder Prozessor ein Randwertproblem zu lösen hat, werden nur die Knoten berechnet, die nicht auf der Berandung des zu berechnenden Gebietes liegen. Da für das Schwarz'sche Verfahren eine Gebietszerlegung durchgeführt wird, werden neue Gebietsberandungen erzeugt, die nicht mit Randwerten vorbelegt sind. Diese Werte werden vor dem Beginn der Berechnungen abgeschätzt, so daß wieder für jedes Teilgebiet ein Randwertproblem angegeben werden kann. Da diese geschätzten Randwerte im Regelfall nicht der exakten Lösung entsprechen, aber auch nicht von dem Prozessor berechnet werden können, auf dessen Gebietsberandung diese Knoten liegen, werden diese Werte von anderen Prozessoren während der Berechnungen aktualisiert. Voraussetzung für einen Austausch ist jedoch, daß der entsprechende Knoten im Inneren des Gebietes des aktualisierenden Prozessors liegt.

Nach diesen Überlegungen läßt sich eine Definition für Pseudo-Randknoten angeben:

> *Pseudo-Randknoten sind die Knoten, die in der Schnittmenge der Knoten der Gebietsberandung des empfangenden und der inneren Knoten des sendenden Gebietes liegen.*

Nach dieser Definition können die Knoten, die ausgetauscht werden müssen, für die drei Arten von Verschiebungen angegeben werden. Bei einer 1D-Verschiebung liegen die Pseudo-Randknoten alle in einer Ebene, bei einer 2D-Verschiebung entsprechend in zwei Ebenen und bei einer 3D-Verschiebung in drei Ebenen. Falls die Teilgebiete keine echten Randknoten, also Knoten, die bei dem Gesamtgebiet schon auf einem

echten Rand gelegen haben, besitzen, kann der Austausch gemäß der Definition durchgeführt werden.

Ein Problem ergibt sich jedoch, wenn ein Teilgebiet echte Randknoten besitzt. Als Beispiel hierfür soll der Prozessor 14 in Abbildung 5.1 betrachtet werden. Das Gebiet dieses Prozessors besitzt auf der rechten Seite keine weiteren Nachbarn, so daß alle Knoten, die auf der rechten Seite des Würfels liegen, echte Randknoten sind. Das Gebiet des Prozessors 12 wird mit einer 1D-Verschiebung in das Gebiet des Prozessors 14 hineingeschoben. Entsprechend der Definition muß Prozessor 12 während der Berechnungen alle Knoten von 14 aktualisieren, die auf der vorderen Seite des Würfels liegen. Ausgenommen von der Aktualisierung sind jedoch die Knoten, die auf den vier vorderen Kanten des Würfels 14 liegen, da sie sich nicht im Inneren von Würfel 12 befinden. Falls auf dem echten Rand Dirichlet-Randwerte vorgeben wurden, sind diese Werte für die in beiden Teilgebieten doppelt auftretenden Knoten gleich groß und müssen nicht ausgetauscht werden. Da bei Dirichlet-Randwerten Zahlenwerte für die entsprechenden Knoten vorgegeben werden, sind diese Randwerte nicht von den Berechnungen abhängig. Falls jedoch Flußrandbedingungen (Neumann'sche oder Cauchy'sche Randbedingungen) vorgegeben wurden, ändern sich diese Werte in Abhängigkeit der Berechnungen, da jeweils Gradienten der Lösung gebildet werden müssen. Da die Randwerte der einzelnen Teilgebiete nur durch Austausch mit den benachbarten Teilgebieten aktualisiert werden können, werden nach der angegebenen Definition für die echten Randwerte nie neue Werte zur Berechnung der Flußrandbedingungen erlangt. Im speziellen für eindimensionale Simulationsrechnungen muß deshalb die oben angegebene Definition erweitert werden.

Erweiterung der Definition für Pseudo-Randknoten:

> *Wenn eine oder mehrere Ebenen des empfangenden Gebietes, auf denen Pseudo-Randknoten liegen, durch echte Randknoten begrenzt werden, müssen diese Randknoten, obwohl sie auf dem Rand des sendenden Gebietes liegen, ebenfalls ausgetauscht werden.*

Mit Hilfe der in diesem Abschnitt angegebenen Definitionen können alle Knoten bestimmt werden, die während der Berechnungen zwischen benachbarten Gebieten ausgetauscht werden müssen. Da maximal acht Teilgebiete einen gemeinsamen Überlappungsbereich besitzen können, ist es möglich, daß bestimmte Pseudo-Randknoten des empfangenden Gebietes von mehreren sendenden Gebieten gleichzeitig aktualisiert werden müssen. Da verschiedene Berechnungsergebnisse in den einzelnen Teilgebieten auftreten können, obwohl es sich um denselben FE-Knoten in Bezug auf das Gesamtgitter handelt, werden die empfangenen Werte gemittelt. Dabei können Pseudo-Randknoten, die auf einem echten Rand des Gesamtgebietes liegen, beim empfangenden Gebiet von maximal zwei Nachbargebieten aktualisiert werden. Pseudo-Randknoten, die keine echten Randknoten sind, können beim empfangenden Gebiet von maximal vier Nachbargebieten aktualisiert werden. Im weiteren sollte beachtet werden, daß die Zahl der Pseudo-Randknoten und die Häufigkeit der Mehrfachaktualisierungen von der Größe des Überlapps abhängen.

5.3 Struktur der auszutauschenden Bereiche

In diesem Abschnitt sollen exemplarisch die Strukturen der auszutauschenden Bereiche für die drei Arten der Verschiebung dargestellt werden. Es wird wieder Abbildung 5.1 betrachtet, wobei der hellere Würfel in der Mitte jeweils die schraffierten Teilbereiche seiner Oberfläche von dem entsprechenden Nachbarn empfangen soll.

5.3.1 1D-Verschiebung

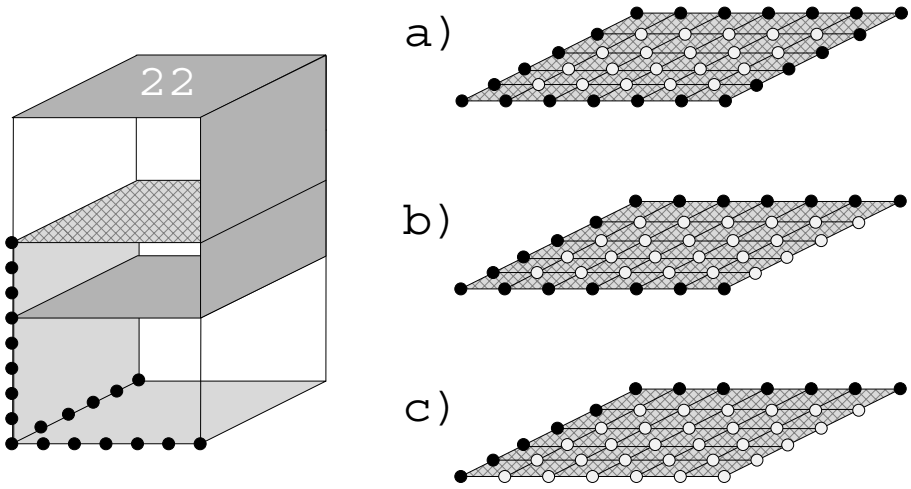


Abbildung 5.2: 1D-Verschiebung, auszutauschende Pseudo-Randknoten

In Abbildung 5.2 ist das Teilgebiet des Prozessors 22 durch eine 1D-Verschiebung senkrecht nach unten um die Zahl der Überlappknoten (4 Knoten bzw. 3 Elemente) in das Gebiet des helleren Prozessors verschoben worden. Falls die beiden betrachteten Teilgebiete keine gemeinsamen echten Randknoten besitzen, wird entsprechend Teilbild a) eine Ebene mit den hell gezeichneten Pseudo-Randknoten ausgetauscht. Im Teilbild b) sind die Pseudo-Randknoten für den Fall dargestellt, daß auf der rechten Seite keine weiteren Nachbarn vorhanden sind. Für das Teilbild c) wurde davon ausgegangen, daß sowohl auf der rechten als auch auf der vorderen Seite keine Nachbarn existieren. Man kann erkennen, daß bei einer 1D-Verschiebung die Größe des Überlapps keinen Einfluß auf die Zahl der Pseudo-Randknoten hat.

5.3.2 2D-Verschiebung

Abbildung 5.3 zeigt eine 2D-Verschiebung. Das Gebiet des Prozessors 21 wird zunächst um die Zahl der Überlappknoten (in der Abbildung sind dies 3 Knoten bzw. 2 Elemente) vertikal nach unten und danach um die Zahl der Überlappknoten horizontal nach rechts in das Gebiet des helleren Prozessors verschoben. Die Pseudo-Randknoten liegen auf zwei Ebenen, wobei im Teilbild a) wieder davon ausgegangen wurde, daß keine echten Randknoten vorhanden sind. Teilbild b) stellt

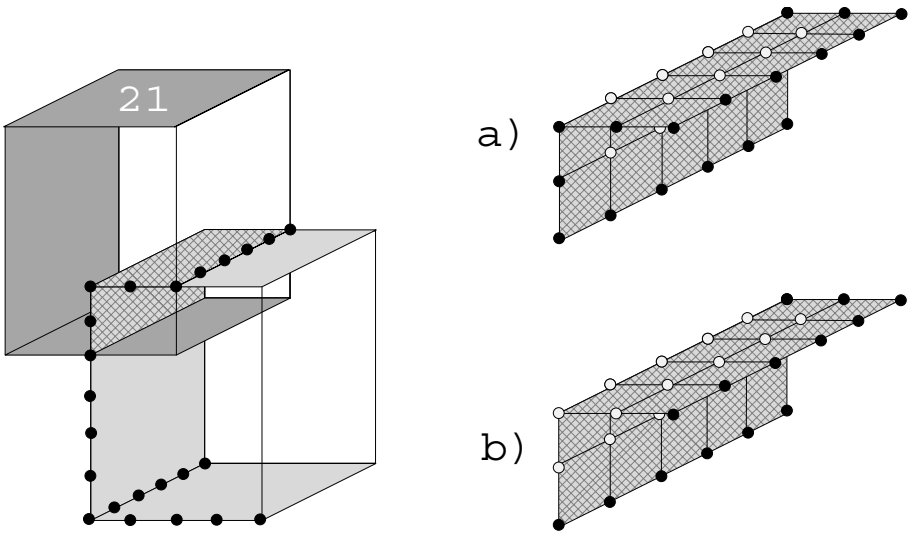


Abbildung 5.3: 2D-Verschiebung, auszutauschende Pseudo-Randknoten

die auszutauschenden Knoten für den Fall dar, daß auf der vorderen Seite keine Nachbarn vorhanden sind.

Im Fall einer 2D-Verschiebung ist die Zahl der Pseudo-Randknoten von der Größe des Überlapps abhängig. Bei einem Überlapp von nur einem Element wird von den zwei Ebenen, auf denen die Pseudo-Randknoten in Abbildung 5.3 liegen, nur die beiden Ebenen gemeinsame Kante ausgetauscht. Mit steigendem Überlapp nimmt auch die Zahl der Pseudo-Randknoten zu, wie aus der Abbildung ersichtlich ist.

5.3.3 3D-Verschiebung

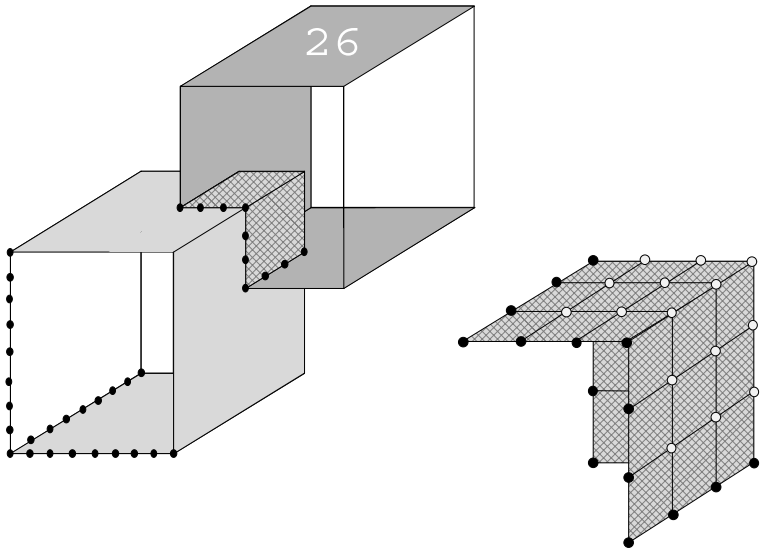


Abbildung 5.4: 3D-Verschiebung, auszutauschende Pseudo-Randknoten

Als Beispiel für eine 3D-Verschiebung (Abbildung 5.4) soll das Gebiet des Prozessors 26 betrachtet werden. Dieses Gebiet wird bei einem Überlapp von 4 Knoten bzw. 3 Elementen zunächst nach unten, dann nach links und anschließend nach vorn in das Gebiet des helleren Prozessors verschoben. Die Pseudo-Randknoten liegen auf insgesamt 3 Ebenen. Da bei der Gebietszerlegung davon ausgegangen wurde, daß die Schnitte immer durch das gesamte Gebiet gelegt werden, kann kein Versatz zwischen den Teilgebieten auftreten. Hieraus ergibt sich, daß bei einer 3D-Verschiebung niemals echte Randknoten auf den auszutauschenden Ebenen auftreten können. Ausgetauscht werden wieder die hell dargestellten Knoten.

Die Zahl der Pseudo-Randknoten bei einer 3D-Verschiebung ist wie bei der 2D-Verschiebung eine Funktion des Überlapps. Bei einem Überlapp von einem Element wird nur der Knoten ausgetauscht, der auf allen drei Ebenen liegt (Eckknoten hinten rechts). Mit wachsendem Überlapp nimmt die Zahl der auszutauschenden Knoten ebenfalls zu.

Kapitel 6

Die Parallelisierung von TRACE

In diesem Kapitel sollen die Änderungen der sequentiellen Programmversion von TRACE für den Einsatz auf einem massiv-parallelen Rechnersystem vorgestellt werden. Um die Portabilität des Programms zu garantieren, sollen alle Änderungen innerhalb des Sprachumfangs von FORTRAN 77 durchgeführt werden. Das Ausnutzen von Befehlen, die spezielle FORTRAN-Compiler zur Verfügung stellen, ist nicht vorgesehen. Zunächst werden allgemeine Programmänderungen betrachtet, die zu einer Vereinfachung und besseren Übersichtlichkeit des Programm-Codes führen sollen. Daran anschließend werden die Implementierung des Schwarz'schen Verfahrens in den Programm-Code und die nötigen Änderungen vorgestellt. Die Notwendigkeit einer Anpassung der Struktur der *WATDAT*-Datei soll ebenso betrachtet werden.

6.1 Vorbereitende Programmänderungen

Da das Programm TRACE über einen längeren Zeitraum von verschiedenen Personen entwickelt und erweitert wurde, war eine Überarbeitung des Programm-Codes notwendig. So wurden unter anderem eine Neuordnung der Datenstrukturen, das Einfügen einer INCLUDE-Datei und die Einrichtung einer zusätzlichen Eingabedatei erforderlich. Durch diese Änderungen wird eine Anpassung von TRACE zur Berechnung unterschiedlicher Simulationsgebiete vereinfacht. Den Programmablauf steuernde Parameter, wie beispielsweise Abbruchkriterien, können nun ohne erneute Übersetzung des Programms geändert werden.

6.1.1 Datenstrukturen

In den bisherigen Versionen des Programms TRACE wurde für die meisten Variablen die implizite Variablendeklaration verwendet, wie sie FORTRAN 77 vorsieht. Ein eigener Deklarationsteil für diese Variablen entfiel. Um den Code übersichtlicher zu gestalten, wurde die *IMPLICIT NONE* Compiler-Direktive in das Hauptprogramm und alle Unterprogramme aufgenommen und die benötigten Variablen explizit, also durch Angabe des Variablennamens und des zugehörigen Datentyps, definiert. Diese Direktive zählt nicht zum FORTRAN 77 Standard. Sie wird inzwischen jedoch von

den meisten Compilern unterstützt und bietet sich für eine übersichtliche Variablendeklaration bei in der Entwicklung befindlichen Programmen an.

Durch die Kopplung der beiden Programme von Yeh (siehe Abschnitt 3.3) hat sich eine Vielzahl von Variablen mit doppelter Bedeutung ergeben. Um diese voneinander unterscheiden zu können, haben alle Variablen, die den Stofftransport kennzeichnen, den Buchstaben c als Erweiterung ihres Namens erhalten. Hiermit war eine klare Unterscheidung dieser Variablen im Hauptprogramm möglich.

Alle für die Berechnungen wichtigen Variablen waren im Programm TRACE in COMMON-Blöcken gespeichert, so daß jede Routine bei Bedarf auf diese Variablen zugreifen konnte. Bei den beiden alten Programmteilen waren oft Routinen vorhanden, die gleiche Namen besaßen und auf Variablen mit gleichen Namen zugriffen. Diese Routinen hatten jedoch unterschiedliche Funktionalität, so daß beide Versionen verwendet werden mußten. Um bei der Zusammensetzung der beiden Programmteile diese voneinander unterscheiden zu können, wurde wieder der Buchstabe c als Erweiterung für den Namen der Routinen verwendet, die den Stofftransport betreffen. Da die Kopplung der Variablen zwischen dem Hauptprogramm und den Routinen über die COMMON-Blöcke durchgeführt wurde, war eine Anpassung der Variablennamen innerhalb der Routinen nicht nötig. Es ergab sich hierdurch eine inkonsistente Variablenstruktur, die die Analyse der einzelnen Programmteile zur Weiterentwicklung oder Fehlersuche erschwerte. Die Struktur der COMMON-Blöcke war zum Teil unübersichtlich gegliedert, da je nach Bedarf einzelne COMMON-Blöcke erweitert oder neu angelegt wurden.

Damit zwischen dem Hauptprogramm und den aufgerufenen Routinen eine klare Schnittstelle definiert werden konnte, über die alle Daten zwischen den einzelnen Programmteilen ausgetauscht werden müssen, wurden alle COMMON-Blöcke durch eine Erweiterung der Parameterlisten der Routinen ersetzt. Durch dieses Vorgehen läßt sich erreichen, daß das Programm TRACE von mehreren Personen weiterentwickelt werden kann, ohne das Anpassungen in nicht von der Weiterentwicklung betroffenen Routinen durchgeführt werden müssen. Außerdem ist durch die Schnittstelle zwischen dem aufgerufenen und dem aufrufenden Programm-Modul sofort ersichtlich, welche Variablen von der aufgerufenen Routine benötigt und gegebenenfalls verändert werden. Ein Nachteil der Parameterlisten ist sicherlich in der zum Teil großen Zahl der Parameter zu sehen, die übergeben werden müssen.

Vom Hauptprogramm aus gesehen werden maximal vier Routinen nacheinander aufgerufen; von den tiefer liegenden Routinen werden meist nur wenige Parameter zur Berechnung benötigt. Hieraus ergibt sich eine flache Struktur des Programms TRACE, bei der die meisten Variablen, die als Parameter übergeben werden, nur von der zuerst aufgerufenen Routine benötigt werden. Zudem wird auf viele Variablen während der gesamten Programmlaufzeit nur lesend zugegriffen. Damit die Länge der Parameterlisten deutlich verkürzt werden kann, sollten diese Variablen in einem zweiten Schritt wieder in COMMON-Blöcken zusammengefaßt werden.

6.1.2 Die INCLUDE-Datei

Die Dimensionierung der einzelnen Felder war bei der bisherigen Version des Programms TRACE nur schwer an die zu simulierende Gebietsgröße anzupassen. Häufig mußten an verschiedenen Stellen im Programm-Code Änderungen von Zahlenwerten, die die Größe der Felder angaben, vorgenommen werden.

Um den Vorgang der Anpassung des Programms für unterschiedliche Simulationsgebiete zu vereinfachen, sind alle Variablen, die zur Dimensionierung der Felder benötigt werden, in einer INCLUDE-Datei zusammengefaßt worden. Diese Dimensionierungsdatei wird in jede Routine, die mindestens einen Wert aus der Datei benötigt, mit Hilfe der *INCLUDE*-Anweisung eingebunden. Diese Anweisung gehört nicht zum FORTRAN 77 Standard. Die Vorteile bei der Verwendung dieses Befehls gleichen jedoch den Nachteil des Verlassens des Standards aus.

Im Anhang B ist eine INCLUDE-Datei angegeben, wie sie vom Programm TRACE verwendet wird.

Interessant erscheint die Möglichkeit, die Struktur des Programms TRACE durch den Einsatz weiterer INCLUDE-Dateien zu verbessern. So könnte beispielsweise eine Aufteilung der INCLUDE-Datei in zwei voneinander unabhängige Dateien dazu genutzt werden, die Trennung der beiden Programmteile zur Berechnung des Wasserflusses beziehungsweise des Stofftransportes wiederherzustellen. Die einzelnen Dateien müßten dann nur noch in die Routinen eingebunden werden, die ihrem Programmteil entsprechen. Hierbei würden bei Änderungen einer der beiden Dateien nicht mehr alle Routinen neu übersetzt werden müssen.

6.1.3 Die CONTROL-Datei

Um einen direkten Einfluß auf die numerischen Verfahren nehmen zu können, die im Programm TRACE implementiert sind, existieren mehrere Variablen, die als Steuerparameter verwendet werden. Beispiele hierfür sind die Residuen für die einzelnen Schleifen, Obergrenzen für die maximale Zahl an Schleifendurchläufen oder auch die Zahl der maximal zu berechnenden Zeitschritte. Diese Variablen konnten bisher teilweise entweder im Programm-Code, verbunden mit einer erneuten Compilierung des Programms, oder aber in der WATDAT-Datei geändert werden.

Durch die Zusammenfassung dieser Variablen in einer eigenen Datei, die zu Beginn der Programmausführung eingelesen werden muß, wurde ein Weg gefunden, der eine leichte und bequeme Änderung dieser Parameter gestattet. Die CONTROL-Datei ist eine reine ASCII-Datei, die Kommentare und Zahlenwerte für die Steuerparameter beinhaltet. Diese Datei wird beim Programmstart über einfache READ-Befehle eingelesen. Falls das Programm TRACE auf einem Mehrprozessorrechner eingesetzt wird, werden die Werte von einem Prozessor eingelesen und mit Message-Passing-Befehlen den anderen Prozessoren zugeschickt.

Ein Beispiel für eine CONTROL-Datei, wie sie vom Programm TRACE eingelesen wird, ist im Anhang C angegeben.

6.1.4 Einschränkungen der Struktur des FE-Gitters

Im Gegensatz zur sequentiellen Version des Programms TRACE sind bei der parallelen Version einige Einschränkungen bezüglich der Struktur des FE-Gitters vorgenommen worden.

Im einzelnen waren die folgenden Einschränkungen nötig. Ein FE-Element soll immer mit genau acht FE-Knoten definiert werden, und zwei benachbarte FE-Elemente besitzen immer genau vier gemeinsame FE-Knoten, die auf einer Randseite der Elemente liegen müssen. Die Zahl der FE-Elemente beziehungsweise FE-Knoten in einer Raumrichtung soll immer gleich groß sein. In verschiedenen Richtungen kann diese Zahl jedoch unterschiedlich sein. Die Numerierung der Knoten beziehungsweise Elemente muß entsprechend Abbildung 4.2 vorgenommen werden. Eine ausführliche Beschreibung der Struktur des FE-Gitters ist in Abschnitt 4.1 angegeben.

Die Einschränkungen beruhen zum Teil darauf, daß das Schwarz'sche Verfahren für nicht strukturierte Gitter nicht allgemein verwendet werden kann. Bei einer dreidimensionalen Gebietszerlegung mit Überlapp sind mehrere Fälle vorstellbar, bei denen eine eindeutige Zuordnung von Pseudoknoten, die während der Rechnung von Nachbargebieten aktualisiert werden müssen, nicht möglich ist. Zusätzlich konnte durch die Einschränkungen die Performance des Programms verbessert werden. Die Ausführung rechenintensiver Routinen, die beispielsweise der Bestimmung von Elementen auf der Oberfläche des Gesamtgebietes dienen, konnte deutlich beschleunigt werden. Auf das Einlesen einer großen Zahl von Daten aus der WATDAT-Datei zur Kopplung der Knotennummern mit den Elementnummern konnte verzichtet werden, da die vorgegebene Numerierung eine Berechnung dieser Zusammenhänge ermöglicht.

6.2 Implementierung der Parallelisierungsstrategie

Entsprechend Abschnitt 2.2 ist das der Parallelisierung des Programms TRACE zugrunde liegende Gebietszerlegungsverfahren das Schwarz'sche Verfahren. Der Einbau dieses Verfahrens in die sequentielle Programmstruktur und die notwendigen Anpassungen beziehungsweise Erweiterungen sollen im folgenden betrachtet werden.

6.2.1 Änderung der sequentiellen Programmstruktur

Die Parallelisierung von TRACE konnte so durchgeführt werden, daß ein großer Teil der sequentiellen Programmstruktur erhalten blieb. Im Hauptprogramm ist im wesentlichen nur eine einzige Schleife zu der bisherigen Struktur hinzugefügt worden. Innerhalb dieser Schleife führt eine neue Routine die Kommunikationsaufgaben während der Lösung des linearisierten Gleichungssystems aus. Um eine parallele Ausführung von TRACE erreichen zu können, waren zusätzlich auch Änderungen an mehreren Routinen erforderlich, die schon in der sequentiellen Programmversion

verwendet wurden. Abbildung 6.1 zeigt die Struktur der parallelen Programmversion von TRACE. Im folgenden sollen die notwendigen Änderungen der Programmstruktur im Vergleich zu der in Abschnitt 3.4.1 beschriebenen sequentiellen Struktur betrachtet werden.

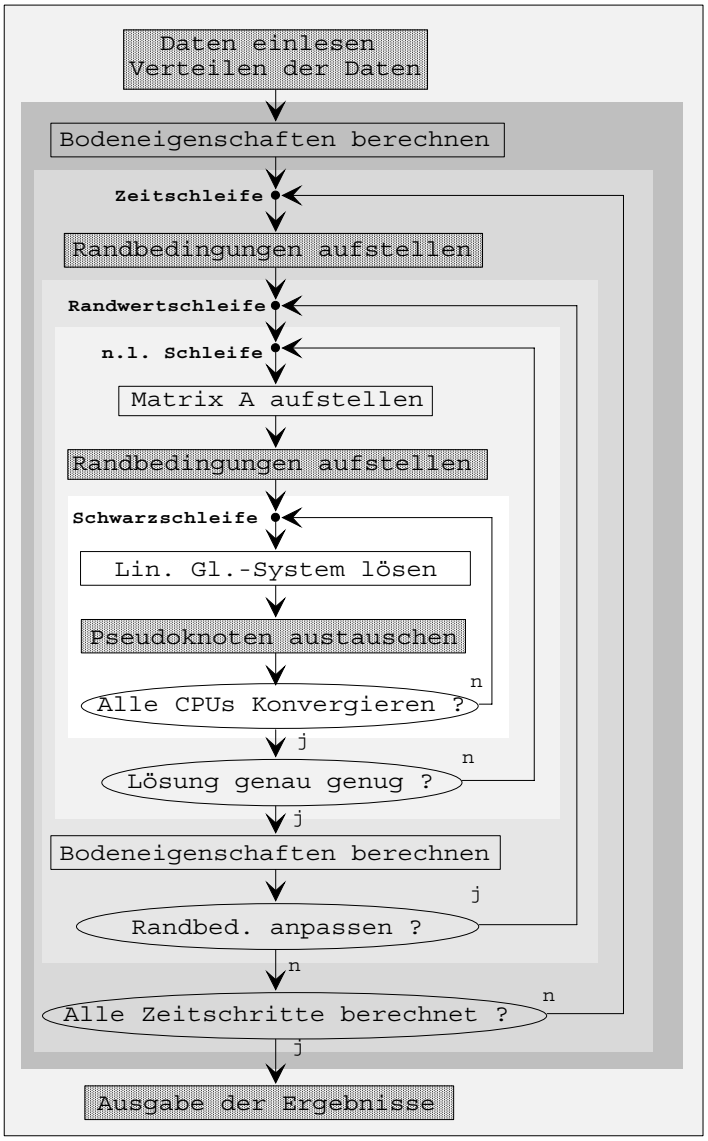


Abbildung 6.1: Struktur des Programms TRACE in der parallelen Version. Die neuen bzw. die zu ergänzenden Programmteile sind durch vertikale Balken auf der rechten Seite hervorgehoben.

Damit das Programm parallel ausgeführt werden kann, müssen zunächst die Bestimmung der Gebietszerlegung und die Verteilung der Daten durchgeführt werden. Hierzu übernimmt der Prozessor mit der Prozessornummer Null die Kontrolle des Programms und führt die notwendigen Berechnungen zur Bestimmung der Teilgebiete und der den Gebieten zugehörigen Daten durch. Nachdem dieser Prozessor eine geeignete Gebietszerlegung gefunden hat, liest er zeilenweise Daten aus der

WATDAT-Datei ein und schickt diese zu den Prozessoren, denen die entsprechenden Teilgebiete zugeordnet wurden. In Abbildung 6.1 sind diese Aufgaben in dem oberen Kasten "Daten einlesen, Verteilen der Daten" zusammengefaßt. Die Realisierung wird innerhalb der Routine DATAIN durchgeführt (siehe Abschnitt 6.2.4). Die Daten aus den anderen Eingabedateien sind für alle Prozessoren gleich, es braucht keine Berechnung zur Verteilung durchgeführt werden. Prozessor Null liest diese Dateien ebenfalls ein und verschickt sie an alle anderen Prozessoren.

Wenn alle Daten verteilt sind, besitzt jeder Prozessor sein eigenes Teilgebiet, auf dem er alle folgenden Berechnungen lokal durchführen kann. Hieraus ergibt sich, daß das Programm TRACE nach der Datenverteilung auf jedem Prozessor wie in der sequentiellen Programmversion ausgeführt werden kann.

Da durch die Gebietszerlegung neue Ränder entstanden sind, mußte die Routine zur Berechnung der Randbedingungen angepaßt werden. Für jeden neuen Rand, der kein Rand des gesamten Gebietes ist, werden Dirichlet-Randwerte angenommen, die in der Routine BC (vergleiche Abschnitt 6.2.5) gesetzt werden.

Im Vergleich zur sequentiellen Programmstruktur (siehe hierzu Abbildung 3.2) ist bei der parallelen Version innerhalb der schon vorhandenen Schleifen (Zeitschleife, Randwertschleife und nichtlineare Schleife) die Schwarz-Schleife neu hinzugekommen. Durch diese Schleife wird die Kopplung der einzelnen Prozessoren bei der Lösung des linearen Gleichungssystem möglich. Nachdem jeder Prozessor sein lokales linearisiertes Gleichungssystem gelöst hat, werden die Randwerte der benachbarten Prozessoren mit den berechneten Lösungen aktualisiert. Falls die Differenz des aktuellen Lösungsvektors und des Lösungsvektors aus dem vorhergehenden Iterationsschritt kleiner ist als ein vorgegebener Grenzwert, wird die Schwarz-Schleife verlassen. Falls der Grenzwert überschritten wird, wird das Gleichungssystem mit den aktualisierten Randwerten erneut von jedem Prozessor lokal gelöst. Der Austausch der Randwerte wird mit der Routine NODECOMM durchgeführt. Eine Beschreibung dieser Routine erfolgt in Abschnitt 6.2.6.

Nachdem alle Zeitschritte berechnet wurden, müssen die Ergebnisse der einzelnen Prozessoren zusammengefaßt und geeignet gespeichert werden. Hierzu sollte wieder, wie beim Einlesen der Daten, ein Prozessor die Kontrolle über das Programm übernehmen und sich die Daten der einzelnen Prozessoren sukzessive zuschicken lassen. Da sich die Teilgebiete überlappen, müssen die mehrfach auftretenden Werte gemittelt werden. Die Ausgabe der Ergebnisse soll in der vorliegenden Arbeit jedoch nicht weiter betrachtet werden.

6.2.2 Die Routinen aus HWUTIL

Damit das Programm TRACE ohne große Änderungen auf massiv-parallelen Rechnern verschiedener Hersteller ausgeführt werden kann, werden alle Befehle, die herstellerspezifisch und deswegen nicht standardisiert sind, nachgebildet (vergleiche Abschnitt 1.3). So wird beispielsweise zum Senden einer Nachricht der Befehl SEND benutzt, der seinerseits die herstellerspezifische Senderoutine, für die Intel Paragon ist dies die Routine CSEND, aufruft. Alle derartigen Befehle sind in der Datei HWUTIL.F zusammengefaßt, so daß bei einem Systemwechsel nur diese Datei an-

gepaßt werden muß.

In Tabelle 6.1 sind die vom Programm TRACE benötigten Befehle aufgeführt, die von HWUTIL zur Verfügung gestellt werden.

Routine	Intel Paragon	Erklärung
ASYNCSEND	ISEND	Asynchrones Senden
ASYNCTEST	IPROBE	Empfangsbuffer abfragen
GETETIME	DCLOCK	Zeitroutine
GIVEUP	FLICK	Programmkontrolle abgeben
GLOBALISUM	GISUM	Globale INTEGER-Summe
MYCPU	MYNODE	Bestimmung Prozessornummer
MYPID	MYPTYPE	Bestimmung Prozesstyp
NUMCPUS	NUMNODES	Gesamtzahl Prozessoren
RECEIVE	CRECV	Synchrones Empfangen
SEND	CSEND	Synchrones Senden
SYNCCPUS	GSYNC	Synchronisation aller Prozessoren
WAITTILSEND	MSGWAIT	Asynchrones Senden abwarten

Tabelle 6.1: Nachbildung herstellerspezifischer Befehle (Intel Paragon)

Eine ausführliche Beschreibung der Funktionalität und der erforderlichen Parameter der einzelnen Routinen für die Intel Paragon ist in [ISC93] angegeben.

6.2.3 Das Hauptprogramm

Im Hauptprogramm wurden an mehreren Stellen Fallunterscheidungen für den Einsatz eines oder mehrerer Prozessoren eingeführt. Im wesentlichen entspricht das Hauptprogramm aber noch immer der sequentiellen Version.

Die Umsetzung des Schwarz'schen Verfahrens soll durch den auf der folgenden Seite gezeigten Ausschnitt aus dem Programm TRACE dargestellt werden.

Falls nur ein Prozessor eingesetzt wird (IANZNO = 1), kann das linearisierte Gleichungssystem direkt mit dem Verfahren der konjugierten Gradienten (CGALG) gelöst werden.

Wenn mehrere Prozessoren verwendet werden, wird von jedem Prozessor die Schwarz-Schleife ausgeführt. Damit eine Kopplung der Prozessoren untereinander hergestellt werden kann, wird innerhalb der Schwarz-Schleife nach der Berechnung der Matrixgleichung die Routine NODECOMM aufgerufen. Diese Routine übernimmt alle Kommunikationsaufgaben zum Austausch der Pseudo-Randknoten. Mit der Variablen PARADIFMAX gibt die Routine NODECOMM die maximale Abweichung des Lösungsvektors von dem Lösungsvektor aus dem vorhergehenden Iterationsschritt zurück. Falls die Lösung aller Prozessoren der vorgegebenen Schranke EPSPARA genügt, wird die Schwarz-Schleife verlassen. Wenn mindestens ein Prozessor das Kriterium nicht erfüllt, wird für alle Prozessoren ein weiterer Iterationsschritt ausgeführt. Da sich durch den Austausch der Pseudo-Randknoten die Berechnungen verändern können, ist es in diesem Fall erforderlich, daß alle Pro-

zessoren das Gleichungssystem erneut lösen, auch wenn sie das Abbruchkriterium schon erfüllt haben.

```
      IF (IANZNO .EQ. 1) THEN
C        Keine parallele Ausfuehrung --> keine Gebietszerlegung
         PARADIFMAX = 0.0D0
         IPLOOP     = 1
         CALL CGALG(NNP, CMATRX, GNOJCN, DIAG, RLD, RI, LSGVEC,
     &              RL, RESEPS, TOLREL, seqMAXIT, VORKON,
     &              ANZIT, IDIAG, IPLOOP, ANZELZ, SQRTDIAG)
      ELSE
C        parallele Ausfuehrung --> Schwarz-Schleife
         DO 5555 iploop = 1, maxploop
            CALL CGALG(NNP, CMATRX, GNOJCN, DIAG, RLD, RI, LSGVEC,
     &                 RL, RESEPS, TOLREL, MAXIT, VORKON,
     &                 ANZIT, IDIAG, IPLOOP, ANZELZ, SQRTDIAG)

C           Schwarzsche Kommunikation
            CALL NodeComm(RL, RLALT, RI, RLD, NNP,
     &                    IPLOOP, PARADIFMAX,
     &                    NeiBuf, xNodes, yNodes, zNodes,
     &                    NPUCount, Overlap, ncDOP)

            IF (IPLOOP .GT. 1) THEN
               IF (PARADIFMAX. LE. EPSPARA) THEN
                  IKONV = 1
               ELSE
                  IKONV = 0
               ENDIF
            ENDIF

            IKONVOLD = IKONV
            CALL GlobalISUM(IKONV, IWORK)

            IF (IKONV .EQ. IANZNO) THEN
C              Sprung aus der Kommunikationsschleife
               GO TO 5560
            END IF

            ikonv = ikonvold

C           Schleifenende Kommunikationsschleife
 5555    CONTINUE

C Sprungadresse
 5560    CONTINUE

C Ende der parallelen Schwarz-Schleife
      END IF
```

6.2.4 Die Routine DATAIN

Die Routine DATAIN ist für das Einlesen und Verteilen der wichtigsten Eingabedateien verantwortlich. In der sequentiellen Version konnte die Datei *WATDAT* (Abschnitt 3.4.3) zeilenweise eingelesen und die Daten direkt in die dafür vorgesehenen Felder gespeichert werden.

Auch in der parallelen Version wird die *WATDAT*-Datei von einem Prozessor eingelesen. In Abhängigkeit von der Zahl der eingesetzten Prozessoren, der Dimensionierung des Simulationsgebietes und der Größe des Überlapps entscheidet dieser Prozessor über die durchzuführende Gebietszerlegung (Kapitel 4). Wenn eine geeignete Zerlegung gefunden wurde, interpretiert der einlesende Prozessor die Daten

und verschickt sie entweder sofort an die entsprechenden Prozessoren oder behält sie in seinen eigenen, dafür vorgesehenen Feldern. Da der Arbeitsspeicher des lesenden Prozessors nicht ausreicht, alle Daten, oder wenigstens einen Datensatz, vollständig einzulesen, müssen die Daten direkt den einzelnen Teilgebieten zugeordnet werden können. Im allgemeinen kann innerhalb eines Datensatzes eine willkürliche Reihenfolge in Bezug auf die Numerierung der FE-Knoten vorliegen, so daß für jeden Wert eine direkte Zuordnung zu einem oder mehreren Gebieten möglich sein muß. Die Kopplung von bestimmten Eigenschaften an einen FE-Knoten, beispielsweise die Nummer des vorgegebenen Profils für einen Dirichlet-Randknoten, wird durch die Position von Zahlenwerten innerhalb verschiedener Datenblöcke angegeben. Da nicht alle eingelesenen Daten gespeichert werden können, mußte die Struktur einzelner Datensätze innerhalb der *WATDAT*-Datei geändert werden, damit diese Abhängigkeiten aufgelöst werden konnten. Im Anhang D ist eine geänderte Datei angegeben.

Nachdem die *WATDAT*-Datei und die Eingabedatei für die Bodeneigenschaften eingelesen und verteilt worden sind, werden von allen Prozessoren lokal mehrere Berechnungen für verschiedene Felder durchgeführt. Als Beispiel soll die Umrechnung der globalen FE-Knotennummern in die lokale Numerierung erwähnt werden.

6.2.5 Die Routine BC

Die Routine BC wird für das Setzen der Randwerte des Simulationsgebietes verwendet. Um die durch die Gebietszerlegung entstandenen Teilgebiete berechnen zu können, müssen die Randwerte dieser Gebiete mit Werten belegt sein. Da diese Ränder auch im Inneren des Gesamtgebietes liegen können, werden dort die vorgegebenen Anfangswerte als Dirichlet'sche Randwerte gesetzt. Die Werte auf einem echten Rand werden entsprechend der sequentiellen Version ebenfalls von dieser Routine berechnet.

6.2.6 Die Routine NODECOMM

Innerhalb der Schwarz-Schleife übernimmt die Routine NODECOMM den Austausch der Pseudo-Randknoten. Die Realisierung der Kommunikation soll mit Hilfe eines Programmausschnittes dargestellt werden, wobei zu beachten ist, daß diese Routine von jedem Prozessor ausgeführt wird.

Nachdem ermittelt wurde, mit wievielen Prozessoren Pseudo-Randknoten ausgetauscht werden müssen (*NeiCount*), wird in die dargestellte REPEAT-UNTIL-Schleife verzweigt. Die Strategie zur Kommunikation basiert darauf, daß, falls noch kein Nachbarprozessor Botschaften gesendet hat, eigene Botschaften verschickt werden können. Falls jedoch mindestens eine Botschaft eingegangen ist, wird die zuerst empfangene Botschaft ausgewertet, bevor eigene Botschaften versendet werden dürfen. Der Empfang von Botschaften wird synchron durchgeführt, während das Senden asynchron, in Verbindung mit einem Wartebefehl, stattfindet.

Mit den Zählern *CtrSend* und *CtrRecv* wird die Zahl der empfangenen beziehungsweise gesendeten Botschaften verwaltet. Die REPEAT-UNTIL-Schleife wird

solange ausgeführt, bis mit allen Nachbarn Botschaften ausgetauscht wurden.

Die Routinen *GetNodeBuf* und *PutNodeBuf* haben die Aufgabe, das Heraussuchen und Speichern der Pseudo-Randknoten durchzuführen. Welche FE-Knoten des lokalen Gitters betroffen sind, ist in Abschnitt 5.3 beschrieben.

```
C ======================================================================
C        REPEAT - UNTIL - Schleife
C ======================================================================
 310     CONTINUE
         IF (CtrRecv .LT. NeiCount) THEN
            IF (ASyncTest(-1) .EQ. 1) THEN
C            Botschaft empfangen
               CALL RECEIVE(MAXCoBNP * 1DOUBLE, NodeBuf(1), -1)
C            empfangene Knoten in RLD eintragen
               CALL PutNodeBuf(RLD, NodeBuf, NPUCount,
     &                         xNodes, yNodes, zNodes, Overlap,
     &                         NeiBuf)
C            Empfangszaehler erhoehen
               CtrRecv = CtrRecv + 1
            ELSE
               IF (CtrSend .EQ. NeiCount) THEN
C                Kontrolle fuer 10ms abgeben
                  CALL GiveUp()
               END IF
            END IF
         END IF

         IF (CtrSend .LT. NeiCount) THEN
C         Botschaft senden
C         Nachbarn ermitteln, der Botschaft empfangen soll
 330        CONTINUE
            IF (NeiBuf(PosNeiBuf) .LT. 0) THEN
               PosNeiBuf = PosNeiBuf + 1
               GOTO 330
            END IF
C           auszutauschende Knoten aus RL bestimmen
            CALL GetNodeBuf(PosNeiBuf, RL, NodeBuf,
     &                      xNodes, yNodes, zNodes, Overlap,
     &                      NeiBuf)
C           Botschaft senden
            IDSend = ASyncSEND(NeiBuf(PosNeiBuf), ME, IANZNO,
     &                         MAXCoBNP * 1DOUBLE, NodeBuf(1),
     &                         ME, PID)
C           Warten, bis Botschaft CPU verlassen hat (Buffer wieder frei)
            CALL WaitTilSend(IDSend)
C           naechsten Nachbarn ermitteln
            PosNeiBuf = PosNeiBuf + 1
C           Sendezaehler erhoehen
            CtrSend = CtrSend + 1
         END IF

         IF ((CtrSend .LT. NeiCount) .OR. (CtrRecv .LT. NeiCount)) THEN
C         Schleife muss erneut ausgefuehrt werden
            GOTO 310
         END IF
C ======================================================================
C        Ende REPEAT - UNTIL - Schleife
C ======================================================================
```

Kapitel 7

Ergebnisse auf der Intel Paragon XP/S

In diesem Kapitel werden zwei Simulationen für unterschiedliche Gittergrößen vorgestellt, die mit dem Programm TRACE auf einer Intel Paragon XP/S 10 durchgeführt wurden. Die erste Simulation (SIM1) wurde für unterschiedliche diskrete Gitter bestehend aus 9216, 29791 und 256000 FE-Knoten berechnet. Bei diesen Berechnungen konnte eine große Zahl von Prozessoren eingesetzt werden, da die Gebietsgrößen eine dreidimensionale Gebietszerlegung zuließ. Bei der zweiten Simulation (SIM2) ist das Modell aus [GER94], [VNL94] verwendet worden. Hierbei wurde ein Gitter eingesetzt, welches aus 9216 FE-Knoten bestand. Die Berechnungen für das zweite Modell gestatteten einen Vergleich der neuen parallelen Version von TRACE mit der in [GER94] eingesetzten Programmversion.

Die Komplexität der Berechnungen ist für das Modell SIM2 erheblich größer als für SIM1. Zunächst sind deshalb mehrere Rechnungen für SIM1 durchgeführt worden, da bei diesem Modell eine einfachere Interpretation der Berechnungsergebnisse möglich war. Bei beiden Simulationen wurden jeweils nur wenige Zeitschritte berechnet. Diese Zahl kann sich bei realistischeren Simulationsläufen auf mehrere hundert Zeitschritte erhöhen.

Zunächst werden die beiden Modelle betrachtet, die den Berechnungen zugrunde lagen. Daran schließt sich eine Betrachtung der verschiedenen Programmläufe an, wobei vor allem die Skalierbarkeit des Programms untersucht werden soll. Die Berechnungen sind jeweils für unterschiedliche Prozessorzahlen und einen variierenden Überlapp durchgeführt worden. Die Erkenntnisse, die bei der Variation dieser Parameter gewonnen wurden, werden ebenfalls in diesem Kapitel vorgestellt.

7.1 Beschreibung der eingesetzten Modelle

7.1.1 Das Simulationsmodell SIM1

Bei der ersten Simulation (SIM1) wird der Wasserfluß in einem dreidimensionalen Bodenbereich berechnet, der in der gesättigten Zone, also unterhalb des Grundwasserspiegels, liegt. Der Boden besitzt eine homogene Permeabilität. Innerhalb des

Bereiches sind die Anfangswerte für die Simulation so gewählt, daß kein Wasserfluß auftritt. Durch entsprechende Vorgabe von Dirichlet-Randwerten für das gesamte Simulationsgebiet entsteht ein Wasserfluß in das Gebiet hinein. Berechnet werden 15 Zeitschritte, was einer Simulationsdauer von 76 Sekunden entspricht. Am Ende der Simulation hat sich der Potentialunterschied, hervorgerufen durch die gewählten Randwerte, ausgeglichen, so daß kein weiterer Wasserfluß mehr auftritt.

Dieses Modell wird für drei verschiedene, quaderförmige Gebiete berechnet. Der Abstand zwischen zwei FE-Knoten beträgt in jedem der Gebiete in jeder Raumrichtung jeweils 5 cm. Zunächst wird ein Gebiet betrachtet, welches aus 6x6x256 (9216) FE-Knoten besteht. Die Größe dieses Gebietes ist so gewählt, daß es noch mit einem Prozessor berechnet werden kann. Bei Verwendung mehrerer Prozessoren können jedoch nur eindimensionale Zerlegungen in der z-Richtung durchgeführt werden. Das zweite Gebiet setzt sich aus 31x31x31 (29791) FE-Knoten zusammen, so daß eine dreidimensionale Gebietszerlegung bei entsprechender Prozessorzahl möglich ist. Für das dritte Gebiet werden 80x80x40 (256000) FE-Knoten benötigt, so daß eine sinnvolle Auslastung einer großen Prozessorzahl erreicht werden kann.

Die räumliche Ausdehnung der drei betrachteten Gebiete ist verhältnismäßig klein. Die Zahl der FE-Knoten des dritten Gebietes entspricht jedoch den Anforderungen einer realistischen Simulation, wobei hier häufig eine geringere Auflösung des FE-Gitters verwendet wird.

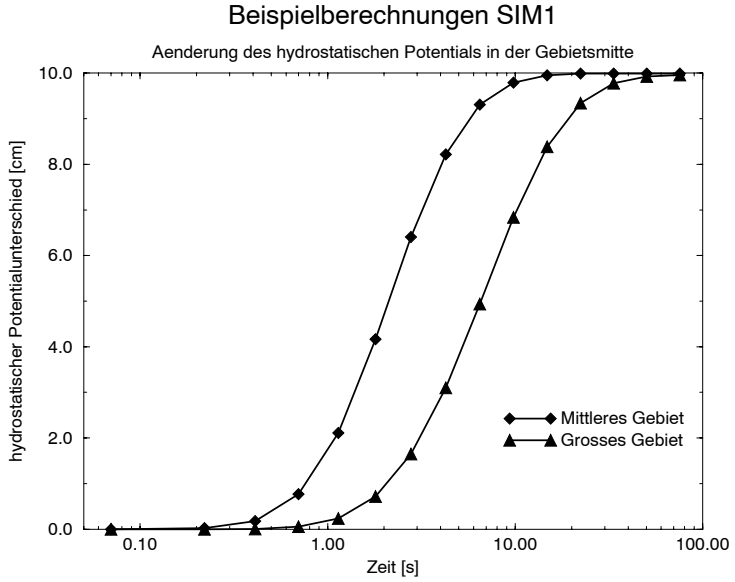


Abbildung 7.1: Zeitliche Veränderung des hydrostatischen Potentials im Inneren des Simulationsgebietes (Modell: SIM1) für das mittlere und das große Simulationsgebiet

Den Verlauf der Änderung des simulierten hydrostatischen Potentials Ψ über der Zeit zeigt Abbildung 7.1. Die Ergebnisse sind jeweils für einen FE-Knoten dargestellt, der in der Mitte des mittelgroßen beziehungsweise des großen Simulationsgebietes liegt. Die Änderung des Potentials wird auf den Anfangswert der Simulation

bezogen. Die Zeitachse ist zur besseren Übersichtlichkeit logarithmisch geteilt. Da die beiden betrachteten Gebiete eine unterschiedliche räumliche Ausdehnung besitzen, ist der Abstand der FE-Knoten, für die der Verlauf des Potentials dargestellt ist, von den jeweiligen Gebietsrändern verschieden. Hieraus erklärt sich der zeitliche Unterschied für den Anstieg des hydrostatischen Potentials, da das zuströmende Wasser in dem großen Gebiet einen weiteren Weg zurücklegen muß.

Bei den durchgeführten Berechnungen stimmten die numerischen Ergebnisse innerhalb der möglichen Genauigkeit mit den analytischen Lösungen überein. Insbesondere waren die Resultate unabhängig von der Zahl der eingesetzten Prozessoren und der Größe des Überlappungsbereiches.

7.1.2 Das Simulationsmodell SIM2

Für die zweite Simulation (SIM2) wird ein dreidimensonaler Bodenbereich betrachtet, in dem eine homogene Permeabilität vorliegen soll. Innerhalb dieses Bereiches liegen sowohl die ungesättigte als auch die gesättigte Zone. Die obere Berandung wird durch den Übergang Atmosphäre-Boden dargestellt.

Das Gebiet wird mit 6x6x256 (9216) FE-Knoten diskretisiert. An den Rändern werden bis auf den Übergang Atmosphäre-Boden Dirichlet-Randwerte angenommen. Die Randwerte auf der oberen Berandung werden als variable Randbedingungen gesetzt. In Abhängigkeit der Sättigung des Bodens kann ihr Typ entweder eine Neumann'sche oder eine Dirichlet'sche Bedingung sein. Ebenso wie bei SIM1 beträgt der räumliche Abstand zwischen zwei FE-Knoten in jeder Raumrichtung jeweils 5 cm.

Simuliert wird der Verlauf des Grundwassers in Abhängigkeit von Niederschlägen und Verdunstung an der oberen Grenzfläche. Insgesamt werden 14 Zeitschritte berechnet; dies entspricht einer Simulationsdauer von 6 Stunden. Im Vergleich zu SIM1, bei der durch die Vorgabe eines relativ großen Potentialunterschiedes zeitlich sehr schnelle Veränderungen des hydrostatischen Potentials auftraten, wird bei SIM2 ein längerer Zeitraum simuliert, da der Einfluß von Niederschlägen und Verdunstung in der ungesättigten Zone in einem kleinen Zeitintervall nur geringe Veränderungen bewirkt.

7.2 Darstellung der Ergebnisse

7.2.1 Allgemeines

Auf der Intel Paragon wurde während der Programmläufe das Betriebssystem OSF/1 Version 1.2.7 eingesetzt. Als Compiler wurde der FORTRAN 77 Compiler if77 von Intel (Version 4.5) [IFF77] mit den Optionen -nx, -O4 und -Knoieee verwendet.

Für alle in diesem Abschnitt angegebenen Berechnungen wurde das Abbruchkriterium der nicht-linearen Schleife auf 10^{-3} und die Abbruchkriterien für das Verfahren der konjugierten Gradienten beziehungsweise für die Schwarz-Schleife auf 10^{-5} gesetzt.

Im Anhang E sind zu den in diesem Abschnitt dargestellten Abbildungen Tabellen angegeben, die die Ausführungszeiten des Programms TRACE und die gewählten Gebietszerlegungen wiedergeben.

Damit eine getrennte Betrachtung der Laufzeiten für die Berechnungen und das Einlesen und Verteilen, beziehungsweise die Ausgabe der Daten erfolgen kann, sind jeweils die Zeiten für die Ausführung des Gesamtprogramms und für die Zeitschleife gemessen worden.

7.2.2 Ergebnisse für SIM1

SIM1 für ein Gebiet mit 9216 FE-Knoten

Abbildung 7.2 zeigt den Speedup des Gesamtprogramms und der Zeitschleife, der bei der Berechnung des kleinen Beispiels für SIM1 mit 1, 2, 4, 8, 16 und 32 Prozessoren erreicht wurde. Bei allen Programmläufen wurden nur Zerlegungen in der z-Richtung durchgeführt, da die Geometrie des Simulationsgebietes nur eine 1D-Zerlegung zuließ.

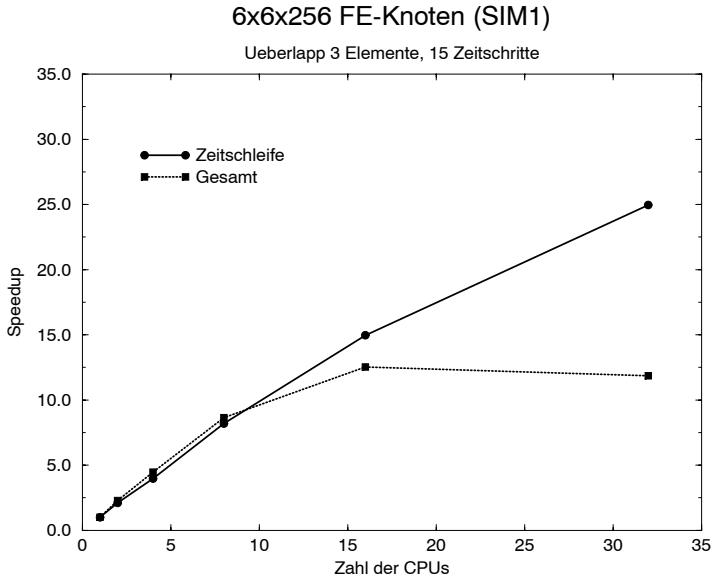


Abbildung 7.2: Speedup des Gesamtprogramms und der Zeitschleife bei der Berechnung von SIM1 mit 9216 FE-Knoten

Wie die Abbildung 7.2 zeigt, steigt der Speedup für die Zeitschleife bei allen eingesetzten Prozessorzahlen an. Der Speedup für das Gesamtprogramm liegt bei Verwendung von 1, 2, 4 und 8 Prozessoren sogar über dem Speedup der Zeitschleife und zeigt hier superlineares Verhalten. Eine Untersuchung der Gründe hat ergeben, daß die Routine PAGEN, die außerhalb der Zeitschleife aufgerufen wird, bei der Verwendung von nur einem Prozessor eine circa fünfzehnfach längere Ausführungszeit besitzt als bei Verwendung von vier Prozessoren. Die Laufzeit der Routine PAGEN ist in erster Näherung proportional zum Quadrat der Anzahl der FE-Knoten. Ein

Zeitgewinn in der Größenordnung 16 ist daher zu erwarten, wenn, wie in diesem Fall, auf nur einem Viertel der Daten operiert wird.

Bei größeren Prozessorzahlen steigt der Speedup langsamer an, was durch die redundanten Berechnungen der FE-Knoten im Überlappungsbereich der Teilgebiete und dem erhöhten Kommunikationsaufwand während der Berechnungen begründet werden kann. Bei Verwendung von 32 Prozessoren macht sich die Verteilung der Daten beim Programmstart stärker bemerkbar, so daß der Speedup für das Gesamtprogramm zurückgeht.

SIM1 für ein Gebiet mit 29791 FE-Knoten

Das mittlere Gebiet für die Simulation SIM1 konnte nicht mehr auf einem Prozessor berechnet werden, da die Größe des Arbeitsspeichers dies nicht zuläßt. Eine Angabe für den Speedup ist deshalb nicht möglich. Das Gebiet ist so dimensioniert, daß bei einer entsprechend gewählten Prozessorzahl eine dreidimensionale Gebietszerlegung durchgeführt werden kann.

Die in den folgenden Abbildungen dargestellten Ergebnisse wurden bei der Ausführung des Programms TRACE mit 4, 8, 16, 27, 32 und 64 Prozessoren erhalten. Bei allen Programmläufen, ausgenommen der 4-Prozessorlauf, wurde das Simulationsgebiet dreidimensional zerlegt.

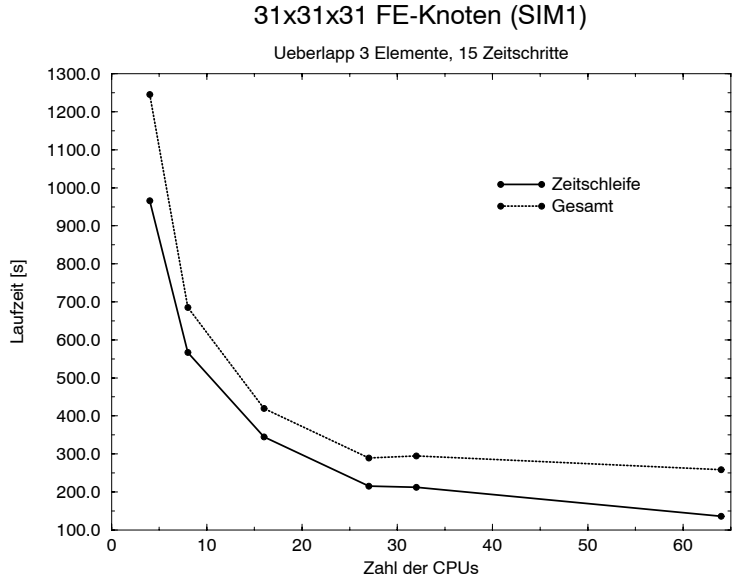


Abbildung 7.3: Laufzeit des Gesamtprogramms und der Zeitschleife bei der Berechnung von SIM1 mit 29791 FE-Knoten

Abbildung 7.3 zeigt, daß die Laufzeit des Programms TRACE für dieses Beispiel bis zum Einsatz von 27 Prozessoren gut skaliert. Die Laufzeit der Zeitschleife verringert sich etwa um einen Faktor 4,5, wenn 6 mal soviele Prozessoren eingesetzt werden. Ebenso wie bei der Berechnung des kleinen Gebietes für SIM1 kann auch bei diesem Beispiel eine relative Verschlechterung des Verhältnisses von Gesamtprogramm zu Zeitschleife bei großen Prozessorzahlen festgestellt werden. Auch hier

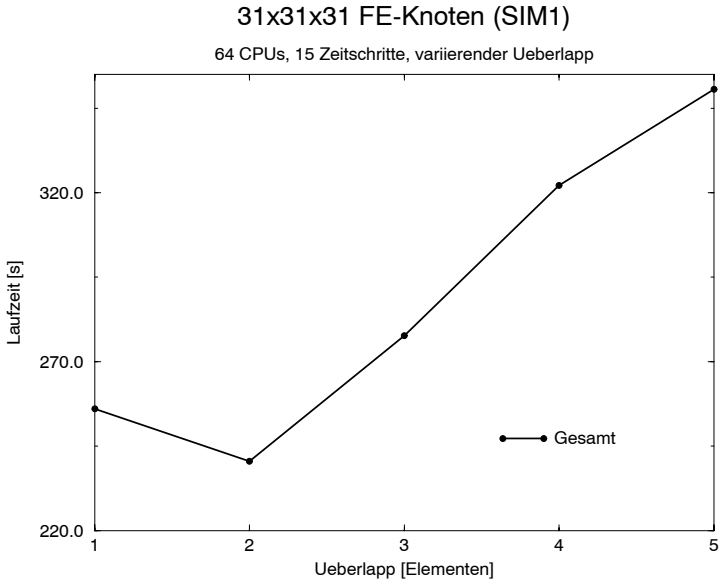


Abbildung 7.4: Einfluß der Größe des Überlapps auf die Laufzeit des Gesamtprogramms bei der Verwendung von 64 Prozessoren

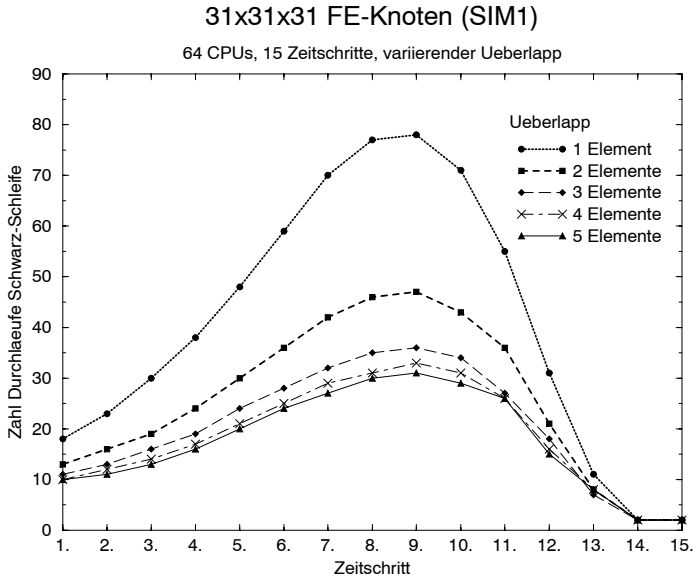


Abbildung 7.5: Zahl der Durchläufe der Schwarz-Schleife in Abhängigkeit des Überlapps und des Zeitschritts beim Einsatz von 64 Prozessoren

macht sich der gestiegene Aufwand für die Zerlegung des Gebietes in eine große Zahl von Teilgebieten bemerkbar.

Abbildung 7.4 stellt die Gesamtlaufzeit des Programms für die Berechnung mit 64 Prozessoren bei variierendem Überlapp dar. Das Simulationsgebiet wurde bei diesen Berechnungen in jeder Raumrichtung jeweils dreimal geschnitten. Bei einem Überlapp von 5 Elementen besitzt jeder Prozessor maximal ein Teilgebiet der Größe 13x13x13 FE-Knoten. Die Wahl eines Überlapps von mehr als 5 Elementen ist für diese Zerlegung nicht mehr möglich. Bei einem Überlapp von 2 Elementen wird ein Minimum für die Programmlaufzeit beobachtet.

In Abbildung 7.5 ist die Zahl der Durchläufe der Schwarz-Schleife für die Programmläufe mit 64 Prozessoren und variierendem Überlapp von einem bis zu fünf Elementen über der Nummer des Zeitschrittes aufgetragen. Hierbei kann erkannt werden, daß die Schwarz-Schleife bei größer werdendem Überlapp seltener ausgeführt werden muß, obwohl die Größe der zu berechenden Teilgebiete zunimmt. Bezogen auf die Gesamtlaufzeit des Programms ist jedoch entsprechend Abbildung 7.4 ein Überlapp von 2 Elementen das Optimum.

Bei den für das Modell SIM1 durchgeführten Berechnungen nahm die Zeitschrittlänge in jedem Zeitschritt zu. Die sich verlängernden Zeitschritte begründen die Zunahme der Zahl der Durchläufe der Schwarz-Schleife in den ersten Zeitschritten. Da bei SIM1 eine konstante Lösung nach einer bestimmten Zeitschrittzahl erreicht wird, fällt die Zahl der Durchläufe wieder ab, da die berechnete Lösung im Rahmen der numerischen Genauigkeit schon gut approximiert wurde. In den letzten beiden Zeitschritten kann keine Verbesserung der Lösung mehr erreicht werden, weswegen die Schwarz-Schleife nur noch mit der im Programm vorgegebenen Mindestzahl von 2 Aufrufen durchlaufen wird.

SIM1 für ein Gebiet mit 256000 FE-Knoten

Um Aussagen über die Partitionierungsstrategie und das Laufzeitverhalten des Programms TRACE für Gebiete mit einer großen Zahl von FE-Knoten treffen zu können, ist das Modell SIM1 ebenfalls für ein Gebiet mit 256000 FE-Knoten berechnet worden. Die Zahl der FE-Knoten befindet sich hierbei in der Größenordnung der bei realistischen Simulationen angestrebten Gitterauflösung.

Die Simulation SIM1 ist für das große Gebiet mit 32, 64, 96, 108, 128 und 136 Prozessoren berechnet worden. Abbildung 7.6 zeigt die gemessenen Laufzeiten für das Gesamtprogramm und für die Zeitschleife. Die Berechnungen wurden für 15 Zeitschritte und einen Überlapp von 3 Elementen durchgeführt.

Bis zu einem Einsatz von 108 Prozessoren nimmt die Laufzeit der Zeitschleife bei den Berechnungen für dieses Beispiel ab. Eine Vergrößerung der Prozessorzahl um einen Faktor 3,4 bedeutet eine Verringerung der Ausführungszeit für die Zeitschleife um den Faktor 2,4. Der Einsatz einer größeren Zahl von Prozessoren führt hingegen zu einer Verschlechterung der Laufzeit des Programms.

Die Gründe für diese Verschlechterung liegen darin, daß bei den vorgegebenen Prozessorzahlen nur Gebietszerlegungen möglich sind, die eine große Zahl von redundant zu berechnenden Überlappknoten besitzen. Ein Vergleich mit der in Abschnitt 4.3 definierten K-Funktion zeigt, daß bei Verwendung von 136 Prozessoren 2,3 mal

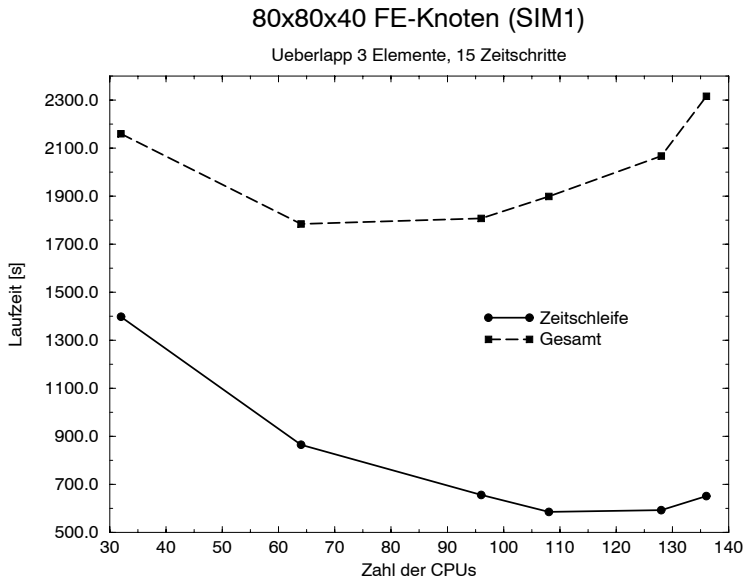


Abbildung 7.6: Laufzeit des Gesamtprogramms und der Zeitschleife bei der Berechnung von SIM1 mit 256000 FE-Knoten

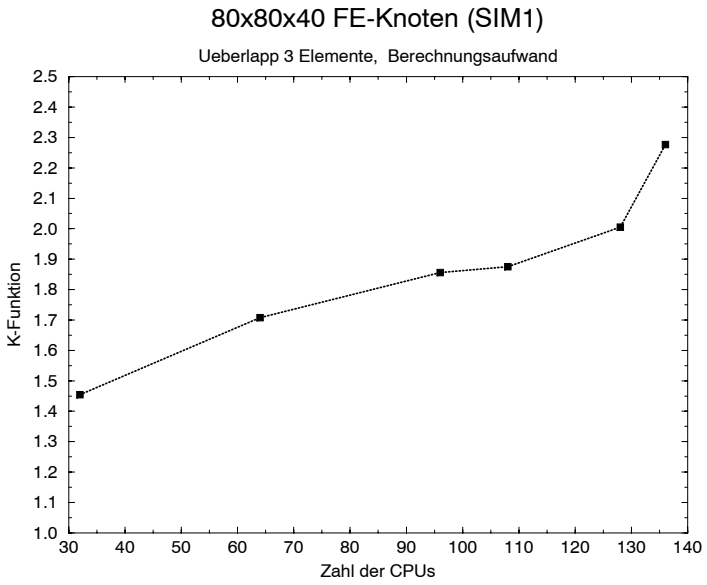


Abbildung 7.7: Die K-Funktion für 80x80x40 FE-Knoten und drei Elemente Überlapp bei der Zerlegung des Gebietes für den Einsatz von 32, 64, 96, 108, 128 und 136 Prozessoren

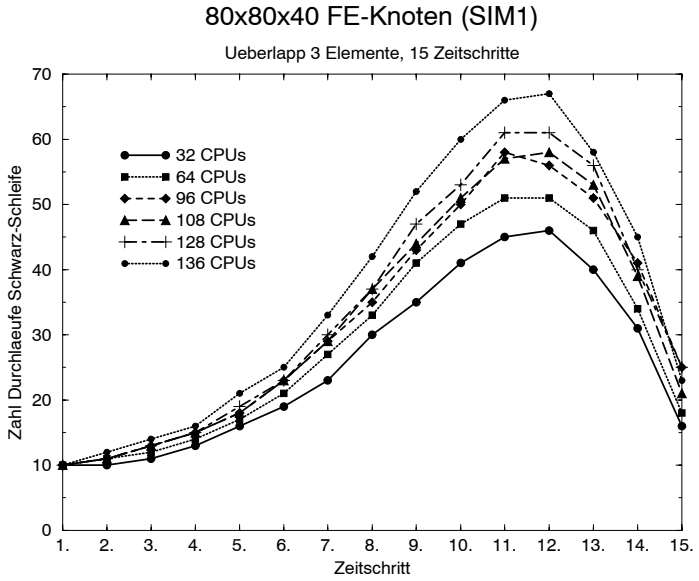


Abbildung 7.8: Zahl der Durchläufe der Schwarz-Schleife in Abhängigkeit der Zahl der eingesetzten Prozessoren und des Zeitschritts

mehr FE-Knoten berechnet werden müssen, als im zu berechnenden Gesamtgebiet vorhanden sind. In Abbildung 7.7 ist die K-Funktion für dieses Gebiet bei einer Zerlegung mit drei Elementen Überlapp für die hier betrachteten Prozessorzahlen dargestellt.

Ein Vergleich der Abbildungen 7.6 und 7.7 legt nahe, daß die Vorgabe einer bestimmten Prozessorzahl zu einer Minimierung der Laufzeit des Programms führen kann. Bei Simulationen mit vielen Zeitschritten können mit Hilfe der K-Funktion geeignete Prozessorvorgaben gefunden werden, die bei Probeberechnungen mit kleiner Zeitschrittzahl auf ein Minimum der Ausführungszeit für die Zeitschleife überprüft werden können.

Auch bei diesem Beispiel kann eine relative Zunahme der Laufzeit des Gesamtprogramms gegenüber der Laufzeit für die Zeitschleife erkannt werden. Da bei der Gebietszerlegung für 136 Prozessoren viele FE-Knoten in den gemeinsamen Überlappbereichen liegen, müssen diese Knoten beim Programmstart von dem einlesenden Prozessor mehrfach verschickt werden, wodurch die Zunahme der Laufzeit für diese Fälle zum Teil erklärt werden kann.

Für die Gesamtlaufzeit des Programms zeigt Abbildung 7.6 für 64 Prozessoren ein Minimum. Da der Einfluß der Gebietszerlegung bei vielen Zeitschritten jedoch relativ abnimmt und die Zeitschleife die weitere Laufzeit des Programms domimiert, erscheint für dieses Beispiel der Einsatz von 108 Prozessoren bei vielen Zeitschritten am günstigsten zu sein.

In Abbildung 7.8 ist die Zahl der Durchläufe der Schwarz-Schleife über der Nummer des berechneten Zeitschritts für unterschiedliche Prozessorzahlen aufgetragen.

Erkannt werden kann, daß die Schwarz-Schleife bei größeren Prozessorzahlen öfter ausgeführt werden muß, was jedoch nicht zwingend die Laufzeit des Programms erhöht.

7.2.3 Ergebnisse für SIM2

Das Modell SIM2 wurde mit verschiedenen Prozessorzahlen zwischen 1 und 64 berechnet. In Abhängigkeit von der Zahl der Teilgebiete und der Größe des Überlapps konnten alle Zerlegungen jedoch nur für Schnitte in der z-Richtung durchgeführt werden, da die Dimensionierung des Simulationsgebietes keine anderen Schnitte zuließ.

In diesem Abschnitt werden jeweils die Ergebnisse des Prozessors betrachtet, der das oberste Teilgebiet berechnet. Durch den Einfluß der variablen Randbedingungen ergibt sich in diesem Teilgebiet der größte Berechnungsaufwand.

Ein Vergleich mit den Berechnungsergebnissen der alten Programmläufe, die in [GER94] angegeben sind, belegt die Übereinstimmung beider Programmversionen. Im weiteren konnte bei dem Modell SIM2, entsprechend zu SIM1, kein Einfluß der Zahl der Teilgebiete oder der Größe des Überlapps auf die Berechnungsergebnisse festgestellt werden.

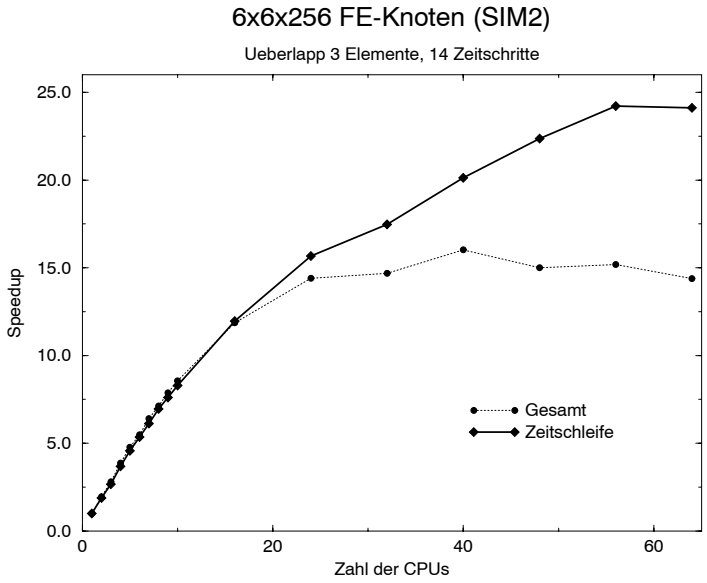


Abbildung 7.9: Speedup des Gesamtprogramms und der Zeitschleife bei der Berechnung von SIM2

Abbildung 7.9 zeigt den Speedup, der bei einem Überlapp von 3 Elementen und einer Berechnungsdauer von 14 Zeitschritten erzielt wurde. Ein Vergleich mit den in [GER94] angegebenen Meßwerten zeigt keine wesentlichen Änderungen für die Ausführung des Programms TRACE auf 1,4 oder 8 Prozessoren. Bei 8 Prozessoren ist eine leichte Verbesserung des Speedups zu erkennen, jedoch wurde auf der Intel

Paragon bei den alten Testläufen eine andere Version des Betriebssystems eingesetzt, die noch nicht den auf jedem Knoten vorhandenen Message-Prozessor unterstützte.

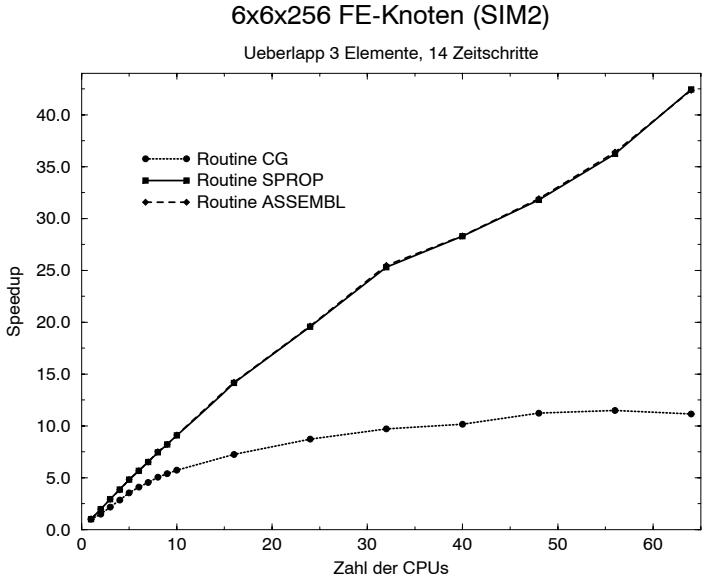


Abbildung 7.10: Speedup für rechenintensive Routinen (SIM2). Der Speedup für die Routinen ASSEMBL und SPROP verläuft weitgehend identisch, weswegen sich die beiden Kurven überlagern

Da die automatische Gebietszerlegung es zuläßt, das Programm auf einer beliebigen Anzahl von Prozessoren auszuführen, ist die in [GER94] beschriebene manuelle Generierung der Eingabedateien nicht mehr nötig. Die Ergebnisse für den Speedup der Zeitschleife zeigen entsprechend Abbildung 7.9 für bis zu 16 Prozessoren ein lineares Verhalten. Erst für größere Prozessorzahlen verschlechtern sich die Werte, was durch den gestiegenen Kommunikationsaufwand und die redundanten Berechnungen für die FE-Knoten in den Überlappbereichen begründet werden kann. Entsprechend den Ergebnissen von SIM1 nimmt der Speedup für das Gesamtprogramm bei größeren Prozessorzahlen wieder relativ zur Zeitschleife ab. Die Ursache hierfür ist wiederum der Einfluß der Datenverteilung beim Programmstart.

Um einen Vergleich verschiedener, rechenintensiver Routinen des Programms TRACE bei unterschiedlicher Prozessorzahl durchführen zu können, sind die Ausführungszeiten der Routinen jeweils ins Verhältnis zur Ausführungszeit mit nur einem Prozessor gesetzt worden. Abbildung 7.10 stellt den Speedup für diese Routinen dar.

Der Berechnungsaufwand der Routine ASSEMBL (Aufstellen der Elementmatrix) und der Routine SPROP (Berechnung der Bodeneigenschaften) hängt direkt von der Zahl der FE-Knoten des Teilgebietes ab. Da die Teilgebiete bei größer werdenden Prozessorzahlen kleiner werden, nimmt der Berechnungsaufwand für ASSEMBL und SPROP ab, weswegen Abbildung 7.10 eine gute Skalierbarkeit zeigt. Die Routine CG (Lösung der linearisierten Matrixgleichung mit dem Verfahren

der konjugierten Gradienten) berechnet iterativ innerhalb der Schwarz-Schleife die Lösungsvektoren. Bei kleiner werdenden Gebieten, entsprechend größeren Prozessorzahlen, skaliert die Routine CG schlechter als ASSEMBL und SPROP. Sie weist bei Verwendung von 64 Prozessoren nur noch einen Speedup von 11,1 auf. Daher begrenzt die Routine CG den Speedup für die Zeitschleife, obwohl die Routinen ASSEMBL und SPROP einen absolut gesehen höheren Zeitbedarf haben als die Routine CG.

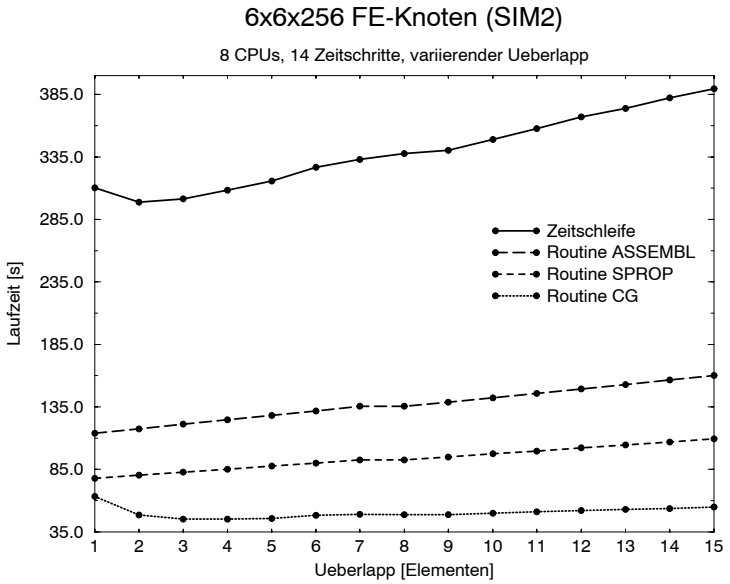


Abbildung 7.11: Einfluß des Überlapps auf die Laufzeit der Berechnungen für SIM2

Eine für dieses Beispiel durchgeführte Untersuchung des Einflusses eines variierenden Überlapps auf die Programmlaufzeit hat ergeben, daß ein Überlapp von zwei Elementen, entsprechend den Programmläufen aus SIM1, zu den besten Ergebnissen führt. In Abbildung 7.11 sind die Ergebnisse der Programmläufe mit jeweils 8 Prozessoren für einen Überlapp zwischen einem und fünfzehn Elementen angegeben. Die Laufzeit der Zeitschleife besitzt für einen Überlapp von 2 Elementen ein Minimum, während die Routine CG für einen Überlapp von 3 bzw. 4 Elementen die kürzesten Laufzeiten besitzt. Da der Berechnungsaufwand der Routinen ASSEMBL und SPROP von der Zahl der FE-Knoten abhängt, steigt der Zeitbedarf bei wachsendem Überlapp linear an, da die Gebietsgröße entsprechend dem Überlapp zunimmt.

Kapitel 8

Zusammenfassung und Ausblick

In der vorliegenden Arbeit ist eine dreidimensionale Partitionierungsstrategie für das Schwarz'sche Verfahren entwickelt und im Programm TRACE implementiert worden. Die erarbeitete Strategie gestattet die Ausführung des Programms TRACE auf einem massiv-parallelen Rechnersystem, wobei die Zahl der einzusetzenden Prozessoren frei gewählt werden kann, ohne daß vom Bediener eine Datenverteilung auf die einzelnen Prozessoren vorgegeben werden muß.

Die sequentielle Struktur des Programms konnte bei der Parallelisierung weitgehend erhalten bleiben, da das Schwarz'sche Verfahren nur eine schwache Kopplung der Prozessoren vorsieht. Für die Umsetzung der Partitionierungsstrategie war es erforderlich, daß vom Programm TRACE eine automatische Verteilung der Daten zu Beginn der Programmausführung in Abhängigkeit von der gewählten Gebietszerlegung durchgeführt wird. Damit die Kopplung der Prozessoren während der Berechnungen hergestellt werden kann, mußte eine Definition für die von den Prozessoren auszutauschenden Bereiche gefunden werden.

Programmläufe mit bis zu 136 Prozessoren auf einer Intel Paragon XP/S 10 mit Simulationsgebieten, die aus maximal 256000 FE-Knoten bestanden, haben gezeigt, daß auch Berechnungen mit mehreren Millionen FE-Knoten möglich sind. Die Berechnungen führten jeweils unabhängig von der Zahl der eingesetzten Prozessoren immer zu den gleichen numerischen Lösungen.

Die bei der parallelen Programmausführung erzielten Ergebnisse bestätigen die Effizienz des Schwarz'schen Verfahrens, wobei jedoch die Performance des Programms bei großen Prozessorzahlen durch die redundanten Berechnungen in den Überlappungsbereichen der Teilgebiete und durch den Aufwand für die Verteilung der Daten während des Programmstarts beschränkt wird. Mit Hilfe der in dieser Arbeit eingeführten K-Funktion kann eine Abschätzung des vergrößerten Berechnungsaufwands, hervorgerufen durch die Gebietszerlegung, durchgeführt werden. Hiermit wird das Auffinden einer geeigneten Prozessorzahl zur Programmausführung erleichtert.

Der Zeitbedarf für die Verteilung des Gesamtgebietes auf die Prozessoren hatte in den Simulationsrechnungen mit einer großen Prozessorzahl einen wesentlichen Einfluß auf die Gesamtlaufzeit des Programms. Da sich die Programmlaufzeit bei realistischen Simulationsrechnungen durch die große Zahl von Zeitschritten jedoch

deutlich verlängert, nimmt der Anteil für die Gebietszerlegung relativ ab. Eine Optimierung der hierfür zuständigen Routinen erscheint dennoch sinnvoll.

Mit der vorliegenden parallelen Programmversion von TRACE kann eine Untersuchung des Einflusses bestimmter Gebietszerlegungen, der Größe des Überlappungsbereiches und der Wahl geeigneter Abbruchkriterien für die einzelnen numerischen Verfahren durchgeführt werden. So hat sich gezeigt, daß eine Überlappung der Teilgebiete von zwei Elementen, anders als bei Rodrigue [ROD89], zu den schnellsten Programmlaufzeiten bei den Berechnungen der Modelle SIM1 und SIM2 geführt hat. Die Beschleunigung der Konvergenzrate des Schwarz'schen Verfahrens, gerade bei der Verwendung vieler Prozessoren, ist ein weiterer Aspekt, der zukünftig untersucht werden sollte.

Da neben einer guten Performance des Programms bei paralleler Ausführung auch die Größe des verfügbaren Arbeitsspeichers zur Berechnung einer möglichst großen Zahl von FE-Knoten von Bedeutung für realistische Simulationen ist, bietet die Partitionierungsstrategie einen Weg, viele Prozessoren mit lokalem Arbeitsspeicher einzusetzen. Die Ausführung von TRACE auf zwei miteinander gekoppelten massiv-parallelen Rechnersystemen erscheint in Hinsicht auf die Vergrößerung des verfügbaren Arbeitsspeichers interessant, da zudem nur eine schwache Kopplung der Prozessoren erforderlich ist.

Die Zusammenfassung und Ausgabe der Berechnungsergebnisse der einzelnen Prozessoren ist in der parallelen Programmversion von TRACE noch nicht implementiert. Eine Lösung dieses Problems könnte aufbauend auf die in der vorliegenden Arbeit realisierte Partitionierungsstrategie erarbeitet werden. Die Parallelisierung des Programmteils zur Berechnung des Stofftransportes im Grundwasser ist eine ebenfalls noch durchzuführende Aufgabe.

Literaturverzeichnis

[AMD67] G. Amdahl, *Validity of the Single-Processor Approach to Achieving Large Scale Computing Capabilities*, AFIPS Conference Proceedings 1967, Vol. 30, AFIPS Press, Montvale, pp. 483-485, 1967

[BAS95] A. Basermann, *Iterative Verfahren für dünnbesetzte Matrizen zur Lösung technischer Probleme auf massiv-parallelen Systemen*, Forschungszentrum Jülich, ZAM, Bericht Jül-3015, 1995

[BER93] R. Berrendorf, M. Gerndt, W.E. Nagel, J. Prümmer, *SVM-Fortran*, Forschungszentrum Jülich, ZAM, No. IB-9322, 1993

[BOE94] T. Bönniger, R. Esser, D. Krekel, *CM-5E, KSR2, Paragon XP/S: A Comparative Description of Massively Parallel Computers*, Forschungszentrum Jülich, ZAM, No. IB-9419, 1994

[BJO89] P.E. Bjørstad, *Multiplicative and Additive Schwarz Methods: Convergence in the Two-Domain Case*, in Domain Decomposition Methods, T. Chan, R. Glowinski, J. Périaux, O. Widlund, Eds., SIAM, Philadelphia, pp. 147-159, 1989

[BJO90] P.E. Bjørstad, R. Moe, M. Skogen, *Parallel Domain Decomposition and Iterative Refinement Algorithms*, in Parallel Algorithms for Partial Differential Equations, W. Hackbusch, Ed., Proceedings of the Sixth GAMM-Seminar, F. Vieweg & Sohn Verlagsgesellschaft mbH, Braunschweig, pp. 28-46, 1990

[BRA93] T. Bräunl, *Parallele Programmierung*, F. Vieweg & Sohn Verlagsgesellschaft mbH, Braunschweig-Wiesbaden, 1993

[BUR93] S. Burkhardt, *Parallele Rechnersysteme*, Verlag Technik GmbH, Berlin-München, 1993

[DAG86] G. Dagan, *Statistical Theory of Groundwater Flow and Transport: Pore to Laboratory, Laboratory to Formation, and Formation to Regional Scale*, Water Resources Research, Vol. 22, No. 9, pp. 120-134, 1986

[DIN49] DIN Deutsches Institut für Normung e.V., DIN 4049-1: Hydrogeologie – Grundbegriffe, 1992, *DIN 4049-2: Hydrogeologie – Begriffe der Grundwasserbeschaffenheit*, 1990, *DIN 4049-3: Hydrogeologie – Begriffe zur quantitativen Hydrogeologie*, 1994

[DON94] J.J. Dongarra, *Performance of Various Computers Using Standard Linear Equations Software*, Mathematical Sciences Section, Oak Ridge National Laboratory, CS-89-85, 1994

[DRY90] M. Dryja, O.B. Widlund, *Multilevel Additive Methods for Elliptic Finite Element Problems*, in Parallel Algorithms for Partial Differential Equations, W. Hackbusch, Ed., Proceedings of the Sixth GAMM-Seminar, F. Vieweg & Sohn Verlagsgesellschaft mbH, Braunschweig, pp. 58-69, 1990

[ESS93] R. Esser, R. Knecht, *Intel Paragon XP/S – Architecture and Software Environment*, Forschungszentrum Jülich, ZAM, No. IB-9305, 1993

[FET93] C.W. Fetter, *Contaminant Hydrogeology*, Macmillan Publishing Company, New York, 1993

[FLY66] M. Flynn, *Very High Speed Computing Systems*, Proceedings of the IEEE, Vol. 54, pp. 1901-1909, 1966

[FOX88] G. Fox, M.A. Johnson, G.A. Lyzenga, S.W. Otto, J.K. Salmon, D.W. Walker, *Solving Problems on Concurrent Processors*, Vol. 1, Prentice Hall International, London, 1988

[FRE79] R.A. Freeze, J.A. Cherry, *Groundwater*, Prentice Hall International, London, 1979

[GAT93] K.E. Gates, W.P. Petersen, *A Technical Description of Some Parallel Computers*, ETH Zürich, 1993

[GEL83] L.W. Gelhar, C.L. Axness, *Three-Dimensional Stochastic Analysis of Macrodispersion in Aquifers*, Water Resources Research, Vol. 19, No. 1, pp. 161-180, 1983

[GEL86] L.W. Gelhar, *Stochastic Subsurface Hydrology – From Theory to Applications*, Water Resources Research, Vol. 22, No. 9, pp. 135-145, 1986

[GER94] M. Gerndt, O. Neuendorf, J. Prümmer, H. Vereecken, *A Comparison of Two Parallelization Strategies for TRACE*, Forschungszentrum Jülich, ICG-4, Internal Report, No. 501994, 1994

[GOL93] G. Golub, J.M. Ortega, *Scientific Computing: An Introduction with Parallel Computing*, Academic Press, Inc., San Diego, 1993

[GUS88] J.L. Gustafson, *Reevaluating Amdahl's Law*, Communication of the ACM, Vol. 31, No. 5, 1988

[HOE89] B. Hölting, *Hydrogeologie*, F. Enke Verlag, Stuttgart, 1989

[HUY83] P.S. Huyakorn, G.F. Pinder, *Computational Methods in Subsurface Flow*, Academic Press, Inc., New York, 1983

[HUY86] P.S. Huyakorn, E.P. Springer, V. Guvanasen, T.D. Wadsworth, *A Three-Dimensional Finite-Element Model for Simulating Water Flow in Variably Saturated Porous Media*, Water Resources Research, Vol. 22, No. 13, pp. 1790-1808, 1986

[IBB89] R.N. Ibbett, N.P. Topham, *Architecture of High Performance Computers*, Vol. II, Macmillan Education Ltd., London, 1989

[IFF77] Intel Supercomputer Systems Division, *Paragon OSF/1 Fortran Compiler User's Guide*, No. 312491-001, 1993

[INT93] Intel Supercomputer Systems Division, *Paragon User's Guide*, No. 312489-002, 1993

[ISC93] Intel Supercomputer Systems Division, *Paragon Fortran System Calls – Reference Manual*, No. 312488-002, 1993

[JUN92] D. Jungmann, H. Stange, *Einführung in die Rechnerarchitektur*, C. Hanser Verlag, München, 1992

[KOE94] C.H. Koebel, D.B. Loveman, R.S. Schreiber, G.L. Steele, M.E. Zosel, *The High Performance Fortran Handbook*, Scientific and Engineering Computation Series, Massachusetts Institute of Technology, 1994

[KUH94] A. Kuhr, *Dokumentation zu INPGEN*, Forschungszentrum Jülich, ICG-4, 1994, nicht veröffentlicht

[LI86] K. Li, *Shared Virtual Memory System on Loosely Coupled Multiprocessors*, Ph.D. Thesis, Yale University, Technical Report YALEU-RR-492, 1986

[LIO88] P.L. Lions, *On the Schwarz Alternating Method I*, in Domain Decomposition Methods for Partial Differential Equations, R. Glowinski, G.H. Golub, G.A. Meurant, J. Périaux, Eds., SIAM, Philadelphia, 1988

[LIO89] P.L. Lions, *On the Schwarz Alternating Method II*, in Domain Decomposition Methods for Partial Differential Equations II, T. Chan, R. Glowinski, G.A. Meurant, J. Périaux, O. Widlund, Eds., SIAM, Philadelphia, 1989

[MAR89] D. Marsal, *Finite Differenzen und Elemente*, Springer Verlag, Berlin, 1989

[MAT83] G. Mattheß, K. Ubell, *Allgemeine Hydrogeologie, Grundwasserhaushalt*, Gebr. Borntraeger, Berlin, 1983

[MON93] B. Monien, R. Feldmann, R. Klasing, R. Lüling, *Parallel Architectures: Design and Efficient Use*, Department of Computer Science, University of Paderborn, Germany, 1993

[PAT89] P.C. Patton, *Performance Limits for Parallel Processors*, in Parallel Supercomputing: Methods, Algorithms and Applications, G.F. Carey, Ed., John Wiley & Sons, New York, pp. 1-16, 1989

[PIN77] G.F. Pinder, W.G. Gray, *Finite Element Simulation of Surface and Subsurface Hydrogeology*, Academic Press, Inc., San Diego, 1977

[RIC31] L.A. Richards, *Capillary Conduction of Liquids through Porous Mediums*, Physics 1, pp. 318-333, 1931

[ROD89] G. Rodrigue, S. Shah, *Pseudo-Boundary Conditions to Accelerate Parallel Schwarz Methods*, Parallel Supercomputing: Methods, Algorithms and Applications, G.F. Carey, Ed., John Wiley & Sons, New York, pp. 77-88, 1989

[SCH69] D. Schroeder, *Bodenkunde in Stichworten*, Verlag Ferdinand Hirt, 4. Aufl., Unterägeri, 1984

[SCH81] U. Schendel, *Einführung in die parallele Numerik*, R. Oldenbourg Verlag GmbH, München, 1981

[SCH84] F. Scheffer, P. Schachtschabel, *Lehrbuch der Bodenkunde*, Ferdinand Enke Verlag, Stuttgart, 1984

[SCH90] H.A. Schwarz, *Ueber einen Grenzübergang durch alternirendes Verfahren*, in Gesammelte mathematische Abhandlungen, Vol. 2, Springer Verlag, Berlin, S. 133-143, 1890

[SOB36] S. Sobolev, *The Schwarz Algorithm in the Theory of Elasticity*, Dokl. Acad. Nauk., USSR, pp. 272, 1936

[SPO86] G. Sposito, *The Physics of Soil Water Physics*, Water Resource Research, Vol. 22, No. 9, pp. 83-88, 1986

[STE93] A.J. van der Steen, *An Overview of (Almost) Available Parallel Systems*, Publication of the NCF, AC's-Gravenhage, Niederlande, 1993

[VER93] H. Vereecken, G. Lindenmayr, A. Kuhr, D.H. Welte, A. Basermann, *Numerical Modelling of Field Scale Transport in Heterogeneous Variably Saturated Porous Media*, Forschungszentrum Jülich, ZAM, IB Nr. 9301, 1993

[VER94] H. Vereecken, G. Lindenmayr, O. Neuendorf, U. Döring, R. Seidemann, *TRACE A Mathematical Model for Reactive Transport in 3D Variably Saturated Porous Media*, Forschungszentrum Jülich, ICG-4, Internal Report No. 501494, 1994

[VNL94] H. Vereecken, O. Neuendorf, G. Lindenmayr, A. Basermann, *A Parallel Schwarz Domain Decomposition Method for the Numerical Solution of Transient Water Flow in Heterogeneous Porous Media*, submitted to Parallel Computing, 1994

[VGE80] M.TH. Van Genuchten, *A Closed-Form Equation for Predicting the Hydraulic Conductivity of Unsaturated Soils*, Soil Science Society of America Journal, No. 44, pp. 892-898, 1980

[WID89] O.B. Widlund, *Domain Decomposition Algorithms and the Bicentennial of the French Revolution*, SIAM News, Vol. 22, No. 4, pp. 15-20, 1989

[WID92] O.B. Widlund, *Some Schwarz Methods for Symmetric and Nonsymmetric Elliptic Problems*, in Fifth Conference on Domain Decomposition Methods for Partial Differential Equations, T.F. Chan, D.E. Keyes, G.A. Meurant, J.S. Scroggs, R.G. Voigt, Eds., SIAM, Philadelphia, pp. 19-36, 1992

[YEH87] G.T. Yeh, *3DFEMWATER: A Three Dimensional Finite Element Model of Water Flow through Saturated-Unsaturated Media*, Oak Ridge National Laboratory, Publication No. 2904, 1987

Anhang A

WATDAT-Datei (sequentielle Version)

```
    1  SIMULATION OF ONE-D COLUMN INFILTRATION-EVAPORATION; L=CM, T=DAY, M=G 101    0
C ******* DATA SET 2:  BASIC INTEGERS
  1190  528   1    0    1    6    1    0    0  800   20    3  300
C ******* DATA SET 3:  BASIC REAL PARAMETERS
  0.05D0     0.20D0     0.3D0    5.0D-2    5.0D-2     1.0D0  7.316D12
  1.1232D2    -1.0D0     0.8D0     0.8D0    1.0D0
                    :  TIME VARIABLES
- START OF CALCULATIONS - - - - - - - - - -  DAY  - - - - -<<    1    >>N
                                             MONTH  - - - -<<    1    >>N
                                             YEAR - - - - -<<  1990   >>N
- END OF CALCULATIONS - - - - - - - - - - -  DAY  - - - - -<<   10    >>N
                                             MONTH  - - - -<<   1     >>N
                                             YEAR - - - - -<<   1990  >>N
C ******* DATA SET 4:  PRINTER, STORAGE CONTROL AND TIME STEP SIZE RESETTING
DIAGNOSTIC FLOW INFO Y(1),N(0) :   1
PRESSURE HEADS       Y(1),N(0) :   0
TOTAL HEADS          Y(1),N(0) :   0
WATER CONTENTS       Y(1),N(0) :   0
VELOCITIES           Y(1),N(0) :   0
    1.0D01  2.0000D1    1.0D38
C ******* DATA SET 5:  MATERIAL PROPERTIES
     1.0D0   0.0D0     1.0D0     0.0D0     0.0D0     0.0D0
C ******* DATA SET 6:  SOIL PROPERTY PARAMETERS
C ******* DATA SET 7:  NODE COORDINATES
          1      12      12       0       0       0     500       0       0
          2      12      12       0       0     500     500       0       0
          3      12      12       0       0    1000     500       0       0
 .....
       1187       0       0   18000     500     260       0       0       0
       1188       0       0   18000     500     340       0       0       0
       1189       0       0   18000     500     390       0       0       0
       1190       0       0   18000     500     450       0       0       0
        0         0       0       0       0       0       0        0      0
C ******* DATA SET 8:  SUBREGIONAL DATA
    2
    1    1    1  595    0
    0    0    0    0    0                       END OF NNPLR(K)
    1  594    1    1    1
    0    0    0    0    0                       END OF GNLR(I,1)
    1  594    1  596    1
    0    0    0    0    0                       END OF GNLR(I,2)
C ******* DATA SET 9:  ELEMENT INCIDENCES
        1    12    11     1    13   608   596     2    14   609   597    12
        2    12    11     2    14   609   597     3    15   610   598    12
        3    12    11     3    15   610   598     4    16   611   599    12
 .....
```

```
      184    30    11   206   218   813   801   207   219   814   802    12
      185    30    11   207   219   814   802   208   220   815   803    12
      516     6     1   569   581  1176  1164   570   582  1177  1165     1
      523     5     1   581   589  1184  1176   582   590  1185  1177     1
       0      0     0     0     0     0     0     0     0     0     0     0
C ******* DATA SET 11:  INITIAL CONDITIONS
        1    13    12 2200 0 0
        2    13    12 1700 0 0
        3    13    12 1200 0 0
 .....
      800    31    12 -550 0 0
      801    31    12 -700 0 0
      802    31    12 -750 0 0
      803    31    12 -800 0 0
        0        0       0       0       0       0
C ******* DATA SET 12:  SOURCE/SINK AND B. C. CONTROL INTEGERS
    0    0    0    0    0    0    0    0   14    7     2    0
   43   90    1    4    0   68  134   1    2    0    6   14    1    2    0
    0    0    0    0    0
C ******* DATA SET 14:  VARIABLE BOUNDARY CONDITIONS
    0.0D0     1.0D00    1.0D0     1.0D00  1.001D0     0.0D0    1.0D38     0.0D0
    1   42    1    1    0
    0    0    0    0    0         END OF IRTYP
    1   11   1    12   24   619  607   12   12   12   12
   13   30   1   208  220   815  803   12   12   12   12
    0    0    0    0    0    0    0    0    0    0    0 END OF ISV(J,I) J=1,4
    1   12    1   12   12
    14  12    1  607   12
    27  31    1  208   12
    59  31    1  803   12
    0    0    0    0    0                       END OF NPVB
    1   12    1          0.0D0     0.0D0       0.0
    14  12    1          0.0       0.0         0.0
    27  31    1          0.0       0.0         0.0
    59  31    1          0.0       0.0         0.0
    0    0    0          0.0       0.0         0.0             END OF HCON
    1   12    1        -90.0D2     0.0D0       0.0
    14  12    1        -90.0d2     0.0         0.0
    27  31    1        -90.0d2     0.0         0.0
    59  31    1        -90.0d2     0.0         0.0
    0    0    0          0.0       0.0         0.0             END OF HMIN
C ******* DATA SET 15:  DIRICHLET BOUNDARY CONDITIONS
     0.0D0     0.0D0    1.0D38    00.0D0
     0.0      50.0d0    1.0d38    50.0d0
     0.0     100.0d0    1.0d38   100.0d0
     0.0     150.0d0    1.0d38   150.0d0
     0.0     160.0d0    1.0d38   160.0d0
     0.0     200.0d0    1.0d38   200.0d0
     0.0     200.0d0    1.0d38   200.0d0
579 578 577 576 588 587 595 1174 1173 1172 1171 1183 1182 1190
    1    6    1    1    1
    8    6    1    1    1
    0    0    0    0    0                     END OF IDTYP
c ******* DATA SET 16:  CAUCHY BOUNDARY CONDITIONS
     0.0d0 0.0 1.0d38 0.0
    1   67     1    1    0
    0   0      0    0    0                       END OF ICTYP
    1   12    1     1    13   608  596  12  12  12  12
    14  4     1     157  165  760  752  8   8   8   8
    19  31    1     197  209  804  792  12  12  12  12
    51   0    0     581  589  1184 1176 0   0   0   0
    52  10    1     1    596  597  2    1   1   1   1
    63  5     1     589  1184 1185 590  1   1   1   1
    0   0     0     0    0    0    0    0   0    0  0  END OF ISC(J,I) J=1,4
    1 13 25  37 49 61 73 85
    97 109 121 133 145 157 165 173
    181 189 197 209 221 233 245 257
    269 281 293 305 317 329 341 353
```

```
    365 377 389 401 413 425 437 449
    461 473 485 497 509 521 533 545
    557 569 581 589 590 591 592 593
    608 620 632 644 656 668 680
    692 704 716 728 740 752 760 768
    776 784 792 804 816 828 840 852
    864 876 888 900 912 924 936 948
    960 972 984 996 1008 1020 1032 1044
    1056 1068 1080 1092 1104 1116 1128 1140
    1152 1164 1176 1184 594 1185 1186
    1187 1188 1189  2 3 4 5 6 7 8 9
    10 11 596 597 598 599 600 601 602 603
    604 605 606
C ******* DATA SET 17:  NEUMANN BOUNDARY CONDITIONS
     0.0D0   -35.0D00   1.0D38   -35.0D0
    1    5    1    1    0
    0    0    0    0    0         END OF INTYP
    1    0    0   152 164   759  747   0    0    0    0
    2    4    1   164 172   767  759   8    8    8    8
    0    0    0    0    0    0    0    0    0    0    0 END OF ISN(J,I) J=1,4
   152 164 747 759 172 180 767 775 188 196
   783 791 204 799
    0    0    0    0    0                      END OF NPNB
```

Anhang B

Die INCLUDE-Datei

```
C Anfang INCLUDE - Datei

C XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
C 345678901234567890123456789012345678901234567890123456789012345678901234567890123
C XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

C Elements
      INTEGER          MAXEL
      PARAMETER        (MAXEL  = 6375)
C Nodal Points
      INTEGER          MAXNP
      PARAMETER        (MAXNP  = 9216)
C Boundary Element Sides
      INTEGER          MAXBES
      PARAMETER        (MAXBES = 5150)
C Boundary Nodal Points
      INTEGER          MAXBNP
      PARAMETER        (MAXBNP = 6216)
C Bandwidth CMATRX
      INTEGER          JBAND
      PARAMETER        (JBAND  =   27)
C Time Steps
      INTEGER          MAXNTI
      PARAMETER        (MAXNTI =  100)
C Delt Changes
      INTEGER          MXNDTC
      PARAMETER        (MXNDTC =   51)
C Subregion Nodal Points
      INTEGER          LTMXNP
      PARAMETER        (LTMXNP = 4608)
C ???
      INTEGER          LMXNP
      PARAMETER        (LMXNP  = 1536)
C Subregion Bandwidth
      INTEGER          LMXBW
      PARAMETER        (LMXBW  =   15)
C Subregions
      INTEGER          MXREGN
      PARAMETER        (MXREGN =  256)
C Source Elements (C)
      INTEGER          MXSELC
      PARAMETER        (MXSELC =    5)
C Source Profiles (C)
      INTEGER          MXSPRC
      PARAMETER        (MXSPRC =    2)
C Data Points / Well-Source-Sink Profile (C)
      INTEGER          MXSDPC
      PARAMETER        (MXSDPC =    4)
C Well Nodal Points (C)
```

```
      INTEGER          MXWNPC
      PARAMETER        (MXWNPC =    7)
C Well-Source-Sink Profiles (C)
      INTEGER          MXWPRC
      PARAMETER        (MXWPRC =    3)
C Data Points / Well-Source-Sink Profiles (C)
      INTEGER          MXWDPC
      PARAMETER        (MXWDPC =    4)
C Cauchy Nodal Points (C)
      INTEGER          MXCNPC
      PARAMETER        (MXCNPC =  200)
C Cauchy Elements Surfaces (C)
      INTEGER          MXCESC
      PARAMETER        (MXCESC =  200)
C Cauchy Flux Profiles (C)
      INTEGER          MXCPRC
      PARAMETER        (MXCPRC =    4)
C Data Points / Cauchy Flux Profile (C)
      INTEGER          MXCDPC
      PARAMETER        (MXCDPC =    8)
C Neumann Nodal Points (C)
      INTEGER          MXNNPC
      PARAMETER        (MXNNPC =   36)
C Neumann Element Surfaces (C)
      INTEGER          MXNESC
      PARAMETER        (MXNESC =   25)
C Neumann Flux Profiles (C)
      INTEGER          MXNPRC
      PARAMETER        (MXNPRC =    4)
C Data Points / Neumann Flux Profiles (C)
      INTEGER          MXNDPC
      PARAMETER        (MXNDPC =    8)
C Variable Element Surfaces (C)
      INTEGER          MXVESC
      PARAMETER        (MXVESC =    8)
C Variable Nodal Points (C)
      INTEGER          MXVNPC
      PARAMETER        (MXVNPC =    4)
C Rainfall Profiles (C)
      INTEGER          MXRPRC
      PARAMETER        (MXRPRC =    4)
C Data Points / Rainfall Profile (C)
      INTEGER          MXRDPC
      PARAMETER        (MXRDPC =    8)
C Dirichlet Nodal Points (C)
      INTEGER          MXDNPC
      PARAMETER        (MXDNPC =   72)
C Dirichlet Total Head Profile (C)
      INTEGER          MXDPRC
      PARAMETER        (MXDPRC =    2)
C Data Point / Dirichlet Total Head Profile (C)
      INTEGER          MXDDPC
      PARAMETER        (MXDDPC =    4)
C Source Elements
      INTEGER          MXSEL
      PARAMETER        (MXSEL  =    1)
C Source Profiles
      INTEGER          MXSPR
      PARAMETER        (MXSPR  =    1)
C Data Points / Well-Source-Sink Profile
      INTEGER          MXSDP
      PARAMETER        (MXSDP  =    1)
C Well Nodal Points
      INTEGER          MXWNP
      PARAMETER        (MXWNP  =    1)
C Well-Source-Sink Profiles
      INTEGER          MXWPR
      PARAMETER        (MXWPR  =    1)
```

```
C Data Points / Well-Source-Sink Profile
      INTEGER          MXWDP
      PARAMETER        (MXWDP  =    1)
C Cauchy Nodal Points
      INTEGER          MXCNP
      PARAMETER        (MXCNP  =  200)
C Cauchy Element Surfaces
      INTEGER          MXCES
      PARAMETER        (MXCES  =  200)
C Cauchy Flux Profiles
      INTEGER          MXCPR
      PARAMETER        (MXCPR  =    2)
C Data Points / Cauchy Flux Profile
      INTEGER          MXCDP
      PARAMETER        (MXCDP  =    2)
C Neumann Nodal Points
      INTEGER          MXNNP
      PARAMETER        (MXNNP  =  200)
C Neumann Element Surfaces
      INTEGER          MXNES
      PARAMETER        (MXNES  =  200)
C Neumann Flux Profiles
      INTEGER          MXNPR
      PARAMETER        (MXNPR  =    1)
C Data Points / Neumann Flux Profiles
      INTEGER          MXNDP
      PARAMETER        (MXNDP  =    2)
C Variable Element Surfaces
      INTEGER          MXVES
      PARAMETER        (MXVES  =  200)
C Variable Nodal Points
      INTEGER          MXVNP
      PARAMETER        (MXVNP  =  200)
C Rainfall Profile
      INTEGER          MXRPR
      PARAMETER        (MXRPR  =    1)
C Data Points / Rainfall Profile
      INTEGER          MXRDP
      PARAMETER        (MXRDP  =  600)
C Dirichlet Nodal Points
      INTEGER          MXDNP
      PARAMETER        (MXDNP  = 9216)
C Dirichlet Total Head Profiles
      INTEGER          MXDPR
      PARAMETER        (MXDPR  =    7)
C Data Points / Dirichlet Total Head Profile
      INTEGER          MXDDP
      PARAMETER        (MXDDP  =    2)
C Soil Parameters / Material
      INTEGER          MXSPPM
      PARAMETER        (MXSPPM =    8)
C ???
      INTEGER          MNCOL
      PARAMETER        (MNCOL  =  435)
C ???
      INTEGER          MNPLS
      PARAMETER        (MNPLS  =    1)
C ???
      INTEGER          MNIRDP
      PARAMETER        (MNIRDP =    5)
C ???
      INTEGER          MNLAID
      PARAMETER        (MNLAID =    5)
C ???
      INTEGER          MIGR
      PARAMETER        (MIGR   =    1)

C Dateivariablen fuer INPUT und OUTPUT
```

```
      INTEGER          LI1
      PARAMETER        (LI1    =    1)
      INTEGER          LI2
      PARAMETER        (LI2    =    2)
      INTEGER          LI3
      PARAMETER        (LI3    =   39)
      INTEGER          LI30
      PARAMETER        (LI30   =   30)
      INTEGER          LI60
      PARAMETER        (LI60   =   60)
      INTEGER          LI99
      PARAMETER        (LI99   =   99)

      INTEGER          OutPutLO1
      PARAMETER        (OutPutLO1 = 1)

      INTEGER          LO1
      PARAMETER        (LO1    =    8)
      INTEGER          LO2
      PARAMETER        (LO2    =    9)
      INTEGER          LO3
      PARAMETER        (LO3    =   33)
      INTEGER          LO4
      PARAMETER        (LO4    =   33)
      INTEGER          LO5
      PARAMETER        (LO5    =   33)
      INTEGER          LO6
      PARAMETER        (LO6    =   13)
      INTEGER          LO7
      PARAMETER        (LO7    =   14)
      INTEGER          LO8
      PARAMETER        (LO8    =   15)
      INTEGER          LO9
      PARAMETER        (LO9    =   15)
      INTEGER          LO10
      PARAMETER        (LO10   =   19)
      INTEGER          LO11
      PARAMETER        (LO11   =   18)
      INTEGER          LO12
      PARAMETER        (LO12   =   17)
      INTEGER          LO77
      PARAMETER        (LO77   =   76)

C Dateinamen fur INPUT und OUTPUT-Dateien
      CHARACTER*30     FLI1
      PARAMETER        (FLI1   = 'data/in/WATDAT.hom')
      CHARACTER*30     FLI2
      PARAMETER        (FLI2   = 'data/in/SOLDAT.hom')
      CHARACTER*30     FLI3
      PARAMETER        (FLI3   = 'data/in/CROPDAT')
      CHARACTER*30     FLI30
      PARAMETER        (FLI30  = 'data/in/CLIMAT')
      CHARACTER*30     FLI60
      PARAMETER        (FLI60  = 'data/in/HYST.DAT')
      CHARACTER*30     FLI99
      PARAMETER        (FLI99  = 'data/in/CONTROL.DAT')

      CHARACTER*30     FLO1
C     PARAMETER        (FLO1   = 'data/out/OUTPUT.###')
      PARAMETER        (FLO1   = '/usr/tmp/rwOUT/OUTPUT.###')
      CHARACTER*30     FLO2
      PARAMETER        (FLO2   = 'data/out/')
      CHARACTER*30     FLO3
      PARAMETER        (FLO3   = 'data/out/OUT3-5')
C     PARAMETER        (FLO3   = 'data/out/PRHEAD')
      CHARACTER*30     FLO4
      PARAMETER        (FLO4   = 'data/out/THEADS')
      CHARACTER*30     FLO5
```

```
      PARAMETER       (FLO5   = 'data/out/WATCON')
      CHARACTER*30    FLO6
      PARAMETER       (FLO6   = 'data/out/DVELOS')
      CHARACTER*30    FLO7
C     PARAMETER       (FLO7   = 'data/out/SOLOUT.###')
      PARAMETER       (FLO7   = '/usr/tmp/rwOUT/SOLOUT.###')
      CHARACTER*30    FLO8
      PARAMETER       (FLO8   = 'data/out/SOLCON')
      CHARACTER*30    FLO9
      PARAMETER       (FLO9   = 'data/out/SOLFLO')
      CHARACTER*30    FLO10
      PARAMETER       (FLO10  = 'data/out/wp')
      CHARACTER*30    FLO11
      PARAMETER       (FLO11  = 'data/out/cp')
      CHARACTER*30    FLO12
      PARAMETER       (FLO12  = 'data/out/test')
      CHARACTER*30    FLO77
C     PARAMETER       (FLO77  = 'data/out/testhom.###')
      PARAMETER       (FLO77  = '/usr/tmp/rwOUT/testhom.###')

C Ende INCLUDE - Datei
```

Anhang C

Die CONTROL-Datei

```
MAXIT: Maximale Anzahl der Iterationen im CG
5
MaxZeitschritte: Maximale Anzahl der Zeitschritte
14
Overlap: Zahl der Elemente, die im Ueberlapp liegen
3
NITER: Number of Picard-Iterations
20
NCYL: Number of Boundary Iterations
20
NPITER: Number of Linear Iterations
200
TOLA:
5.0D-2
TOLB: Residuum Picard-Schleife
1.0D-3
VORKON:
1
TOLREL: Residuum 1 CG-Schleife
1.0D-5
RESEPS: Residuum 2 CG-Schleife
1.0D-10
MAXPLOOP: Maximale Anzahl der Schwarz-Schleifen
100
EPSPARA: Residuum Schwarz-Schleife
1.0D-5
Output in OUTPUT.###
0
OUTPUT in SOLOUT.###
0
OUTPUT in rwOUT.###
0
OUTPUT in distOUT.###
0
OUTPUT in commOUT.###
0
```

Anhang D

WATDAT-Datei (parallele Version)

```
> Testsimulation Programm TRACE
   SIMULATION OF 3D REGION  [Laenge]=cm, [Zeit]=DAY, [Masse]=g
  0       IBUG
  1       ICHNG
  1       WatDatTyp
>****** DATA SET 2:  BASIC INTEGERS
  256000  Number Nodes
  80      Number Nodes in x-Direction
  80      Number Nodes in y-Direction
  40      Number Nodes in z-Direction
  243399  Number Elements
  0       Type of Material 0: homogen / 1: heterogen
  1       KSS: Boundary Conditions 0: stationary / 1: instationary
  6       ?NMPRM
  1       Gravitation 0: No / 1: Yes
  0       Lumping 0: No / 1: Yes
  0       ?IMID
  3       ?NDTCHG
>****** DATA SET 3:  BASIC REAL PARAMETERS
   0.005D0 DELT
   0.20D0  CHNG
   1.0D0   DELMAX
   1.0D0   W
   1.0D0   OME
   0.5D0   OMI
   1.0D0   DAYINC
                :  TIME VARIABLES
- START OF CALCULATIONS - - - - - - - - -  DAY  - - - - -<<    19   >>N
                                           MONTH  - - - -<<    01   >>N
                                           YEAR - - - - -<<   1980  >>N
- END OF CALCULATIONS - - - - - - - - - -  DAY  - - - - -<<    01   >>N
                                           MONTH  - - - -<<    03   >>N
                                           YEAR - - - - -<<   1994  >>N
>****** DATA SET 4:  PRINTER, STORAGE CONTROL AND TIME STEP SIZE RESETTINGS
DIAGNOSTIC FLOW INFO Y(1),N(0) :   1
PRESSURE HEADS       Y(1),N(0) :   0
TOTAL HEADS          Y(1),N(0) :   0
WATER CONTENTS       Y(1),N(0) :   0
VELOCITIES           Y(1),N(0) :   0
   0.25D0   0.5D0   1.0D38
>****** DATA SET 5:  MATERIAL PROPERTIES
 1.0D0   1.0D0   1.0D0   0.0D0   0.0D0   0.0D0    0.0D0
>****** DATA SET 7:  NODE COORDINATES
       1  79  1  0.     0.          0.     5.00000  0.  0.
      81  79  1  0.     5.00000     0.     5.00000  0.  0.
     161  79  1  0.    10.00000     0.     5.00000  0.  0.
```

```
.....
  255841  79  1  0.   390.000    195.000   5.00000  0.  0.
  255921  79  1  0.   395.000    195.000   5.00000  0.  0.
       0   0  0  0.     0.         0.      0.       0.  0.
>****** DATA SET 11: INITIAL CONDITIONS
       1  6399  1    210.000  0.D0  0.D0
    6401  6399  1    205.000  0.D0  0.D0
.....
  243201  6399  1    20.0000  0.D0  0.D0
  249601  6399  1    15.0000  0.D0  0.D0
       0     0  0     0.0     0.    0.
>****** DATA SET 12: SOURCE/SINK AND B.C. CONTROL INTEGERS
 0  0  0   0   0   0   0   0   24808    1    2    0
 0  0  0   2   0   0   0   0       2    0    0    0    0    2    0
 0  0  0   0   0
>****** DATA SET 15: DIRICHLET BOUNDARY CONDITIONS         XX  97
 0.0D0  220.0D0 1.0D38  220.0D0
       1    6399       1  249601       1       1       0
    6401    6399       1       1       1       1       0
   12801      79       1    6401       1       1       0
.....
   24801      77       1  236960      80       1       0
   24881      77       1  243360      80       1       0
   0   0   0   0   0   0   0
```

Anhang E

Tabellarische Übersicht der Ergebnisse

Zeit [s]	mittleres Gebiet FE-Knoten Nr. 14895	großes Gebiet FE-Knoten Nr. 131240
0.07	0.001	0.000
0.22	0.022	0.000
0.41	0.180	0.007
0.70	0.768	0.054
1.14	2.108	0.237
1.80	4.162	0.716
2.78	6.408	1.649
4.26	8.218	3.096
6.47	9.306	4.936
9.79	9.791	6.835
14.77	9.949	8.381
22.25	9.985	9.339
33.46	9.991	9.779
50.27	9.991	9.926
75.50	9.991	9.959

Tabelle E.1: Änderung des hydrostatischen Potentials (in cm) bei der Simulation SIM1 für das mittlere (31x31x31 FE-Knoten) und das große (80x80x40 FE-Knoten) Gebiet (Abbildung 7.1)

CPUs	Laufzeit Gesamt	Speedup Gesamt	Laufzeit Zeitschleife	Speedup Zeitschleife
1	834.0	1.00	634.2	1.00
2	362.9	2.29	300.6	2.11
4	187.1	4.46	159.4	3.97
8	96.5	8.64	77.4	8.19
16	66.5	12.54	42.4	14.96
32	70.3	11.86	25.4	24.97

Tabelle E.2: Laufzeiten (in s) und Speedups für die Berechnung des kleinen Gebietes von SIM1 (6x6x256 FE-Knoten, 3 Elemente Überlapp, 15 Zeitschritte), Aufteilung der Prozessoren jeweils 1x1xCPUs (Abbildung 7.2)

CPUs	Laufzeit Gesamt	Laufzeit Zeitschleife	CPUs x	y	z
4	1245.7	965.9	1	2	2
8	685.0	567.0	2	2	2
16	419.8	345.2	2	2	4
27	289.7	215.6	3	3	3
32	295.1	212.2	2	4	4
64	258.8	135.8	4	4	4

Tabelle E.3: Laufzeiten (in s) und Gebietszerlegung für die Berechnung des mittleren Gebietes von SIM1 (31x31x31 FE-Knoten, 3 Elemente Überlapp, 15 Zeitschritte) (Abbildung 7.3)

Zeit-	Überlapp [Elemente]				
schritt	1	2	3	4	5
1	18	13	11	10	10
2	23	16	13	12	11
3	30	19	16	14	13
4	38	24	19	17	16
5	48	30	24	21	20
6	59	36	28	25	24
7	70	42	32	29	27
8	77	46	35	31	30
9	78	47	36	33	31
10	71	43	34	31	29
11	55	36	27	26	26
12	31	21	18	16	15
13	11	8	7	8	8
14	2	2	2	2	2
15	2	2	2	2	2

Tabelle E.4: Zahl der Durchläufe der Schwarz-Schleife bei der Berechnung des mittleren Gebietes für SIM1 mit 64 Prozessoren (31x31x31 FE-Knoten, 15 Zeitschritte), Aufteilung der CPUs: 4x4x4 (Abbildung 7.5)

Überlapp	1	2	3	4	5
Laufzeit	256.0	240.5	277.7	322.2	350.6

Tabelle E.5: Laufzeiten (in s) des Gesamtprogramms bei variierendem Überlapp für das mittlere Gebiet bei SIM1 bei Berechnung mit 64 Prozessoren (31x31x31 FE-Knoten, 15 Zeitschritte), Aufteilung der CPUs: 4x4x4 (Abbildung 7.4)

CPUs	Laufzeit	Laufzeit	K-Funktion	CPUs		
	Gesamt	Zeitschleife		x	y	z
32	2159.8	1397.7	1.45	4	4	2
64	1784.1	864.2	1.71	4	8	2
96	1807.7	655.7	1.86	6	8	2
108	1898.5	585.4	1.88	6	6	3
128	2067.2	592.4	2.00	8	8	2
136	2316.2	651.5	2.28	4	17	2

Tabelle E.6: Laufzeiten (in s) des Gesamtprogramms und der Zeitschleife, sowie die Zerlegung des Gebietes und die zugehörigen Werte der K-Funktion für die Berechnung des großen Gebietes von SIM1 (80x80x40 FE-Knoten, 3 Elemente Überlapp, 15 Zeitschritte) (Abbildungen 7.6 und 7.7)

Zeit-	CPUs					
schritt	32	64	96	108	128	136
1	10	10	10	10	10	10
2	10	11	11	11	11	12
3	11	12	13	13	13	14
4	13	14	15	15	15	16
5	16	17	18	18	19	21
6	19	21	23	23	23	25
7	23	27	29	29	30	33
8	30	33	35	37	37	42
9	35	41	43	44	47	52
10	41	47	50	51	53	60
11	45	51	58	57	61	66
12	46	51	56	58	61	67
13	40	46	51	53	56	58
14	31	34	41	39	40	45
15	16	18	25	21	25	23

Tabelle E.7: Zahl der Durchläufe der Schwarz-Schleife bei der Berechnung des großen Gebietes für SIM1 mit unterschiedlicher Prozessorzahl (80x80x40 FE-Knoten, 15 Zeitschritte), Aufteilung entsprechend Tabelle E.6 (Abbildung 7.8)

CPUs	Laufzeit Gesamt	Speedup Gesamt	Laufzeit Zeitschleife	Speedup Zeitschleife
1	2295.8	1.00	2094.8	1.00
2	1177.8	1.95	1107.8	1.89
3	820.2	2.80	784.9	2.67
4	593.3	3.87	569.3	3.68
5	481.7	4.77	458.5	4.57
6	418.4	5.49	391.0	5.36
7	358.6	6.40	341.8	6.13
8	317.6	7.23	301.6	6.95
9	291.6	7.87	275.1	7.61
10	268.3	8.56	252.7	8.29
16	193.4	11.87	175.1	11.96
24	159.3	14.41	133.7	15.67
32	156.3	14.69	119.9	17.47
40	143.3	16.02	104.1	20.12
48	153.1	15.00	93.7	22.36
56	151.2	15.18	86.5	24.22
64	159.7	14.38	86.9	24.11

Tabelle E.8: Laufzeiten (in s) und Speedups des Gesamtprogramms und der Zeitschleife für die Berechnung von SIM2 (6x6x256 FE-Knoten, 3 Elemente Überlapp, 14 Zeitschritte), Aufteilung der Prozessoren jeweils 1x1xCPUs (Abbildung 7.9)

	Überlapp									
Laufzeiten	1	2	3	4	5	6	7	8	9	10
Zeitschleife	310.2	298.8	301.3	308.5	315.7	326.6	333.1	337.8	340.4	348.9
ASSEMBL	114.0	117.5	121.1	124.6	128.1	131.8	135.4	135.4	138.7	142.3
SPROP	77.9	80.4	82.8	85.2	87.6	90.1	92.6	92.6	94.9	97.4
CG	63.4	48.6	45.2	45.2	45.7	48.3	49.1	48.8	48.7	50.0

Tabelle E.9: Laufzeiten (in s) der Zeitschleife und rechenintensiver Routinen bei variierendem Überlapp für SIM2 bei Berechnung mit 8 Prozessoren (6x6x256 FE-Knoten, 14 Zeitschritte), Aufteilung der CPUs: 1x1x8 (Abbildung 7.11)

CPUs	Laufzeit CG	Speedup CG	Laufzeit SPROP	Speedup SPROP	Laufzeit ASSEMBL	Speedup ASSEMBL
1	228.4	1.00	619.9	1.00	906.9	1.00
2	154.8	1.48	313.6	1.98	459.1	1.98
3	104.7	2.18	211.6	2.93	309.4	2.93
4	80.0	2.86	160.4	3.86	235.9	3.84
5	64.5	3.54	129.0	4.81	188.5	4.81
6	56.0	4.08	109.5	5.66	160.3	5.66
7	50.4	4.53	95.0	6.53	138.9	6.53
8	45.1	5.06	83.2	7.45	120.9	7.50
9	42.4	5.39	75.5	8.21	110.3	8.22
10	39.9	5.72	68.2	9.09	99.6	9.11
16	31.5	7.25	43.8	14.15	63.9	14.19
24	26.2	8.72	31.7	19.56	46.2	19.63
32	23.5	9.72	24.5	25.30	35.6	25.47
40	22.5	10.15	21.9	28.31	32.0	28.34
48	20.3	11.25	19.5	31.79	28.4	31.93
56	19.9	11.48	17.1	36.25	24.9	36.42
64	20.5	11.14	14.6	42.46	21.4	42.38

Tabelle E.10: Laufzeiten (in s) und Speedups rechenintensiver Routinen für die Berechnung von SIM2 (6x6x256 FE-Knoten, 3 Elemente Überlapp, 14 Zeitschritte), Aufteilung der Prozessoren jeweils 1x1xCPUs (Abbildung 7.10)

Danksagung

Die vorliegende Arbeit wurde als Diplomarbeit im Fach Elektrotechnik am Lehrstuhl für Technische Informatik und Computerwissenschaften der Rheinisch-Westfälischen Technischen Hochschule (RWTH) Aachen erstellt.

Dem Lehrstuhlinhaber Herrn Prof. Dr. F. Hoßfeld danke ich für die Möglichkeit, die Arbeit am Zentralinstitut für Angewandte Mathematik (ZAM) des Forschungszentrums Jülich (KFA) anfertigen zu können.

Mein besonderer Dank gilt Herrn Dr. M. Weber und Herrn Dr. W.E. Nagel für die Betreuung der Arbeit und viele konstruktive Anregungen.

Für die sehr angenehme Arbeitsatmosphäre und die ständige Hilfsbereitschaft danke ich Herrn Dr. H. Vereecken und allen seinen MitarbeiterInnen im ICG-4.

An dieser Stelle danke ich auch meinen Eltern für ihre Unterstützung während meiner Studienzeit.