Fakultät für Informatik
der Technischen Universität München

Ein iteratives Lösungsverfahren bei der Finite-Element-Methode unter Verwendung von rekursiver Substrukturierung und hierarchischen Basen

Reiner Hüttl

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender:		Univ.-Prof. Dr. Jürgen Eickel
Prüfer der Dissertation:		
	1.	Univ.-Prof. Dr. Christoph Zenger
	2.	Univ.-Prof. Dr. Walter Wunderlich

Die Dissertation wurde am 2. November 1995 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 6. März 1996 angenommen.

Ein iteratives Lösungsverfahren bei der Finite-Element-Methode unter Verwendung von rekursiver Substrukturierung und hierarchischen Basen

Zusammenfassung

Das Verfahren der rekursiven Substrukturierung, das eine effiziente, parallelisierbare Variante der Finite-Element-Methode darstellt, wird mit hierarchischen Ansatzfunktionen kombiniert und in ein iteratives Lösungsverfahren weiterentwickelt. Die Anwendung des Prinzips der rekursiven Substrukturierung ermöglicht eine verteilte Modellierung und Berechnung von technischen Objekten. Sie unterstützt eine adaptive Zerlegung von Bauteilen und die modulare Ankopplung von verschiedenen FE-Programmen. Durch die Entwicklung eines iterativen Lösungsalgorithmus bleiben die Matrizen der Gleichungssysteme dünnbesiedelt und man erhält eine Reduktion des Rechen- und Speicheraufwands. Insbesondere an der Wurzel des Substrukturierungsbaums werden die Gleichungssysteme deutlich kleiner, was die Skalierbarkeit einer parallelen numerischen Simulation entscheidend verbessert. Die Verwendung von hierarchischen Ansatzfunktionen sorgt für eine gute Konvergenz der Gleichungslösung. Die Kombination von rekursiver Gebietszerlegung und hierarchischen Basen ermöglicht die Anwendung der Methode der adaptiven Relaxation. Damit erfolgt eine adaptive Steuerung der Iteration auf Gitterpunkte, die einen gewissen Mindestbetrag zur Approximation beitragen. Zudem ergibt sich dadurch ein Kriterium zur adaptiven Verfeinerung des FE-Netzes.

An dieser Stelle möchte ich mich bei allen Kollegen und Kolleginnen am Lehrstuhl für Informatik von Prof. Zenger und am Lehrstuhl für Baustatik für die stets gute Zusammenarbeit bedanken. Besonders hervorheben möchte ich dabei Martin Rudolph und Michael Schneider mit denen mich eine sehr enge projektbezogene Zusammenarbeit verbunden hat. Weiterhin bedanke ich mich bei Stefan Mayer und Alexander Pfaffinger für die gründliche und kritische Durchsicht des Manuskripts dieser Arbeit.

Außerdem bedanke ich mich bei meiner Lebensabschnittsgefährtin Julia Schwarz, die mir die notwendige moralische Unterstützung gegeben hat und schließlich die Arbeit in orthographischer Hinsicht überarbeitete. Dies versuchten auch mein Vater, Ute Schrüfer und Harry Artmaier.

Bei Prof. Dr. Walter Wunderlich bedanke ich mich für die Übernahme des zweiten Gutachtens und der Finanzierung von weiten Phasen meiner Forschungsarbeiten an der TU-München. Mein besonderer Dank gilt meinem Doktorvater Prof. Dr. Christoph Zenger ohne dessen zahlreichen Anregungen, fruchtbaren Diskusionen und unerschöpflichen Ideenreichtum diese Arbeit wohl nie zustande gekommen wäre.

Inhaltsverzeichnis

1 Einleitung

Komplexe Ingenieuraufgaben werden im Team bearbeitet. Analog geschieht die Berechnung solcher Aufgaben verteilt auf mehreren Workstations und Supercomputern [73]. Die Software der Zukunft muß deshalb die Verteilung von Berechnungen auf heterogenen Netzwerken gewährleisten. Diese Forderung hat Auswirkungen auf die Entwicklung von Software, die eine Einbindung kommerzieller Programme und die Ausnutzung von spezialisierten Rechnern ermöglichen sollte sowie eine genaue Festlegung von Schnittstellen erfordert.

Viele Vorgänge in Naturwissenschaft und Technik werden durch partielle Differentialgleichungen modelliert. Zur analytischen Lösung dieser Gleichungen ist die Finite-Element-Methode (FEM) ein weit verbreitetes Verfahren. Aufgrund ihrer hohen Robustheit und Flexibilität in der numerischen Simulation von komplexen technischen Objekten findet sie speziell im Bereich der Ingenieurwissenschaften starke Anwendung.

Die wachsende Komplexität heutiger Probleme macht die Verteilung einer numerischen Simulation basierend auf der Finite-Element-Methode, sinnvoll bzw. teilweise dringend erforderlich. Weitere Techniken zur Bewältigung größerer Probleme, die in dieser Arbeit zum Einsatz kommen, sind das Prinzip der Adaptivität und die Verwendung eines schnellen Lösungsverfahren für die anfallenden linearen Gleichungssysteme.

Die meisten von Ingenieuren verwendeten kommerziellen Programme für die numerische Simulation, wie z.B. NASTRAN ([46]), ASKA ([40]) und NISA II ([22]), sowie viele Programme in Wissenschaft und Forschung, wie z.B. ISARES ([58]) wurden vor nicht allzu langer Zeit entwickelt, als eine verteilte Berechnung auf einem lose gekoppelten Netzwerk von Arbeitsplatzrechnern bezüglich Software und Hardware kaum unterstützt wurden. Deshalb sind die Programme für eine sequentielle Ausführung konzipiert.

In letzter Zeit sind aber starke Bestrebungen an der Entwicklung von Algorithmen zu einer verteilten Finite-Element-Berechnung erkennbar (s. z.B. [1], [18], [23], [41], [44]). Dabei stützen sich die Verfahren auf die Methoden der Substrukturierung und der Gebietszerlegung. In [39] und [63] wurde zum ersten Mal ein allgemeines Verfahren entwickelt, das für verschiedenste FE-Programme eine Parallelisierung auf einem Workstationnetz erlaubt. Das Verfahren basiert auf dem Prinzip der rekursiven Substrukturierung und gehört zur Klasse der divide-and-conquer Algorithmen. Es ermöglicht eine mehrstufige Zerlegung eines technischen Objektes in Teilprobleme und den Einsatz von kommerziellen FE-Programmen.

Bei vielen anderen Vorgehensweisen ist entweder keine oder nur eine einstufige Substrukturierung möglich. Bei der Modellierung der einzelnen Bauteile spielt die Beschreibung der Schnittstellen eine untergeordnete Rolle. Nach einer Vernetzung der Bauteile ist diese entweder gar nicht oder nur schwer modifizierbar und die Berechnung damit nicht adaptiv. Es gibt Verfahren, die eine mehrstufige

Zerlegung von Bauteilen unterstützen (z.B. [16], [21]) und Verfahren, die mit einer einstufigen Substrukturierung adaptive Berechnungen ermöglichen (z.B. [13], [14], [52]). In [63] erfolgt dagegen die Modellierung, die Zerlegung und die Berechnung eines technischen Objekts in einer rekursiven baumartigen Datenstruktur. Dieses Prinzip erlaubt die Kombination von schnittstellenorientierter Sichtweise, adaptiver Anpassung der Vernetzung, paralleler Modellierung und paralleler Berechnung.

Ein großer Nachteil der Vorgehensweise in [63] ist, daß ein direktes Lösungsverfahren zur Lösung der entstehenden linearen Gleichungssysteme eingesetzt wird. Obwohl dieser Löser für die behandelte Problemklasse bezüglich Speicherbedarf, Rechenoperationen und Parallelisierbarkeit anderen direkten Verfahren überlegen ist und in vielen Anwendungen direkte Löser zum Einsatz kommen (z.B. [23], [36], [45], [50], [78]), erreicht er bei wachsender Problemgröße schneller die Grenzen der verfügbaren Rechnerleistung als ein effizientes iteratives Verfahren. Ziel dieser Arbeit ist es deshalb, ein iteratives Lösungsverfahren in das Prinzip der rekursiven Substrukturierung zu integrieren.

Das Prinzip der Substrukturierung führt zu einer Zerlegung des Berechnungsgebietes in nicht überlappende Teilgebiete. Für solche Zerlegungen gibt es iterative Lösungsverfahren, die abwechselnd in den einzelnen Teilgebieten und auf den Separatoren zwischen den Teilgebieten rechnen. Solche iterativen Verfahren, basierend auf einer Gebietszerlegung, werden auch Schwarzsche Methoden genannt und in einer Vielzahl von Arbeiten behandelt (z.B. [18], [32], [44]). Die Attraktivität ihrer Anwendung in der FE-Methode liegt in der Möglichkeit, die Lösung der Systemgleichungen zu parallelisieren.

Die Methode der Gebietszerlegung löst parallel die Gleichungssysteme in den Teilgebieten mit direkten oder iterativen Verfahren. Das Gleichungssystem der Separatoren wird mit einem vorkonditionierten Iterationsverfahren gelöst. Ein mehrfacher Wechsel zwischen den Teilgebieten und dem Gesamtgebiet führt zur Konvergenz der Iteration. Durch den Einsatz von schnellen hierarchischen Iterationsverfahren in den Teilgebieten, wie z.B. Mehrgitterverfahren, und den Einsatz von problemangepaßten Vorkonditionierern für die Separatorgleichungen, ergeben sich schnell konvergierende Lösungsverfahren für die Gleichungssysteme der FEM (z.B. [33]).

Da die Gebietszerlegungsverfahren im Bereich der Mathematik entwickelt werden, konzentrieren sie sich auf eine effiziente Lösung der Systemgleichungen. Sie berücksichtigen nicht den Aspekt der Modellierung des zu berechnenden Objekts. Deshalb beschränken sich alle oben genannten Gebietszerlegungsverfahren auf eine einstufige Substrukturierung und sind meist schwer mit adaptiven Verfahren zu koppeln. Der Zugang zu adaptiven FE-Berechnungen wird durch Partitionierungsalgorithmen für unstrukturierte FE-Netze erreicht (z.B. [45], [52], [65]). Bei der rekursiven Substrukturierung wird dagegen die Gebietszerlegung bereits durch die Modellierung und der adaptiven rekursiven Zerlegung automatisch mitgeliefert.

In [30] und [31] werden auch Gebietszerlegungsverfahren mit rekursiven Zerlegungen betrachtet. Anstelle der normalerweise verwendeten Knotenbasisfunktionen wird hier ein Erzeugendensystem bestehend aus allen Knotenbasen der feinen und groben Gitter zur Darstellung der Lösung benutzt (s. [29]). In diesem Erzeugendensystem lassen sich gebietsorientierte Iterationsverfahren implementieren, die ein mehrgitterartiges Konvergenzverhalten besitzen und in ihrer Komplexität für die betrachtete Problemklasse optimal sind [32].

In dem in der vorliegenden Arbeit entwickelten Lösungsverfahren werden bei einer adaptiven Verfeinerung ähnliche rekursive Zerlegungen wie in [31] verwendet. Zusätzlich sind aber auch allgemeinere Teilungsstrategien möglich. Anstelle des Erzeugendensystem erfolgt hier aber der Einsatz von hierarchischen Basen [71]. Dadurch wird das baumartige System von Gleichungen ohne Einführung zusätzlicher Freiheitsgrade mit einem Lösungsverfahren der Komplexität $O(n \cdot log(n)^2)$ bzgl. der arithmetischen Operationen gelöst, was deutlich besser als das direkte Verfahren ist. Es sind lediglich einige zusätzliche Transformationen der Matrizen notwendig. Eine Integration der Idee des Erzeugendensystems in das Prinzip der rekursiven Substrukturierung ist möglich, erfordert aber umfassende Modifikationen des Berechnungsverfahren und konnte in der vorliegenden Arbeit nicht mehr realisiert werden. Für zukünftige Weiterentwicklungen des Verfahrens, insbesondere um eine optimale Komplexität zu erzielen, scheint dieser Schritt aber sinnvoll.

Die explizite Transformation der Matrizen bezüglich der hierarchischen Basis ist ein wesentlicher Unterschied und Vorzug gegenüber den meisten anderen Verfahren. Da das iterative Verfahren explizit mit den hierarchischen Basen rechnet, kann die Iteration auf die Freiheitsgrade konzentriert werden, die einen wesentlichen Beitrag zur Verbesserung der Lösung liefern. Diese Methode der adaptiven Relaxation (s. [59], [61]) kann in den Lösungsprozeß der rekursiven Substrukturierung integriert werden und liefert gleichzeitig ein Kriterium zur Steuerung der adaptiven Verfeinerung. Herkömmliche Iterationsverfahren verwenden zwar ebenfalls die hierarchische Idee, sie wird aber meistens nur zur Vorkonditionierung verwendet (z.B. [17], [33]). Der eigentliche iterative Lösungsprozeß erfolgt in Knotenbasen, die alle einen gleichwertigen Beitrag zur Lösung beisteuern und eine adaptive Relaxation nicht unterstützen.

Die Zielsetzung der Arbeit ist die Entwicklung eines parallelen Berechnungsverfahren der Finite-Element-Methode für verschiedenste Anwendungsgebiete, das die Teamarbeit von Ingenieuren unterstützt. Dabei soll der Einsatz von im Rechnernetz vorhandener Soft- und Hardware ermöglicht werden. Das Verfahren soll ein einheitliches Konzept von der Modellierung über den Lösungsalgorithmus bis zur Analyse der Berechnungsergebnisse in einer geeigneten Datenstruktur realisieren und die Möglichkeit einer adaptiven Netzverfeinerung entsprechend dem gestellten Problem bieten. Zur Lösung der entstehenden Systemgleichungen ist ein iteratives Lösungsverfahren mit hierarchischem Konzept zu entwickeln.

Die Realisierung dieser Vorgaben basiert auf dem Prinzip der rekursiven Substrukturierung. Durch die schnittstellenorientierte Modellierung bietet sich die Möglichkeit der Implementierung von offenen Schnittstellen zu externen FEM Programmen. Es kommt ein iteratives Lösungsverfahren zum Einsatz, das explizit in hierarchischen Basen rechnet und mit der Methode der adaptiven Relaxation kombiniert wird. Mit der baumartigen Zerlegung des Problems können parallele numerische Simulationen in natürlicher Weise auf lose gekoppelten Netzwerken von Arbeitsplatzrechnern durchgeführt werden.

Im einzelnen ist die Arbeit wie folgt gegliedert:

Kapitel 2 gibt eine kurze Einführung in die Finite-Element-Methode. Dabei wird im Hinblick auf eine Parallelisierung der Berechnungsablauf in Teilschritte gegliedert. Weiter wird das adaptive Verfahren der FEM basierend auf Fehlerschätzung und Netzverfeinerung erläutert und die in der Arbeit behandelten Anwendungsgebiete angesprochen.

Kapitel 3 befaßt sich mit dem Prinzip der Rekursiven Substrukturierung in der FEM. Dieses Prinzip bildet die Basis der vorliegenden Arbeit. Ausgehend von der Beschreibung eines technischen Objektes wird in vier Berechnungsschritten ein Finite-Element-System aufgebaut und gelöst. Die baumartige Zerlegung des Objektes spiegelt sich im gesamten Lösungsprozeß inklusive der Gleichungslösung wider. Es handelt sich um ein effizientes direktes Lösungsverfahren, das ideal für adaptive Netzverfeinerung geeignet ist und die Möglichkeit zur Parallelisierung bietet.

Basierend auf dem Berechnungsverfahren aus Kapitel 3 wird in Kapitel 4 ein neues iteratives Lösungsverfahren für die Gleichungssysteme des Substrukturierungsbaumes entwickelt. Das Vorgehen wird motiviert durch den Vorteil iterativer Löser gegenüber direkten Methoden, insbesondere bezüglich der Besetzungsstruktur der Matrizen und des geringeren Rechenaufwands. Der iterative Algorithmus wird in zwei Teilschritte gegliedert und im Detail beschrieben. Abschließend wird eine symmetrische Variante des Algorithmus vorgestellt.

In Kapitel 5 werden die hierarchischen Basen eingeführt und in das iterative Verfahren integriert. Dadurch entsteht ein effizientes Lösungsverfahren für eine adaptive parallele Realisierung der FEM. Es folgen die Beschreibung der Auswirkungen der hierarchischen Basen auf die einzelnen Teilschritte der Berechnung, die Systemmatrizen sowie die Konvergenz des Verfahrens. Weiter wird gezeigt, daß die Vorgehensweise auch für Finite Elemente mit einem höheren Ansatzgrad geeignet ist.

Das bis dahin beschriebene Berechnungsverfahren wird in Kapitel 6 mit dem Prinzip der adaptiven Relaxation kombiniert. Durch diese Methode können in dem hierarchisierten, substrukturierten Gesamtsystem die Glättungsschritte auf die Teilgebiete konzentriert werden, die einen wesentlichen Anteil zur Lösung beitragen. Zudem liefert sie ein Kriterium zur adaptiven Verfeinerung der Netze.

In Kapitel 7 werden die Berechnungskosten des iterativen Algorithmus bezüglich Speicherbedarf und Rechenoperationen analysiert und mit anderen Verfahren, insbesondere dem direkten Verfahren der rekursiven Substrukturierung, verglichen.

In Kapitel 8 wird die Parallelisierung des Berechnungsverfahren erläutert. Am Beispiel einiger Verteilungsstrategien wird eine Analyse der verteilten FE-Simulation durchgeführt.

Die Aussagen der Arbeit werden in Kapitel 9 anhand verschiedener praktisch durchgeführter numerischer Simulationen belegt. Abschließend erfolgt in Kapitel 10 eine Zusammenfassung und ein Ausblick der Arbeit.

2 Die Finite-Element-Methode

In diesem Kapitel erfolgt eine Beschreibung gewisser Aspekte der Finite-Element-Methode (FEM), die für das Verständnis der Arbeit hilfreich sind. Eine detaillierte Einführung ist z.B. in [9], [12] oder [64] zu finden.

2.1 Das Berechnungsverfahren der FEM

Die *Finite-Element-Methode* ist ein rechnerorientiertes mathematisches Verfahren zur numerischen Approximation von Differential- und Integralgleichungen, die die Antwort eines physikalischen Systems auf externe Einflüsse beschreiben [12]. Die FEM ist damit ein Lösungsverfahren zur numerischen Simulation von technischen Objekten. Aufgrund ihrer hohen Robustheit, Flexibilität und ihres großen Anwendungsspektrums findet sie in der Industrie und Forschung breite Anwendung. So lassen sich zum Beispiel verschiedenste Probleme aus der Festkörpermechanik, Wärmeleitung, Strömungsmechanik und Akustik bis hin zur Schaltkreissimulation mit ihr behandeln. Die simulierten technischen Gebilde können beliebige geometrische Formen in mehreren Dimensionen haben. Weitere Vorteile sind, daß ihre physikalischen Eigenschaften, wie z.B. Materialparameter, Steifigkeiten oder Wärmeleitungskoeffizienten über das System variieren können und externe Einflüsse, sogenannte Lasten, in jeder physikalischen Form in die Simulation einbezogen werden können. Schließlich kann man mit der FEM sowohl lineare als auch nichtlineare Problemstellungen lösen.

Wenn man die Finite-Element-Methode in einem Computer-Programm realisieren will, ist es sinnvoll, das Verfahren in mehrere Teilschritte zu gliedern.

Schritt 1_{FE}: Man bildet ein Modell des technischen Objekts und diskretisiert es in Form eines Netzes. Dadurch wird das Grundgebiet in disjunkte Teilgebiete, sogenannte Elemente, zerlegt. Es werden dabei vorgegebene Elementtypen verwendet, z.B. Dreiecke, Vierecke, Tetraeder und Quader. Die Anzahl der Elemente und ihre Verteilung ist abhängig vom Problem und hat Einfluß auf die Genauigkeit der Berechnung.

Schritt 2_{FE}: In jedem Teilgebiet wird entsprechend der Differentialgleichung, die das gegebene Problem beschreibt, ein lineares Gleichungssystem (LGS) gebildet. Die Ermittlung der elementlokalen Gleichungssysteme ist abhängig von der Wahl der Ansatzfunktionen (die auch als Formfunktionen bezeichnet werden), der Form der Elemente und dem zugrunde liegenden Extremalprinzip [64]. Die Matrix des Elementgleichungssystems enthält die Strukturinformation des Elements, die rechte Seite Informationen über die inneren und äußeren Lasten.

Schritt 3_{FE}: Durch die Forderung nach Stetigkeit der Lösung über das gesamte Gebiet werden die Elementgleichungssysteme zu einem großen, globalen

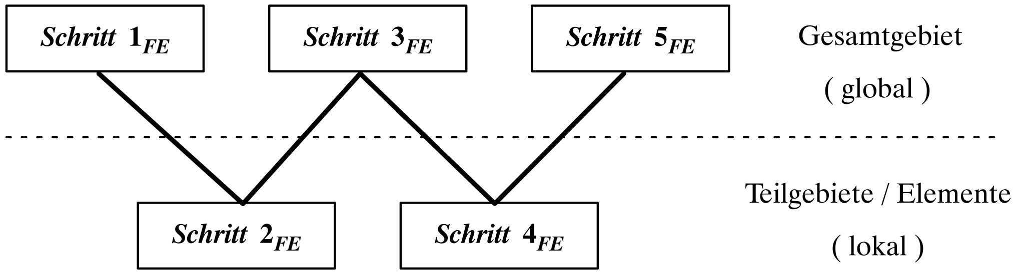


Abbildung 1: Darstellung der FEM in Form eines W-Zyklus

Gleichungssystem assembliert. Die Unbekannten dieses Systems werden als Freiheitsgrade bezeichnet. Danach werden die Randbedingungen in das System integriert.

Im zweiten Teil dieses Schritts wird das globale lineare Gleichungssystem gelöst. Als Ergebnis erhält man den globalen Lösungsvektor mit den diskreten Werten der Freiheitsgrade an den Gitterpunkten des FE-Netzes.

Schritt 4_{FE}: Jedes Teilgebiet erhält seinen Anteil vom globalen Lösungsvektor. Mit der elementlokalen Strukturinformation können aus dem lokalen Lösungsvektor weitere für den Ingenieur wichtige Zustandsgrößen abgeleitet werden. In der ebenen Elastizität handelt es sich hierbei z.B. um Spannungen und Verzerrungen.

Schritt 5_{FE}: Die topologische Information über das diskretisierte Modell und die berechneten Ergebnisse werden für eine Auswertung der Berechnung zusammengefaßt. Die entstehende tabellarische Information über Elementeinteilung, Koordinaten der Gitterpunkte, Lösungsvektor und abgeleitete Größen können z.B. an einen Fehlerschätzer oder einen Postprozessor zur graphischen Darstellung der numerischen Simulation übergeben werden.

Die Unterteilung der FEM in fünf Teilschritte läßt sich graphisch in Form eines W-Zyklus darstellen (s. Abb. 1) und wird in der weiteren Arbeit als Standardverfahren der Finite-Element-Methode (Standard-FEM) bezeichnet. In der Graphik wird verdeutlicht, daß gewisse Teilschritte nur elementlokale Information brauchen, während andere Schritte eine globale Sichtweise des gesamten Systems erfordern. So können die Schritte 2_{FE} und 4_{FE} in den verschiedenen Teilgebieten voneinander unabhängig (d.h. parallel) behandelt werden.

Ein weiterer Aspekt dieser Unterteilung ist, daß die Teilschritte 2_{FE} und 4_{FE} sowie der später eingeführte Schritt der Fehlerschätzung problemabhängig sind. Die Schritte 1_{FE}, 3_{FE} und 5_{FE} sind problemunabhängig und werden für alle Probleme identisch ausgeführt. Dies läßt sich, wie in Kapitel 3.2.6 gezeigt wird, durch Einsatz eines modularen Programmkonzepts über Schnittstellen ausnutzen und ermöglicht so eine flexible Anpassung an verschiedene FE-Problemstellungen.

2.2 Die Anwendungen: Scheibenprobleme, Wärmeleitung

Um einen Einblick in die mathematischen und mechanischen Grundlagen der FEM zu gewinnen, wird in diesem Kapitel kurz auf die in der vorliegenden Arbeit behandelten Problemstellungen eingegangen. Es handelt sich dabei um Berechnungen von Wärmeleitungsproblemen und Scheibenproblemen aus dem Gebiet der ebenen Elastizität (s. z.B. [64]). Die zu Grunde liegenden Variationsformulierungen und Gleichgewichtsbedingungen, die eine Bestimmung der Elementmatrizen ermöglichen, werden im folgenden anhand des Scheibenproblems kurz skizziert.

Für die numerische Simulation von Scheiben bildet man ein zweidimensionales Berechnungsmodell. Allgemein wird bei der FE-Simulation jede mechanische Beanspruchung eines Kontinuums durch Zustandsgrößen beschrieben. Für das Scheibenproblem sind dies die *Verschiebungsgrößen* $u = (u_x, u_y)$ in x- und y-Richtung, die *Schnittkraftgrößen* $\sigma = (\sigma_x, \sigma_y, \tau_{xy})$, wobei σ_x und σ_y Normalspannungen sind und τ_{xy} eine Schubspannung darstellt sowie die *Verzerrungsgrößen* $\varepsilon = (\varepsilon_x, \varepsilon_y, \gamma_{xy})$, bestehend aus den Dehnungen ε_x, ε_y und der Schiebung γ_{xy}. Mit Hilfe dieser Zustandsgrößen kann man eine *Gleichgewichtsbedingung* formulieren [42]

$$D_g \cdot \sigma + p = 0, \tag{1}$$

wobei $p = (p_x, p_y)$ eine äußere oder innere Belastung und D_g der Differentialoperator

$$D_g = \begin{pmatrix} \frac{\partial}{\partial x} & 0 & \frac{\partial}{\partial y} \\ 0 & \frac{\partial}{\partial y} & \frac{\partial}{\partial x} \end{pmatrix} \tag{2}$$

ist.

In einem elastischen Werkstoff legt ein *Elastizitätsgesetz* den linearen Zusammenhang zwischen Schnittkräften und Verzerrungen fest:

$$\sigma = C \cdot \varepsilon \tag{3}$$

Dabei ist C die sogenannte Stoff- oder Elastizitätsmatrix.

Schließlich leitet man durch *kinematische Aussagen* die Verzerrungen aus den Ableitungen der Verschiebungsgrößen her:

$$\varepsilon = D_{\varepsilon u} \cdot u \tag{4}$$

Auch hier ist $D_{\varepsilon u}$, die *Verzerrungs-Verschiebungs-Matrix*, ein Differentialoperator.

$$D_{\varepsilon u} = \begin{pmatrix} \frac{\partial}{\partial x} & 0 \\ 0 & \frac{\partial}{\partial y} \\ \frac{\partial}{\partial y} & \frac{\partial}{\partial x} \end{pmatrix} \tag{5}$$

Aus den Gleichungen (1) bis (5) kann man für den konkreten Fall des Scheibenproblems mit ebenem Spannungszustand zwei Differentialgleichungen 2. Ordnung herleiten [64]:

$$\frac{E}{1-\nu^2} \cdot \frac{\partial^2 u_x}{\partial x^2} + \frac{E}{2(1-\nu)} \cdot \frac{\partial^2 u_y}{\partial x \partial y} + \frac{E}{2(1+\nu)} \cdot \frac{\partial^2 u_x}{\partial y^2} = -p_x \tag{6}$$

$$\frac{E}{1-\nu^2} \cdot \frac{\partial^2 u_y}{\partial y^2} + \frac{E}{2(1-\nu)} \cdot \frac{\partial^2 u_x}{\partial x \partial y} + \frac{E}{2(1+\nu)} \cdot \frac{\partial^2 u_y}{\partial x^2} = -p_y \tag{7}$$

Für das System der beiden partiellen Differentialgleichungen (6), (7) läßt sich nur in wenigen Spezialfällen eine exakte analytische Lösung bestimmen. Deswegen wird eine Näherungslösung basierend auf einer Energieformulierung bestimmt.

Bei Problemen der Strukturmechanik, zu denen das Scheibenproblem gehört, betrachtet man dazu die *totale potentielle Energie* eines Systems.

$$\Pi = \frac{1}{2}\int_{\Omega} \sigma^T \varepsilon d\Omega - \int_{\Omega} p_{\Omega}^T u d\Omega - \int_{\Gamma} p_{\Gamma}^T u d\Gamma \tag{8}$$

In (8) entspricht der erste Term der Arbeit der inneren Spannungen und die übrigen Terme dem Potential der angreifenden äußeren Kräfte, wobei zwischen Volumen- und Randkräften unterschieden wird. Das Prinzip der minimalen potentiellen Energie besagt, daß unter allen möglichen Verschiebungszuständen, welche den kinematischen Randbedingungen genügen, der tatsächliche Gleichgewichtszustand die potentielle Energie Π minimiert.

Eine äquivalente Möglichkeit, das Gleichgewicht des Körpers auszudrücken, bietet das Prinzip der virtuellen Verschiebungen (s. z.B. [9]).

Nachdem durch die Vernetzung der eigentlich kontinuierliche Körper in Elemente eingeteilt wurde, definiert man lokal begrenzte Ansatzfunktionen als Funktionen der Verschiebungen der Knotenpunkte der Finiten Elemente. Unter allen Lösungen, die sich von den Ansatzfunktionen aufspannen lassen, ermittelt das Finite-Element-Verfahren diejenige, die die potentielle Energie minimiert. Bezüglich der Ansatzfunktionen für die Verschiebungen der Knotenpunkte können die Integrale der Energie- und Arbeitsprinzipien ausgewertet und so die Elementmatrizen ermittelt werden.

Anschließend an den Berechnungsschritt 2_{FE} zur Ermittlung der Elementgleichungssysteme ist in Schritt 3_{FE} ein globales Gleichungssystem für das technische Objekt zu erstellen und zu lösen. Am Ende dieses Schritts liegt in allen Gitterpunkten des FE-Netzes die Lösung der Verschiebungsgrößen vor. Bei der Berechnung von technischen Objekten sind Ingenieure aber häufig nicht nur an diesen Verschiebungen interessiert. Zur Beurteilung der Belastbarkeit eines Objektes sind aus den Verschiebungen abgeleitete Größen, sogenannte Schnittkräfte, von Interesse. Bei den Scheiben kann man mit Hilfe der Gleichungen (3) und (4) die Spannungen und Verzerrungen aus den Verschiebungen berechnen:

$$\sigma = C D_{\varepsilon u} \cdot u \tag{9}$$

Die Ermittlung der Schnittkräfte erfolgt somit bei linearen FE-Berechnungen durch eine lineare Abbildung.

Ein weiteres Anwendungsfeld für die Finite-Element-Methode, das in dieser Arbeit behandelt wird, ist die stationäre Temperaturverteilung bei Wärmeleitungsproblemen. Hierbei wird der Spezialfall betrachtet, daß keine inneren Wärmequellen vorhanden sind und daher die Temperaturfunktion $u(x, y)$ die *Laplace-Gleichung*

$$\triangle u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0 \tag{10}$$

erfüllen muß. Auch hier kann das Problem durch Zerlegung des Berechnungsobjekts in Elemente und mit einem die Energie minimierenden Extremalprinzip übergeführt werden in die Verarbeitung linearer Elementgleichungssysteme (s. z.B. [64]).

2.3 Die adaptive Finite-Element-Methode

Ein Nachteil der FEM liegt darin, daß ihre Ausführung auf einem Rechner bei größeren Problemen einen hohen Speicher- und CPU-Bedarf hat. Eine Möglichkeit, diesen Bedarf zu reduzieren, bieten adaptive Verfahren, die man seit einigen Jahren in der Forschung einsetzt, die aber in industriellen Anwendungen erst langsam Einzug halten.

Eine FE-Lösung ist eine Approximationslösung, deren Genauigkeit von der Maschenweite des Elementnetzes abhängt. Durch eine Verfeinerung des Netzes erhält man eine verbesserte Lösung. Desweiteren ist der Fehler einer FE-Lösung meistens lokal unterschiedlich verteilt. In vielen Anwendungsproblemen ist es möglich, den Fehler aus einer FE-Berechnung abzuschätzen und die Elemente mit einem relativ großen lokalen Fehler zu ermitteln. Mit Hilfe dieser Fehlerschätzung kann man ein FE-Netz adaptiv, dem Problem angepaßt, lokal verfeinern. Diese Erkenntnisse führen zu einem adaptiven FE-Verfahren, das aus zwei wesentlichen zusätzlichen Teilschritten besteht, nämlich der a-posteriori *Fehlerschätzung* und der lokalen *Netzverfeinerung*.

Die a-posteriori Fehlerschätzung greift auf Ergebnisse eines bereits berechneten Netzes zurück, um die Genauigkeit der Lösung zu beurteilen sowie die Elemente zu bestimmen, die zu verfeinern sind. Dabei werden Aussagen über die Verteilung des Fehlers, die Größe des Gesamtfehlers und die Erfüllung von Abbruchkriterien getroffen. Anschließend an die Fehlerschätzung kann nun die Diskretisierung des technischen Objekts (d.h. seine Vernetzung) so verändert werden, daß man in einem erneuten Berechnungslauf eine genauere Lösung erhält. So ergibt sich das in Abb. 2 gezeigte rückgekoppelte Verfahren der adaptiven Finite-Element-Methode. Das zu untersuchende Objekt wird mehrmals berechnet und

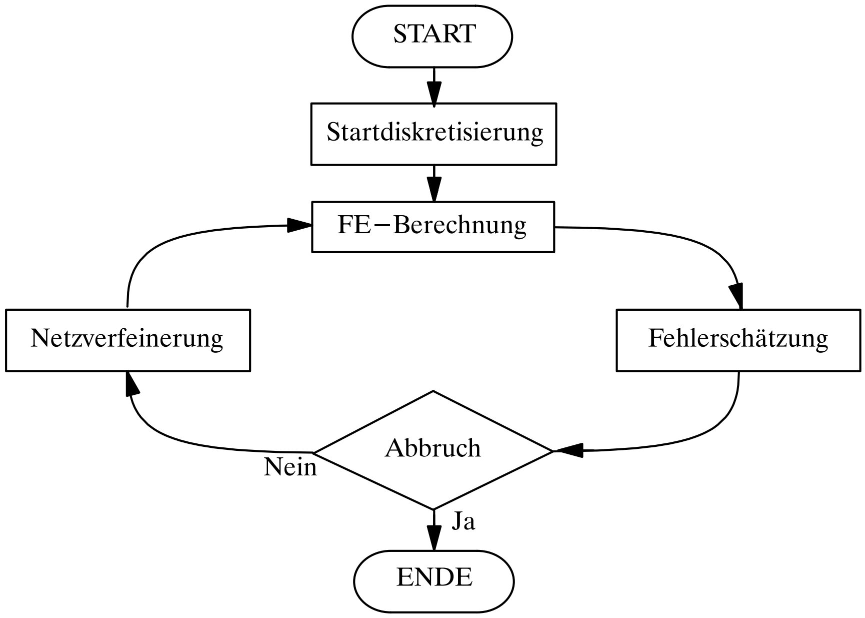


Abbildung 2: Adaptives Verfahren der Finite-Element-Methode

verfeinert, bis eine vom Ingenieur vorgegebene Genauigkeitsanforderung erfüllt ist.

Das in Kapitel 2.1 beschriebene Verfahren erweitert man auf ein adaptives Verfahren, indem die gesammelte Information am Ende von Schritt 5_{FE} an einen Fehlerschätzer übergeben wird. Anschließend wird durch einen modifizierten Schritt 1_{FE} die Diskretisierung dem Problem entsprechend verfeinert.

2.3.1 Fehlerschätzung

Die genaue Durchführung der *Fehlerschätzung* ist problemspezifisch, und es wird hier nur kurz auf die in dieser Arbeit eingesetzten Indikatoren eingegangen. Um für jedes Element den lokalen Fehler zu ermitteln und so die Elemente zu bestimmen, die zu verfeinern sind, wird als Maß die Energienorm des Fehlers

$$\|e\|_E = (\int_\Omega e^T D_{\varepsilon u}^T C D_{\varepsilon u} e \quad d\Omega)^{\frac{1}{2}} \tag{11}$$

mit $e = u - u_h$, der Differenz aus exakter und diskreter Lösung, verwendet. Wegen (9) läßt sich der Elementfehler in der Energienorm mit $e_\sigma = \sigma - \sigma_h$ auch wie folgt schreiben.

$$\|e\|_i^2 = \int_{\Omega_i} e_\sigma^T C^{-1} e_\sigma d\Omega \tag{12}$$

Im Bereich der ebenen Elastizität wurden in den letzten Jahren mehrere Fehlerestimatoren entwickelt, von denen zwei hier vorgestellt werden. Beide Estimatoren werten die Tatsache aus, daß die berechnete Finite-Element-Lösung zwar in den Verschiebungen stetig ist, aber in den Spannungen Unstetigkeiten an den Elementrändern aufweist.

Im ersten Fehlerschätzer wird der Fehler in der Energienorm abgeschätzt. Dabei ergibt sich laut Babuska und Miller ([3], [4]) folgende Abschätzung für den Elementfehler in der Energienorm:

$$\|e\|_i^2 = C_1 h_i^2 \int\limits_{\Omega_i} (Lu - f)^T (Lu - f) d\Omega \quad + \quad C_2 h_i \int\limits_{\Gamma_i} J_\sigma^T J_\sigma d\Gamma \tag{13}$$

mit $L = D_{\varepsilon u}^T C D_{\varepsilon u}$. Der erste Term enthält das Residuum der Differentialgleichung im Element und entspricht der Verletzung des inneren Gleichgewichts. Der zweite Term berücksichtigt J_σ, die Spannungssprünge normal zu dem Elementrändern. Die Konstanten C_1 und C_2 sind abhängig vom Problem- und Elementtyp und sind für die hier behandelten linearen Scheibenprobleme berechenbar; h_i ist der Elementdurchmesser.

Eine zweite Möglichkeit der Fehlerschätzung ist es, die Energienorm der Differenz zwischen exakter Spannung σ und den FE-Spannungen σ_h mit (12) zu messen. Da die Werte für σ unbekannt sind, verwendet man mit $e_\sigma = \sigma^* - \sigma_h$ eine Abschätzung für den Fehler in den Spannungen. Das Problem besteht darin, die verbesserten Spannungswerte σ^* durch sogenannte Glättungsverfahren ohne eine teure Neuberechnung zu bestimmen. Zienkiewicz verwendet in [74] höhere stetige Ansatzfunktionen für die Spannungen σ^* und minimiert die Differenz zwischen σ_h und σ^* (L_2-Glättung). In einem anderem Glättungsverfahren wird eine Spannungsfunktion aus den superkonvergenten Punkten ermittelt, an denen die Spannungen eine Genauigkeit höherer Ordnung haben ([75], [77]). Ein sehr einfaches, aber effektives Verfahren zur Spannungsglättung ist die Mittelung der Spannungen an den Knotenpunkten, die deshalb in dieser Arbeit eingesetzt wurde. All diese Glättungsverfahren haben den Nachteil, daß für die verbesserte Lösung mathematisch nicht garantiert ist, ob sie tatsächlich näher an der exakten Lösung liegt. In den betrachteten Anwendungen erweisen sie sich aber als sehr robust und effektiv.

Weitere Verfahren zur Fehlerschätzung, die im Bereich der ebenen Elastizität eingesetzt werden, sind z.B. die Auswertung der Spannungsgradienten oder die Spannungsdifferenzen zwischen zwei aufeinanderfolgenden Netzen [62].

Ein weiteres Kriterium zur Netzverfeinerung ergibt sich aus der Methode der adaptiven Relaxation, die in Kapitel 6 beschrieben wird. Dabei handelt es sich aber um keinen Fehlerschätzer, sondern lediglich um einen Verfeinerungsindikator. Im Gegensatz zu den bisher behandelten Indikatoren ist dieser aber auf alle in der Arbeit betrachteten Problemklassen anwendbar und erfordert keine globale Sichtweise der Berechnungsergebnisse.

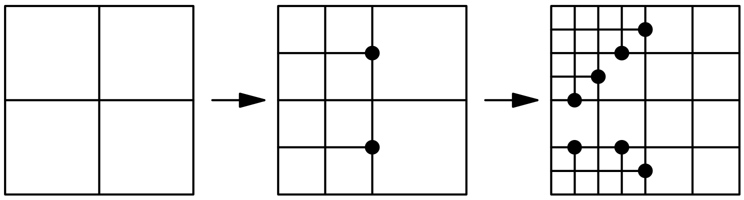


Abbildung 3: Hierarchische Netzverfeinerung mit inkompatiblen Punkten

2.3.2 Netzverfeinerung

Der zweite zusätzliche Teilschritt eines adaptiven FE-Verfahrens umfaßt die *Netzverfeinerung.* In dieser Arbeit wird als Netzverfeinerungsstrategie die h-Verfeinerung verwendet. Die *h-Verfeinerung* ist eine Netzverfeinerungsstrategie, die einzelne Elemente eines FE-Netzes in mehrere neue Elemente zerlegt und so lokal die Maschenweite des Netzes verkürzt. Wird diese Zerlegung von Elementen in Teilelemente rekursiv angewandt, entstehen hierarchisch geschachtelte Finite-Element-Netze. In Abb. 3 sieht man eine Folge von adaptiv verfeinerten FE-Netzen. In diesen Netzen wird ein Viereck so in vier neue Elemente zerlegt, daß die Elementseiten halbiert werden. Bei einer solchen adaptiven Verfeinerung entstehen *inkompatible Punkte* an der Grenze von verfeinerten zu unverfeinerten Elementen. Ein Punkt ist inkompatibel, wenn er nicht zu allen ihn berührenden Elemente gehört, d.h. wenn er auf dem Rand eines Elements liegt, aber keine entsprechende Freiheitsgrade in diesem Element hat. Die inkompatiblen Punkte an der Grenze von verfeinerten zu unverfeinerten Elementen werden aus dem Gleichungssystem eliminiert (s. Kap. 3.2.2). Ihre Freiheitsgrade werden durch eine Interpolation aus den angrenzenden kompatiblen Punkten bestimmt, um so eine Verletzung der Stetigkeitsforderung zu verhindern .

Diese Verfeinerungsstrategie läßt sich ideal mit effizienten hierarchischen Rechentechniken wie der rekursiven Substrukturierung und den Multilevelverfahren kombinieren. Außerdem garantiert sie durch die strenge Schachtelung der Ansatzräume einen von Verfeinerung zu Verfeinerung monoton fallenden Fehler in der Energienorm.

In der Literatur gibt es zahlreiche Varianten zur Netzverfeinerung und zur Behandlung der inkompatiblen Punkte, z.B. Verfahren, die nach der Fehlerschätzung das Gebiet komplett neu vernetzen oder Verfahren, die inkompatible Punkte mit temporären Übergangselementen in kompatible Punkte umwandeln (s. z.B. [55], [76]). Für unser angestrebtes Ziel der Entwicklung eines allgemeinen parallelisierbaren FE-Algorithmus unter Verwendung effizienter Lösungsalgorithmen bringen diese Verfahren keine wesentlichen Vorteile.

Die Verfahren, bei denen eine Neuvernetzung basierend auf der Fehlerverteilung durchgeführt wird, ermöglichen zwar eine sehr schnelle Anpassung des Netzes an die Problemstellung, löschen aber das alte Netz. Dadurch sind die effizienten hierarchischen Rechentechniken nicht anwendbar. Verfahren mit temporären Übergangselementen verletzen die strenge hierarchische Schachtelung der FE-Räume, da das alte Netz nicht im neuen Netz enthalten ist, und benötigen i.a. ungünstig geformte Elemente in den Übergangsbereichen. Eine weitere Alternative, die p-Verfeinerung, die eine lokale Erhöhung des Ansatzgrades durchführt, kann nur bis zu einem gewissen Grad getrieben werden und ist nur in Verbindung mit der h-Version sinnvoll.

Durch den Einsatz von Adaption ist es möglich, eine Genauigkeit der Berechnung zu erzielen, die bei einer uniformen Verfeinerung nur mit sehr viel mehr Elementen erreichbar ist. Dadurch ergibt sich ein wesentlich günstigeres Verhältnis von Rechenaufwand zu Ergebnisgenauigkeit. Ein weiterer Vorteil besteht darin, daß ein Ingenieur das technische Objekt nur in verhältnismäßig wenige Teilgebiete mit zulässiger Elementform zerlegen muß. Das endgültige FE-Netz entsteht vollautomatisch durch mehrere aufeinanderfolgende FE-Berechnungen mit einer im weiteren genauer dargestellten rekursiven Zerlegung der Elemente.

3 Das Prinzip der rekursiven Substrukturierung

Die Anwendung der rekursiven Substrukturierung in der FEM wird in [63] sehr ausführlich und formal behandelt. Da die vorliegende Arbeit auf diesem Verfahren aufbaut und alle seine für die numerische Simulation günstige Eigenschaften übernimmt, wird das Verfahren hier nochmals in kompakter und anschaulicher Art und Weise aufbereitet. Für das Verständnis der Arbeit ist diese Darstellung ausreichend, bei weitergehenden Interesse wird der Leser an [63] verwiesen.

3.1 Substrukturierung in der numerischen Simulation

Das Prinzip der Substrukturierung ist ein bei den Ingenieuren seit langem beliebtes Hilfsmittel sowohl zur Modellbildung als auch auf dieser Form der Modellbildung basierenden Berechnung von technischen Objekten (s. z.B. [16], [21], [51], [56]). Der Einsatz der Substrukturierung bewirkt in der FEM eine wesentliche Speicherplatzersparnis und ermöglicht so die Berechnung von sehr komplexen Gebilden. Durch die Entwicklungen im Soft- und Hardwarebereich kommt als zusätzlicher Aspekt der Substrukturierung die Möglichkeit der Verteilung der einzelnen Teilstrukturen auf verschiedene Rechnereinheiten und damit eine Parallelisierung der FEM hinzu [1].

Technische Gebilde sind auf natürliche Weise rekursiv in Teilgebilde aufteilbar. In einem Bausteinkonzept wird ein Gesamtsystem anschaulich aus Teilstrukturen zusammengesetzt, die jeweils eine logische Einheit bilden. Durch den stufenweisen Zusammenbau der Teilmodelle zu einen technischen System erhält man eine Mehrebenen-Substrukturtechnik [70]. Bereits bei der Konstruktion kann eine personelle Aufteilung der Bauteile zu verschiedenen Ingenieuren erfolgen. Mögliche Anwendungen sind z.B. die numerische Simulation von Autos, Flugzeugen, Schaltkreisen oder komplexen Tragsystemen. Ein Konstrukteur eines nicht mehr zerlegten Bauteils kennt alle Details dieses Bauteils, hat aber kein genaueres Wissen über andere Bauteile oder das gesamte Objekt. Ein Ingenieur, der mehrere Bauteile zu einer Baugruppe zusammensetzt, kennt wiederum nur die Schnittstellen und das prinzipielle Verhalten der Bauteile. Die Vorgehensweise bei der Substrukturierung entspricht der Arbeitsweise der Ingenieure bei der statischen Berechnung komplexer Tragsysteme.

Am Beispiel der einstufigen Substrukturierung wird kurz die prinzipielle Vorgehensweise erläutert. Zuerst wird das zu simulierende Objekt unter Einführung von *Separatoren* in disjunkte Gebiete zerlegt. Jedes Teilgebiet wird gemäß der Standard-FEM vernetzt und es wird ein entsprechendes FE-Gleichungssystem aufgestellt. Durch die statische Kondensation (s. Kap. 3.2.2) eliminiert man die Unbekannten im Gebietsinneren und überträgt damit die strukturellen Eigenschaften der Teilgebiete auf ihre Ränder. Nun wird ein Gesamtgleichungssystem aus den Teilgebietsrändern assembliert, das eine niedrigere Dimension hat als das vergleichbare System bei Verwendung der Standard-FEM mit einer Assemblie-

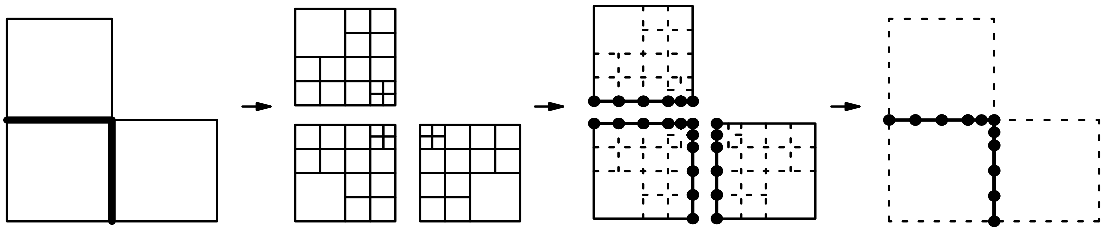


Abbildung 4: Vorgehensweise der Substrukturtechnik: Zerlegung, Vernetzung der Teilgebiete, statische Kondensation, Assemblierung

rung aller Elemente in ein Gleichungssystem. In Abb. 4 sind diese vier Schritte an einer Winkelscheibe veranschaulicht, einem Beispiel aus der Tragwerksanalyse. In der ersten Abbildung erkennt man zwei fett gezeichnete Separatoren, in der zweiten die drei vernetzten Teilgebiete, in der dritten die kondensierten Unbekannten im Inneren der Teilgebiete (gestrichelt dargestellt) und die Unbekannten auf den Teilgebietsrändern (durch Punkte dargestellt) und schließlich in der letzten Abbildung das verbleibende System.

Anschließend kann man aus der Lösung des Gleichungssystems der Separatoren die Lösung in den Teilgebieten berechnen. Diese Vorgehensweise ermöglicht die Berechnung von speicherplatzintensiveren Modellen und steigert die Effektivität der Simulation. Zudem bieten sich Vorteile bei der Wiederholungsrechnung mit leicht geänderten Berechnungsmodellen (Reanalyse) und einer lokalen Netzverdichtung [70].

Wenn man bei der Modellierung und der Berechnung Teilgebiete wieder rekursiv in Teilgebiete zerlegt, erhält man das *Prinzip der rekursiven Substrukturierung* in der Finite-Element-Methode (s. z.B. [39]). Die rekursive Zerlegung des Gesamtgebietes über mehrere Ebenen ergibt einen *Substrukturierungsbaum*. Jeder Knoten des Baums entspricht einem Teilgebiet, die Blätter entsprechen den Finiten Elementen. Während man bei der Standard-FEM nur zwei Ebenen hat (das Gesamtsystem und die Finiten Elemente), erhält man hier eine hierarchische Darstellung des Problems in mehreren Detaillierungsstufen. Dies bringt einige Vorteile für die numerische Simulation mit sich:

- Die Denkweise des Ingenieurs wird in der Modellierung bereits berücksichtigt.

- Der Bedarf an Gleitpunktoperationen und Speicher wird im Vergleich zur Standard-FEM reduziert.

- Eine grobgranulare Parallelisierung der Konstruktion und Berechnung in einem Rechnernetz ist möglich.

- Die Mehrfachverwendung von Bauteilen sowie die Wiederverwendung von unverfeinerten Teilgebieten in einem adaptiven Verfahren ergeben zusätzliche Einsparungen.

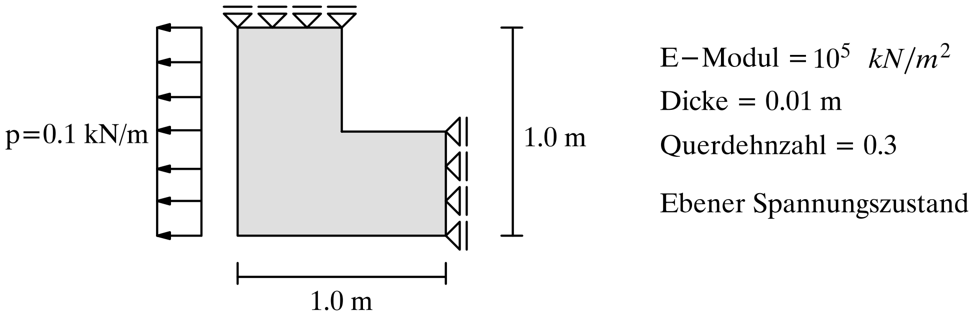


Abbildung 5: Statisches Modell einer Winkelscheibe unter Zug

3.2 Das Prinzip der rekursiven Substrukturierung in der FEM

Im folgenden wird das Prinzip der rekursiven Substrukturierung in seiner Anwendung auf die Finite-Element-Berechnung etwas genauer betrachtet. Wie bei der Standard-FEM läßt sich der Algorithmus in mehrere Teilschritte gliedern.

Schritt 1_{RS}: Zerlegung und Verfeinerung des Berechnungsgebiets

Schritt 2_{RS}: Erstellung der Knotengleichungssysteme

Schritt 3_{RS}: Lösung der Knotengleichungssysteme

Schritt 4_{RS}: Berechnung abgeleiteter Größen und Fehlerschätzung.

3.2.1 Zerlegung und Verfeinerung des Berechnungsgebiets

Zunächst stellt sich die Frage, wie man ein technisches Gebilde modellieren kann. Bei der numerischen Analyse eines technischen Objekts ist als erster Schritt eine Beschreibung dieses Objekts zu erstellen. Dabei sind die Geometrie des Objekts, seine Eigenschaften sowie alle auf das Objekt einwirkenden Einflüsse zu beachten.

Ein Objekt wird durch Daten bezüglich seines Inneren und Daten bezüglich seines Randes charakterisiert. Erstere beschreiben die physikalischen und mechanischen Eigenschaften und innere Kräfte. Darunter fallen die Materialeigenschaften, der Problemtyp, wie z.B. Wärmeleitungs-, Scheiben- oder Plattenprobleme, und sonstige charakteristische Größen, die je nach Problemtyp unterschiedlich sind. Unter Randdaten versteht man auf das Objekt einwirkende Kräfte und Einschränkungen in der Bewegungsfreiheit. Im mathematischen Sinn handelt es sich dabei um Neumann- und Dirichlet-Randbedingungen . In Abb. 5 ist das statische Modell einer Winkelscheibe unter Zug zu sehen. Darin erkennt man die geometrischen Abmessungen, die Materialparameter und die Randbedingungen. Am oberen und am rechten Rand ist die Verschiebung normal zur Kantenrichtung unterdrückt, die Verschiebung entlang der Kante ist frei. Am linken Rand wirkt eine über die Kante konstante Zugkraft.

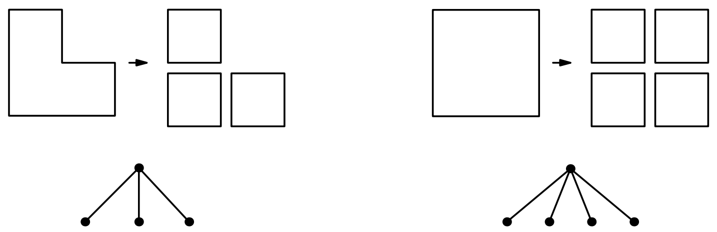


Abbildung 6: (a) Zerlegung eines Berechnungsgebiets in bedarfsterminale Bauteile (b) Zerlegung eines bedarfsterminalen Bauteils

Ausgehend von dieser groben Beschreibung eines größeren technischen Gebildes, kann nun ein Ingenieur eine Aufteilung in mehrere zu berechnende Teilobjekte vornehmen. Durch eine genaue Spezifikation der Zerlegung und eine Definition der Schnittstellen kann man die Teilobjekte separat rechnen, anschließend zusammensetzen und schließlich eine Lösung für das gesamte technische Gebilde ermitteln. Diese Zerlegung eines technischen Objekts entspricht einem Substrukturierungsbaum und läßt sich mit Hilfe einer kontextfreien Grammatik beschreiben. Die problemspezifischen Eigenschaften und die äußeren Einflüsse reicht man in Form von Attributen im Baum weiter. In [63] ist die Modellierung eines technischen Gebildes basierend auf der rekursiven Substrukturierung und die Durchführung einer adaptiven Berechnung auf diesem Modell formal beschrieben. Hier werden nur die wesentlichen Prinzipien zur rekursiven Zerlegung eines Objekts in Bauteile erläutert.

Man unterscheidet drei Arten von Bauteilen: *terminale*, *nichtterminale* und *bedarfsterminale* Bauteile. Ein *terminales Bauteil* wird auf ein Finites Element abgebildet und entspricht einem Blatt des Substrukturierungsbaumes. Die *nichtterminalen Bauteile* setzen sich aus mehreren Bauteilen zusammen und bilden die Knoten des Substrukturierungsbaumes. *Bedarfsterminale Bauteile* schließlich werden entweder auf Finite Elemente abgebildet oder setzen sich aus mehreren Bauteilen zusammen. Sie ermöglichen die Adaptivität einer Berechnung. Auf die terminalen Bauteile kann man verzichten, wenn man, wie in unserem Fall, bedarfsterminale Bauteile verwendet, die wie Finite Elemente geformt sind.

Wir betrachten nun die Modellierung und Verfeinerung eines Berechnungsgebiets. Ein oder mehrere Ingenieure spezifizieren rekursiv die Zerlegung des Gebiets in bedarfsterminale Bauteile. Diese Zerlegung bildet die Diskretisierung für den ersten Berechnungsschritt. Während einer adaptiven Berechnung werden bedarfsterminale Bauteile selbständig zerlegt, d.h. ohne interaktiver Steuerung durch den Ingenieur. Dazu wird in der Bauteilbeschreibung der bedarfsterminalen Bauteile festgelegt, wie sie in Teilbauteile zerlegt werden.

In Abb. 6 sieht man die Zerlegung der Winkelscheibe in drei bedarfsterminale Bauteile und die Zerlegung eines bedarfsterminalen Bauteils vom Typ Viereck in vier bedarfsterminale Bauteile vom Typ Viereck. Darunter befindet sich für beide Zerlegungen der entsprechende Substrukturbaum.

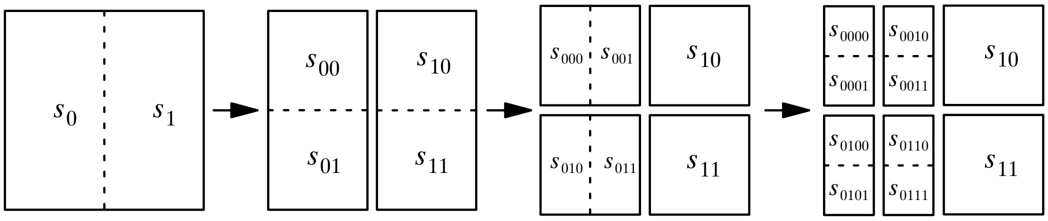


Abbildung 7: Adaptive Zerlegung eines Vierecks mittels Bisektion

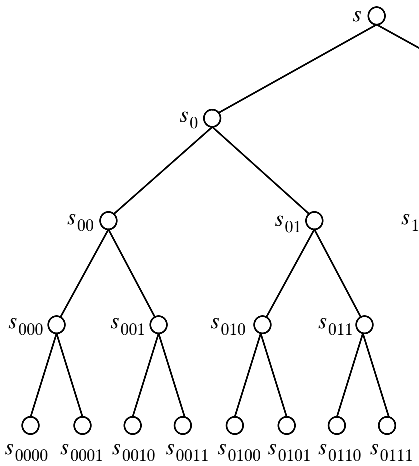


Abbildung 8: Substrukturierungsbaum für Zerlegung mittels Bisektion

Für die Zerlegung eines bedarfsterminalen Bauteils gibt es verschiedene Teilungsstrategien. Entscheidend ist, daß die entstehenden Teilbauteile von einem vorgegebenen Typ sind, für den eine Bauteilbeschreibung vorliegt, so daß eine weitere rekursive Teilung möglich ist. Am Beispiel eines Bauteils vom Typ Viereck betrachten wir zwei mögliche Zerteilungstrategien: zwei aufeinanderfolgende Halbierungen (Strategie *Bisektion*) (s. Abb. 7) bzw. eine Viertelung des Teilgebiets (Strategie *Viertelung*) (s. Abb. 9). Hier erkennt man ein wesentliches Prinzip der rekursiven Substrukturierung. Mit einem oder mehreren Separatoren wird ein Gebiet in Teilgebiete zerlegt, die voneinander unabhängig sind und nur über die Separatoren kommunizieren.

Die erste Strategie teilt ein Viereck mit einem vertikalen Separator (gestrichelt dargestellt) durch zwei gegenüberliegenden Seitenmittelpunkte in zwei Vierecke. Bei einer weiteren Anwendung der Teilungsstruktur mit horizontalen Separatoren erhält man eine Zerlegung des ursprünglichen Vierecks in vier Vierecke. In Abb. 7 wird diese Teilungsstrategie auf ein Viereck und anschließend auf zwei der neu entstandenen Vierecke angewandt. Intern wird diese adaptive Zerlegung dargestellt durch einen mehrstufigen Binärbaum (Abb. 8).

Alternativ dazu gibt es auch eine Teilungsstrategie, die eine Zerlegung eines Vierecks in vier Teilgebiete in einem Schritt erledigt. Durch Verwendung von

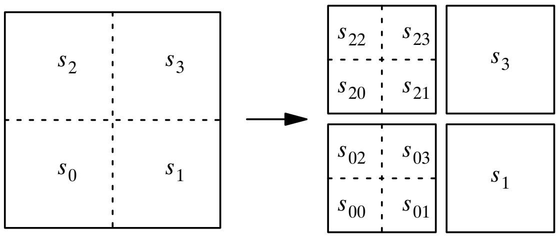


Abbildung 9: Adaptive Zerlegung eines Vierecks durch Viertelung

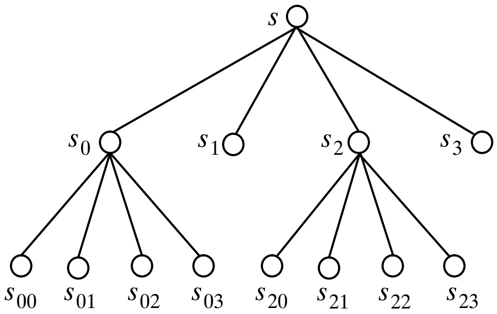


Abbildung 10: Substrukturierungsbaum für Zerlegung durch Viertelung

zwei senkrecht aufeinander stehenden Separatoren wird ein Baumknoten mit vier Söhnen erzeugt. In den Abbildungen 9 und 10 sind eine mehrfache Zerlegung durch Viertelung und der zugehörige Substrukturierungsbaum dargestellt.

Beide Strategien führen zum selben FE-Netz. Ihre unterschiedliche Repräsentation als Substrukturierungsbaum hat aber Auswirkungen auf die Berechnung des technischen Objekts. Bei der Bisektion erhält man einen tieferen Baum mit mehr Knoten, was im Verlauf der Berechnung zu mehr Datentransporten zwischen den Knoten und zu stärkerer Datenredundanz führt (s. Kap. 7). Dafür sind hier die Separatoren im Vergleich zur Strategie der Viertelung kürzer und damit der Aufwand zur Bearbeitung eines Knotens geringer.

Mit der baumartigen Zerlegung von Objekten mit genau definierten Schnittstellen ist die Voraussetzung für eine adaptive Modellierung und Simulation gegeben. Sie unterstützt die schnittstellenorientierte Sichtweise der Ingenieure bei der Modellierung. Jeder Knoten im Substrukturierungsbaum besteht aus der Beschreibung seiner nach außen sichtbaren Schnittstelle und seiner inneren Struktur. In der weiteren Arbeit werden diese beiden Bestandteile auch als *Gebietsrand* und *Gebietsinneres* bezeichnet. Sämtliche Daten auf dem Gebietsrand sind auf der nächsthöheren Baumebene sichtbar. Die interne Struktur eines Knotens, d.h. seine Zerlegung in weitere Teilobjekte, ist dagegen in seinem Vaterknoten nicht erkennbar. Dies entspricht der lokalen Sichtweise der Ingenieure beim Konstruktionsprozeß, die in Kapitel 3.1 dargestellt wurde. Ein Baumknoten hat alle Informationen über die Gebietsränder seiner Söhne und die Information, um die Schnittstellen zusammenzufügen. Er hat aber kein Wissen über die Zerlegungs-

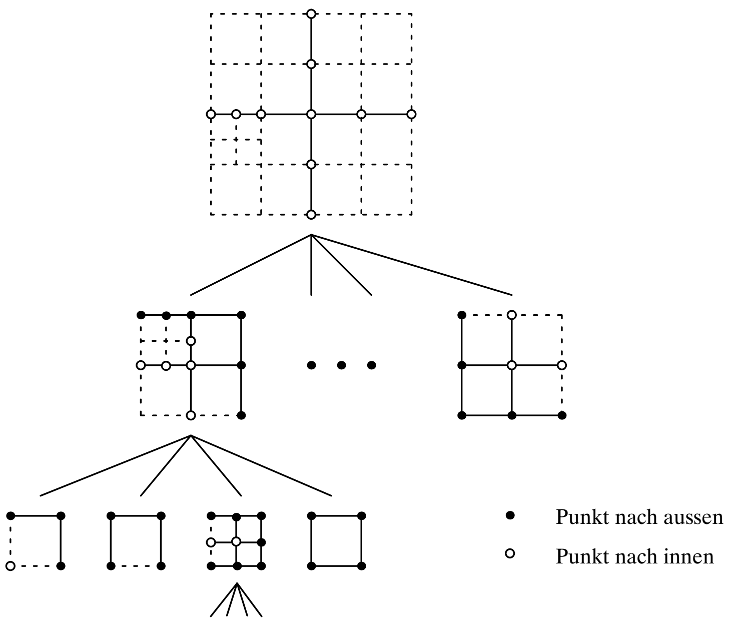


Abbildung 11: Mehrstufige Zerlegung eines Quadrats

struktur seiner Söhne.

In Abb. 11 sieht man in einer mehrstufigen Zerlegung eines Quadrates. Die für den jeweiligen Baumknoten verdeckte Zerlegungsstruktur der Söhne und deren Nachfolger ist durch gestrichelte Separatoren dargestellt. Die gestrichelte Darstellung des Gesamtgebietsrand deutet an, daß die Randbedingungen ebenfalls in den unteren Baumebenen behandelt werden (s. Kap. 3.2.2). Außerdem erkennt man, welche Stützpunkte in den verschiedenen Baumknoten sichtbar sind.

Dabei werden als *Stützpunkte* die Knoten der Finiten Elemente bezeichnet. Jedem Stützpunkt wird eine Menge von Freiheitsgraden zugeordnet, die vom Problemtyp abhängt. Ein Stützpunkt entspricht einem Gitterpunkt in der Standard-FEM. In der weiteren Arbeit wird aber der Begriff Stützpunkt bevorzugt, da in der schnittstellenorientierten Sichtweise in einem Baumknoten das zugrunde liegende FE-Netz nicht sichtbar ist, sondern lediglich die Freiheitsgrade auf den Separatoren und den Teilgebietsrändern.

Hier zeigt sich ein wesentlicher Unterschied zur Standard-FEM, bei der ein zentrales Bild von der Vernetzung des technischen Gebildes existiert. In diesem globalen System ist eine Identifizierung der Gitterpunkte der Elemente, die Erkennung von inkompatiblen Gitterpunkten und die Behandlung der Randbedingungen möglich. In der rekursiven Substrukturierung dagegen gibt es kein zentrales Bild der Vernetzung, sondern einen Baum von lokalen Sichtweisen unterschiedlicher Detaillierungsstufen. Dadurch ist eine Verarbeitung der Stützpunkte und der Randbedingungen dezentral in den Baumknoten möglich.

Die rekursive Zerteilung von Objekten bewirkt also eine Aufspaltung in lokale Teilsysteme und bietet eine Möglichkeit, Parallelität in eine FE-Berechnung einzuführen. Abschließend ist noch festzustellen, daß Schritt 1_{RS} in einem top-down-Baumdurchlauf erfolgt, d.h. im Wurzelknoten startet und in den Blattknoten endet.

3.2.2 Erstellung der Knotengleichungssysteme

Nach der Zerlegung eines Randwertproblems in einen Substrukturierungsbaum, sind nun für diesen Baum lineare Gleichungssysteme aufzustellen. Im Gegensatz zu Schritt 1_{RS} wird in Schritt 2_{RS} der Baum in bottom-up Reihenfolge durchlaufen. Die Erstellung eines linearen Gleichungssystems (LGS) für einen Baumknoten erfolgt dabei in vier Teilschritten:

(a) Aufstellung von Matrix und Lastvektor.

(b) Trennung von inneren und äußeren Freiheitsgraden.

(c) Behandlung der Randbedingungen und der inkompatiblen Punkte.

(d) Statische Kondensation.

Da zwischen den Blättern und den sonstigen Knoten des Substrukturierungsbaumes gewisse Unterschiede gelten, betrachten wir zuerst die Erstellung eines Blattgleichungssystem.

Erstellung eines Blattgleichungssystems

Die Blätter des Substrukturierungsbaumes sind ein Sonderfall, da sie Finiten Elementen entsprechen. Mit den Informationen über den Problemtyp, den geometrischen und physikalischen Eigenschaften des Elements, kann man ein lineares Gleichungssystem aufstellen. Dieses LGS enthält die gesamten strukturellen Eigenschaften dieses Elements, die für das Gesamtsystem von Bedeutung sind. Das LGS hat folgende Form:

$$K_e u_e = f_e \tag{14}$$

wobei K_e die *Steifigkeitsmatrix*, u_e der *Verschiebungsvektor* und f_e der *Lastvektor* des Elements ist.

Hier liegt eine Schnittstelle zu externen Programmen oder internen Routinen vor, die aus der Elementbeschreibung und der gegebenen Problemklasse die Elementmatrizen liefern können. Der Lastvektor ist abhängig von den Randbedingungen und wird nach der Trennung der Freiheitsgrade erstellt.

Im Teilschritt (b) werden die n_K Freiheitsgrade (FG) des Blattknotens in die Menge der *inneren Freiheitsgrade* FG_i und die Menge der *äußeren Freiheitsgrade* FG_a aufgeteilt. Innere Freiheitsgrade liegen entweder im Inneren des Elements oder sie liegen auf dem Elementrand und treten in keinem anderen Element auf, d.h. sie liegen auf dem Gesamtgebietsrand. Die restlichen Freiheitsgrade gehören zu den FG_a. In Abb. 11 sind die Mengen FG_i und FG_a für verschiedene Knoten eines Substrukturierungsbaumes dargestellt.

Nach dieser Zerlegung der FG in zwei disjunkte Mengen läßt sich das Elementgleichungssystem wie folgt darstellen:

$$\begin{pmatrix} K_{aa} & K_{ai} \\ K_{ia} & K_{ii} \end{pmatrix} \begin{pmatrix} u_a \\ u_i \end{pmatrix} = \begin{pmatrix} f_a \\ f_i \end{pmatrix} \tag{15}$$

Die Randbedingungen legt der Ingenieur bei der Beschreibung eines Randwertproblems für ein technisches Objekt fest. Sie werden im Schritt 1_{RS} rekursiv vom Vaterknoten zu den Sohnknoten weitergereicht und lassen sich daher dezentral verarbeiten. Dies ist ein großer Vorteil des Verfahrens der rekursiven Substrukturierung.

Randbedingungen sind Vorschriften für die Verschiebungen und Kräfte eines Objekts. Solche Bedingungen können bei FE-Modellen auch für das Gebietsinnere gestellt werden. Wir betrachten aber in dieser Arbeit nur Randbedingungen, die auf dem geometrischen Rand des Gesamtgebietes liegen. Eine Erweiterung auf Randbedingungen im Gebietsinneren, wie z.B. innere Kräfte, ist aber möglich.

Es wird unterschieden zwischen Neumann- und Dirichlet- Randbedingungen. Auf den Blattknoten werden sämtliche Neumann-Randbedingungen, die äußeren angebrachten Lasten entsprechen, in das Gleichungssystem integriert. Da die Lastfunktion auf Schnittstellen i.a. eine Last pro Längen- oder Flächeneinheit beschreibt, muß man zuerst für jeden Stützpunkt auf dieser Schnittstelle den zugehörigen Lastwert ermitteln. Dann wird für jeden Stützpunkt, auf dem eine Last spezifiziert ist, der Lastwert auf die entsprechende Komponente des Lastvektors addiert. Die Verarbeitung von Dirichlet-Randbedingungen ist nur bei Stützpunkten möglich, die der Menge FG_i angehören, und wird bei den Baumknoten genauer erläutert.

Im letzten Teilschritt wird das Gleichungssystem umgeformt. Man erzeugt die Information auf dem Teilgebiets- bzw. Elementrand K_{aa} und f_a, die für die Erstellung des Knotengleichungssystems auf der nächsten Ebenen des Substrukturierungsbaumes benötigt wird. Das LGS (15) läßt sich in zwei Teilgleichungssysteme (16) und (17) zerlegen.

$$K_{aa}u_a + K_{ai}u_i = f_a \tag{16}$$

$$K_{ia}u_a + K_{ii}u_i = f_i \tag{17}$$

Durch Umformung von (17) zu

$$u_i = K_{ii}^{-1}(f_i - K_{ia}u_a) \tag{18}$$

und Einsetzten in (16) ergibt sich das Gleichungssystem

$$(K_{aa} - K_{ai}K_{ii}^{-1}K_{ia})u_a = f_a - K_{ai}K_{ii}^{-1}f_i \tag{19}$$

das exakt und ohne Kenntnis der Unbekannten FG_i im Gebietsinneren gelöst werden kann. Der Vorgang der Erstellung des Gleichungssystems wird als *statische Kondensation* bezeichnet. Dabei wird die Strukturinformation der Punkte im Gebietsinneren auf die Punkte am Gebietsrand des Elements übertragen.

Nachdem das LGS eines Blattes wie eben beschrieben umgeformt ist, wird die neue Matrix K_{aa} und der neue Lastvektor f_a an den Vaterknoten weitergereicht, um dort den Schritt 2_{RS} ausführen zu können.

Erstellung eines Knotengleichungssystems

Bei einem Baumknoten ändert sich im Vergleich zu einem Blatt der erste Teilschritt, weil hier die Matrix nicht über eine sogenannte Elementroutine berechenbar ist. Stattdessen muß man die Matrizen aus den Informationen zu den externen Schnittstellen der Sohnknoten zusammensetzen. Die Schritte (b) und (d) werden analog zu den Blättern durchgeführt, der Schritt (c) erfordert andere Umformungen.

In Schritt (a) setzt man die Matrizen K_{aa} und die Vektoren f_a der Söhne, die die Kopplung der Freiheitsgrade an den nach außen sichtbaren Rändern der Teilgebiete beschreiben, entsprechend der Zerteilungsinformation aus Schritt 1 zusammen. Dabei werden einander entsprechende Punkte identifiziert und ihre Einträge aus den Matrizen und Lastvektoren der Sohnknoten in das Gleichungssystem des Vaterknotens addiert. Dieser Vorgang, auch *Assemblierung* genannt, wird durch eine bei der Zerlegung gewonnene baumartige Datenstruktur ermöglicht. Diese Datenstrukturen, sogenannte *Schnittstellenbäume*, wurden in [63] entwickelt und ermöglichen eine adaptive, rekursive Zerlegung von technischen Objekten. Wie man einen Baumknoten aus zwei Sohnknoten zusammengesetzt und dabei die Freiheitsgrade auf den Schnittstellen miteinander identifiziert, ist in Abb. 12 dargestellt. Dort werden zwei Kanten zusammengesetzt, die eine unterschiedliche Verteilung der Stützpunkte besitzen. Die Darstellung der Stützpunkte in einem Schnittstellenbaum ordnet einer Kante den Mittelpunkt sowie eine linke und rechte Teilkante zu. Die rekursive Anwendung dieses Prinzips auf die Teilkanten führt zu einem Baum, dessen terminale Elemente Teilkanten sind, die keine Stützpunkte mehr enthalten und im Baum mit N als Nullkanten gekennzeichnet sind. Beim Zusammensetzen der Kanten sind die beiden Schnittstellenbäume zu addieren (s. [63]) und ergeben einen neuen Schnittstellenbaum für den Separator der beiden Teilgebiete.

Behandlung der inkompatiblen Punkte

Im Umformungsschritt (c) sind die inkompatiblen Stützpunkte zu behandeln. Inkompatible Punkte entstehen, wenn zwei Teilgebiete aufeinander treffen, die entlang ihrer gemeinsamen Grenzstruktur (d.h. des entsprechenden Separators ihres Vaterknotens) unterschiedlich vernetzt sind. Die Entstehung von inkompa-

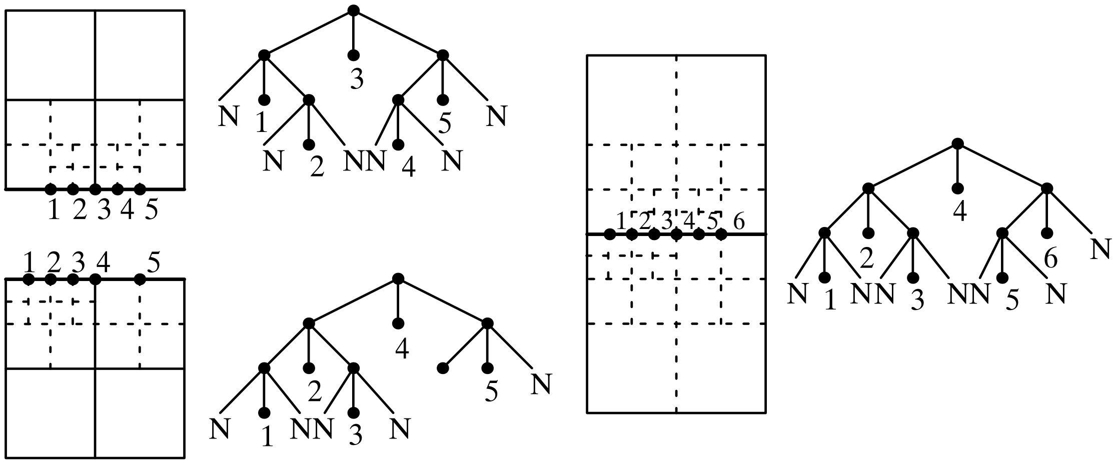


Abbildung 12: Zusammensetzen von Schnittstellen und der zugehörigen Schnittstellenbäume

tiblen Punkten ist bei einer adaptiven Vernetzung nicht zu vermeiden (s. Abb. 3). Ohne Sonderbehandlung bewirken sie eine Unstetigkeit der Lösung und damit einen schwer abzuschätzenden Berechnungsfehler. Ihre Behandlung ist abhängig von der Wahl der Basisfunktionen. Während in [63] nur der Fall von bilinearen Ansatzfunktionen und Knotenbasen behandelt wurde, ist in dieser Arbeit auch die Verwendung von biquadratischen Ansatzfunktionen sowie erstmalig die Kombination von rekursiver Substrukturierung und hierarchischen Basisfunktionen, sowohl für den bilinearen als auch biquadratischen Fall, realisiert worden. Auf die Verwendung von biquadratischen Ansatzfunktionen und die Hierarchisierung wird im Kapitel 5 genauer eingegangen.

Hier wird nur kurz die Behandlung der inkompatiblen Punkte bei bilinearen Knotenbasen anhand eines Beispiels beschrieben. Eine formalere Beschreibung der Umformung erfolgt in Kapitel 5.5. Zu jedem inkompatiblen Punkt muß man zwei hierarchische Nachbarpunkte bestimmen. Die Existenz solcher Nachbarpunkte wurde in [63] nachgewiesen. In Abb. 13 ist ein Baumknoten mit drei inkompatiblen Punkten p_1, p_4 und p_5 zu sehen. Alle anderen inkompatiblen Punkte des FE-Netzes sind bereits in den Teilgebieten eine Ebene tiefer behandelt worden. Der Lösungswert des inkompatiblen Punktes p_1 wird als Mittelwert der Freiheitsgrade der hierarchischen Nachbarpunkte p_2 und p_3 festgelegt.

$$u(p_1) = \frac{1}{2}(u(p_2) + u(p_3)) \tag{20}$$

Die entsprechende Umformung des Knotengleichungssystem ist die Addition der Zeile und Spalte des inkompatiblen Punktes zu gleichen Teilen (d.h. mit dem Faktor 1/2) auf die Zeilen und Spalten der hierarchischen Nachbarpunkte. Analog wird mit der rechten Seite verfahren. Anschließend wird der inkompatible Punkt aus dem Gleichungssystem entkoppelt, d.h. die Zeile und Spalte des Punktes sind

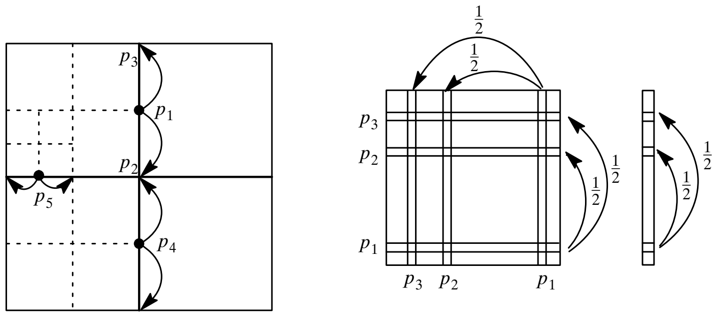


Abbildung 13: Behandlung der inkompatiblen Punkte: FE-Netz und lineares Gleichungssystem

zu streichen. Im rechten Teil der Abb. 13 sind diese Umformungen an der Matrix dargestellt.

Der Aufwand dieser Umformung fällt im Vergleich zu den anderen Umformungen kaum ins Gewicht, da die Anzahl der inkompatiblen Stützpunkte stets viel kleiner als die der kompatiblen ist [63].

Behandlung der Randbedingungen

Nun folgt die Integration der Randbedingungen. Während die Neumann-Randbedingungen schon in den Blattknoten behandelt wurden, sind nun die Dirichlet-Randbedingungen zu bearbeiten. Deren Verarbeitung ist im allgemeinen in den Blättern nicht möglich. Um die Stetigkeit der Lösung zu garantieren, kann man eine Randbedingung für einen Randpunkt erst anwenden, wenn in der Steifigkeitsmatrix und im Lastvektor alle Anteile aufgesammelt sind, die den Randpunkt betreffen. Das bedeutet, daß nur Punkte behandelt werden können, für die alle Teilgebiete, die sie enthalten, bereits verarbeitet sind. Solche Punkte werden wie bei den Blättern der Menge FG_i zugeordnet und nicht mehr an den Vaterknoten weitergegeben. Dadurch verhindert man eine mehrfache Aufbringung von Randbedingungen.

Abb. 14 zeigt ein adaptiv verfeinertes Quadrat, auf dessen linker Kante eine Dirichlet-Randbedingung vorliegt, und den zugehörigen Substrukturierungsbaum. Im rechten Teil der Abbildung ist verdeutlicht, in welcher Baumebene die Randbedingungen auf die einzelnen Stützpunkte der Kante aufgebracht werden, wobei hier die Ebene 0 dem Wurzelknoten und die Ebene 3 den Blattknoten entspricht.

Die Aufbringung einer Randbedingung auf einen Freiheitsgrad des Knotengleichungssystems erfolgt analog zu dem Vorgehen in der Standard-FEM. Am Beispiel eines Randpunktes, der i-ten Unbekannten im Verschiebungsvektor, mit dem vorgeschriebenen Randwert RW erhält man folgende Umformung des Gleichungssystems. Die Spalte i der Matrix wird mit RW multipliziert und von der rechten Seite abgezogen. Anschließend wird der Punkt i entkoppelt, indem sei-

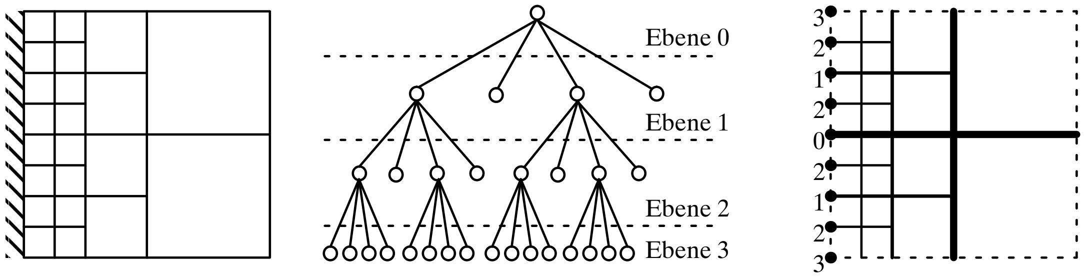


Abbildung 14: Behandlung der Dirichlet-Randbedingungen

ne nichtdiagonalen Einträge gelöscht, das Diagonalelement auf 1 gesetzt und der Randwert RW auf der rechten Seite eingetragen wird.

Alternativ dazu kann man sämtliche Randbedingungen im Wurzelknoten behandeln. Diese Vorgehensweise führt zur identischen Lösung, ist aber weniger effizient. Durch das dezentrale Aufbringen der Randbedingungen verkleinern sich die Knotengleichungssysteme zum frühest möglichen Zeitpunkt. Dadurch wird eine Ersparnis an Speicher und Rechenaufwand erzielt gegenüber der Behandlung der Randbedingungen im Wurzelknoten.

Statische Kondensation

Da die statische Kondensation der aufwendigste Berechnungsschritt ist, werden im folgenden die einzelnen Teilschritte etwas genauer beschrieben. Denn nur bei einer geschickten Realisierung der notwendigen Operationen kann man den vollen Gewinn bezüglich Speicherbedarf und Rechenaufwand gegenüber der Standard-FEM erzielen. Hinter jedem Teilschritt ist die Anzahl der benötigten Operationen unter Berücksichtigung der Symmetrie der Matrizen angegeben. [1] Unter einer Operation wird eine Einheit aus Addition und Multiplikation gemeinsam verstanden. Dies entspricht den Architekturmerkmalen moderner RISC-Prozessoren, die diese Operationen häufig gleichzeitig ausführen können. Bei den Operationszahlen und im weiteren Text bedeutet

- n_a die Anzahl äußeren Freiheitsgrade
- n_i die Anzahl der inneren Freiheitsgrade.

Um eine explizite Invertierung der Matrix K_{ii} zu vermeiden, wird diese mit Hilfe einer Cholesky-Zerlegung [67] in das Produkt zweier Dreiecksmatrizen aufgespalten:

$$K_{ii} = L_{ii} \cdot L_{ii}^T \qquad \text{mit:} \quad \forall k > j: \quad L_{ii}[j,k] = 0 \qquad (n_i^3/6) \tag{21}$$

[1] In den in dieser Arbeit betrachteten Problemstellungen sind die Steifigkeitsmatrizen symmetrisch. Dies wird bei den Matrixtransformationen und der Speicherung berücksichtigt. Eine Erweiterung auf unsymmetrische Steifigkeitsmatrizen ist leicht möglich.

Die Auflösung der Gleichung $L_{ii}(L_{ii}^T \cdot K'_{ia}) = K_{ia}$ durch Vorwärts-/Rückwärtssubstitution ergibt:

$$K'_{ia} := K_{ii}^{-1} \cdot K_{ia} \qquad (n_a \cdot n_i^2) \tag{22}$$

Damit kann man die neue Teilmatrix K'_{aa} für die Kopplungen auf dem Teilgebietsrand bestimmen.

$$K'_{aa} := K'_{ia} - K_{ai} \cdot K'_{ia} \qquad ((n_i \cdot n_a^2)/2) \tag{23}$$

Die Auflösung der Gleichung $L_{ii}(L_{ii}^T \cdot f'_i) = f_i$ unter der Verwendung der Cholesky-Zerlegung ergibt

$$f'_i := K_{ii}^{-1} \cdot f_i \qquad (n_i^2), \tag{24}$$

womit man den neuen Kraftvektor für den Teilgebietsrand f'_a bestimmen kann.

$$f'_a := f_a - K_{ai} \cdot f'_i \qquad (n_i \cdot n_a) \tag{25}$$

Nach diesen Umformungen erhält das LGS des Baumknotens eine neue Form. Zusätzlich zu den oben beschriebenen Modifikationen auf den Teilen der Matrizen und der rechten Seite, die für die Kopplung der Teilgebietsränder verantwortlich sind, werden auch Zwischenergebnisse der Umformungen auf den restlichen Teilen zwischengespeichert. Dabei handelt es sich um die Cholesky-Zerlegung der Matrix K_{ii} und der Produkte von $K_{ii}^{-1} \cdot K_{ia}$ und $K_{ii}^{-1} \cdot f_i$, die bei der Berechnung der Lösung für die Unbekannten im Gebietsinneren u_i verwendet werden. Damit hat die Matrix und die rechte Seite des Gleichungssystems (15) eines Baumknotens nach Teilschritt (d) von Schritt 2_{RS} folgende Gestalt:

$$\begin{pmatrix} K_{aa} - K_{ai} \cdot K_{ii}^{-1} \cdot K_{ia} & 0 \\ K_{ii}^{-1} \cdot K_{ia} & L_{ii} \end{pmatrix} \qquad \begin{pmatrix} f_a - K_{ai} \cdot K_{ii}^{-1} \cdot f_i \\ K_{ii}^{-1} \cdot f_i \end{pmatrix} \tag{26}$$

Es sei nochmals darauf hingewiesen, daß die unteren n_i Zeilen von (26) keine algebraische Bedeutung besitzen, sondern lediglich der Zwischenspeicherung dienen.

3.2.3 Lösung der Gleichungssysteme

Im Schritt 2_{RS} wurde ausgehend von den Elementgleichungssystemen in den Blattknoten mit Hilfe der Zerlegungsinformation aus Schritt 1_{RS} für jeden Knoten des Substrukturierungsbaumes ein LGS erstellt. Die Gesamtheit der LGS aller Knoten ergibt ein System von LGS, dessen Lösung die Antwort des zu berechnenden Objekts unter den auf ihn einwirkenden Randbedingungen darstellt. Diese

Lösung wird in einem erneuten top-down-Durchlauf des Substrukturierungsbaumes berechnet.

Begonnen wird die Lösung dieser Gleichungssysteme bei der Wurzel des Substrukturierungsbaumes, die gemäß der Definition aus Kapitel 3.2.2 nur innere Freiheitsgrade besitzt. In der Wurzel wurde in Schritt 2_{RS} wie bei allen Baumknoten die Assemblierung, die Behandlung der Randbedingungen und der inkompatiblen Punkte sowie die Cholesky-Zerlegung durchgeführt. Da keine äußeren Freiheitsgrade existieren, kann der Vektor der Unbekannten unmittelbar bestimmt werden. Die Lösung der Wurzel ist die Lösung auf den Teilgebietsrändern ihrer Söhne und wird an die Söhne weitergegeben.

Jeder Sohnknoten bekommt von seinem Vaterknoten die Werte auf seinem Rand, d.h. u_a. Damit kann er die Lösung für seine inneren Freiheitsgrade berechnen. Dieser Vorgang nennt sich *Rückwärtssubstitution* und läßt sich wie folgt beschreiben.

$$u_i := K_{ii}^{-1} \cdot f_i - K_{ii}^{-1} \cdot K_{ia} \cdot u_a \tag{27}$$

Anschließend werden wiederum die berechneten Werte an die entsprechenden Söhne geschickt. Dieser Vorgang wird bis in die Blattknoten fortgesetzt.

Eine Sonderbehandlung erfordern wieder die inkompatiblen Punkte. Nach der Berechnung der Unbekannten eines Baumknotens, sind die Werte der inkompatiblen Punkte aus den angrenzenden kompatiblen hierarchischen Nachbarpunkten zu interpolieren (s. Gleichung (20). Diese Interpolation wird bei der Verwendung von hierarchischen Basen automatisch erledigt (s. Kap. 5.5) und ist nicht explizit durchzuführen.

Durch die Abspeicherung der Ergebnisse der Umformungen in Schritt 2_{RS} sind $n_i \cdot n_a$ Gleitpunktoperationen für die Berechnung von (27) nötig und es entsteht kein zusätzlicher Speicherbedarf.

3.2.4 Berechnung abgeleiteter Größen und Fehlerschätzung

Am Ende von Schritt 3_{RS} liegt in allen Knoten des Substrukturierungsbaumes die korrekte Lösung für die Unbekannten vor. Bei der Beurteilung der Belastbarkeit eines technischen Objekts sind Ingenieure aber, wie in Kapitel 2.2 bereits bemerkt, auch an weiteren Größen interessiert, die aus den Verschiebungen abgeleitet werden können. Bei der ebenen Elastizität handelt es sich dabei um die Schnittkräfte, z.B. Spannungen, Verzerrungen, Momente und Querkräfte.

Die Ermittlung der Schnittkräfte erfolgt bei linearen Finite-Element-Problemstellungen durch die lineare Abbildung (9) mit Hilfe der Verzerrungs-Verschiebungs-Matrix $D_{\varepsilon u}$ und der Elastizitätsmatrix C. Somit kann die Berechnung der abgeleiteten Größen durch eine Matrix-Vektor-Multiplikation erreicht werden.

$$\sigma = S_k \cdot u \tag{28}$$

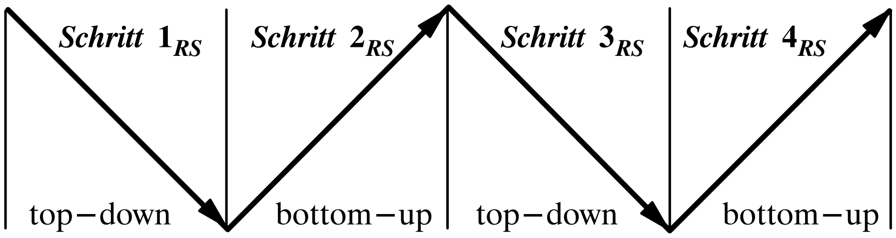


Abbildung 15: Darstellung der rekursiven Substrukturierung in Form eines W-Zyklus

Die *Abbildungsmatrix zur Spannungstransformation* S_k ermittelt man wie die Elementmatrizen abhängig vom Problemtyp über externe oder interne Schnittstellen (s. Kap. 3.2.6). Dieser Teilschritt findet nur in den Blättern des Substrukturierungsbaumes, den Finiten Elementen, statt. In den Zwischenknoten ist keine Berechnung der Spannungswerte erforderlich.

Anschließend werden sämtliche berechneten Ergebnisse sowie die topologischen Informationen über das Berechnungsobjekt in einem bottom-up-Baumdurchlauf von den Blättern bis zum Wurzelknoten aufgesammelt und an eine Routine zur Fehlerschätzung oder einen Postprozessor übergeben. Da es sich um die gleiche tabellarische Darstellung eines Berechnungslaufs wie bei der Standard-FEM handelt, können die gleichen Schnittstellen verwendet werden.

Bei den in Kapitel 2.3.1 beschriebenen Fehlerschätzern für Elastizitätsprobleme benötigt man Vergleichsdaten aus benachbarten Elementen zur Ermittlung von Spannungssprüngen oder geglätteten Spannungswerten. Deswegen ist eine globale Sichtweise der Berechnungsergebnisse notwendig, und die Fehlerschätzung kann nur im Wurzelknoten stattfinden. Wenn dagegen elementlokale Fehlerkriterien, wie z.B. Elementresiduen oder hierarchische Überschüsse verwendet werden, kann die Fehlerschätzung bereits in den Blättern des Baums, unmittelbar nach der Berechnung der abgeleiteten Größen erfolgen. In diesem Fall kann man auf die tabellarische Darstellung der Ergebnisse verzichten. Es genügt eine Liste der zu verfeinernden Elemente nach oben zu geben. Die rekursive Substrukturierung besitzt hier als großen Vorteil die Möglichkeit, je nach Problemtyp, eine zentrale oder eine verteilte Fehlerschätzung einzusetzen.

3.2.5 Berechnungsablauf

Analog zur Standard-FEM (s. Abb. 1) kann der Berechnungsablauf der rekursiven Substrukturierung durch die Unterteilung des Algorithmus in Teilschritte in Form eines W-Zyklus dargestellt werden (s. Abb. 15). Hier ist die Graphik aber anders zu interpretieren. Bei der Standard-FEM werden die Teilschritte abwechselnd auf dem Gesamtgebiet mit globalen Datenstrukturen und auf Teilgebieten (Elementen) mit lokalen Datenstrukturen ausgeführt. In der rekursiven Substrukturierung erfolgen die Teilschritte abwechselnd in einem top-down- und einem bottom-up-Baumdurchlauf. So wird in Schritt 1_{RS} ausgehend von der Wurzel des Substrukturierungsbaumes die Zerlegungs- bzw. Verfeinerungsinformation bis in die Blattknoten weitergereicht, die dann rekursiv zu zerlegen sind. Schritt 2_{RS} startet in den Blattknoten durch Aufruf der Elementroutinen und assembliert und transformiert die Knotengleichungssysteme bis hinauf zum Wurzelknoten. Im nächsten top-down-Durchlauf wird in jedem Knoten die Lösung berechnet und an die Sohnknoten übergeben. Schließlich werden die abgeleiteten Größen in den Blattknoten ermittelt und alle Ergebnisse des Berechnungslaufs von Baumebene zu Baumebene bis zum Wurzelknoten aufgesammelt, wo eine Fehlerschätzung stattfinden kann. Falls ein weiterer Berechnungsschritt mit einem verfeinerten Netz notwendig ist, wird der W-Zyklus erneut in derselben Reihenfolge durchlaufen.

Im Vergleich mit dem W-Zyklus der Standard-FEM aus Abb. 1 entspricht Schritt 1_{RS} dem Schritt 1_{FE}. Der Berechnungsschritt 2_{RS} umfaßt den Schritt 2_{FE} und einen wesentlichen Teil der Lösung des Gleichungssystems in Schritt 3_{FE}. Den restlichen Teil von Schritt 3_{FE} erledigt Schritt 3_{RS}. Schließlich erfolgen in Schritt 4_{RS} die Bearbeitungsschritte 4_{FE} und 5_{FE}.

Es zeigt sich, daß das Prinzip der rekursiven Substrukturierung gut für ein adaptives Verfahren geeignet ist. Nach der Fehlerschätzung wird entschieden, ob die geforderte Genauigkeit erreicht ist und die Berechnung beendet wird oder ob eine Liste von Elementen ausgegeben wird, die zu verfeinern sind. Diese Liste wird in einem modifizierten Schritt 1_{RS} anhand der Substrukturierung in Teilgebietslisten aufgeteilt und an die entsprechenden Sohnknoten geschickt. In jedem Knoten teilt man die Verfeinerungsliste erneut und verschickt sie an die Söhne, bis sie schließlich in den Blattknoten angekommen ist, die rekursiv zu teilen sind. Dabei sendet man solche Verfeinerungslisten nur an Baumknoten, in deren Unterbäumen mindestens ein Blatt verfeinert wird. Nur in diesen Baumknoten, d.h. in allen Knoten, die auf einem Pfad von einem zu verfeinernden Blattknoten zum Wurzelknoten liegen, sind die Schritte 1_{RS} und 2_{RS} durchzuführen. In den anderen Baumknoten kann man die LGS vom letzten Berechnungsschritt übernehmen. Die Schritte 3_{RS} und 4_{RS} sind dagegen in allen Baumknoten zu vollziehen, da selbst eine lokale Verfeinerung der Diskretisierung sich auf die Lösungswerte aller Freiheitsgrade auswirkt. In den meisten Implementierungen der Standard-FEM ist bei einer lokalen Verfeinerung das gesamte Gleichungssystem neu zu assemblieren, da sonst die Bandstruktur zerstört wird. Lediglich die Elementmatrizen unverfeinerter Elemente kann man retten.

Das adaptive Verfahren, charakterisiert durch ein mehrfaches Durchlaufen des W-Zyklus, ist in Abb. 16 schematisch dargestellt.

Ein weiterer Vorteil des Berechnungsschema aus Abb. 15 ist die hohe Nebenläufigkeit der Teilschritte. Es gibt folgende drei Regeln bezüglich der Abhängigkeiten in der Bearbeitung der Knoten:

- Wenn ein Berechnungsschritt eines top-down-Durchlaufes in einem Knoten

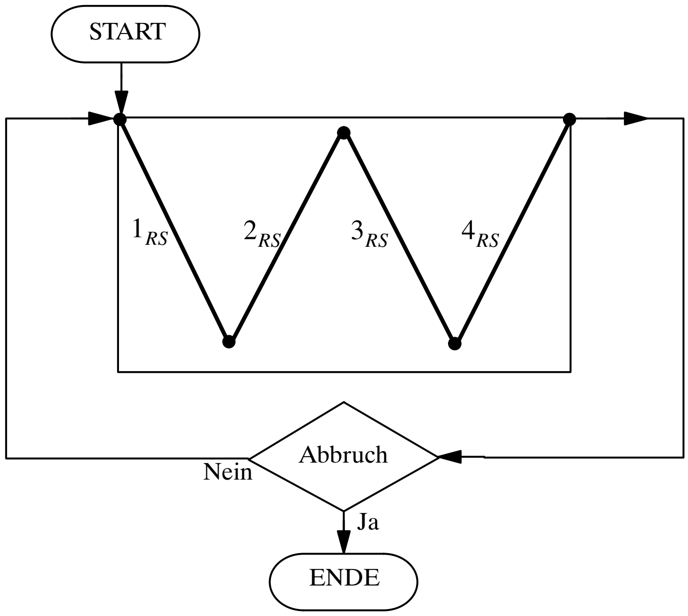


Abbildung 16: Adaptives Verfahren der FEM unter Verwendung der rekursiven Substrukturierung

beendet ist, können alle Söhne parallel und voneinander unabhängig diesen Schritt durchführen.

- Ein Berechnungsschritt eines bottom-up-Durchlaufes kann erst begonnen werden, wenn alle Sohnknoten diesen Schritt beendet haben.

- Erst wenn ein Berechnungsschritt in einem Knoten beendet ist, kann der nächste Berechnungsschritt in diesem Knoten begonnen werden.

Zusammenfassend läßt sich feststellen, daß sich die Methode der rekursiven Substrukturierung zur Implementierung eines Finite-Element-Programms für adaptive Berechnungen verschiedenster Problemstellungen auf allgemein geformten Berechnungsgebieten eignet. Problemspezifische Schnittstellen befinden sich nur an den Blattknoten und an der Wurzel des Substrukturierungsbaumes und können so implementiert werden, daß man bereits vorhandene FE-Software einbinden kann. Es wird sowohl eine rekursive Zerlegung des Objekts in Teilstrukturen durch den bearbeitenden Ingenieur als auch eine automatische Zerlegung im Rahmen einer adaptiven Verfeinerung unterstützt. Die rekursive Zerlegung in nebenläufige Teilprobleme ermöglicht eine Parallelisierung der Berechnung.

3.2.6 Realisierung im Rahmen eines modularen Programmsystems

Zur praktischen Durchführung von numerischen Simulationen mit dem Prinzip der rekursiven Substrukturierung wurde von M. Schneider im Rahmen seiner

Promotion [63] das prototypische (Meta-)Programm ARESO [2] entwickelt.

ARESO unterstützt die Teamarbeit von Ingenieuren bei der Modellierung im Sinne der rekursiven Substrukturierung. Es ermöglicht eine parallel numerische Simulation eines technischen Systems mit verteilter Datenhaltung auf einem lose gekoppelten Rechnernetz. ARESO ist ein Meta-Programm, das die Schritte 1_{FE}, 3_{FE} und 5_{FE} einer FEM-Berechnung selbst durchführt und für die problemabhängige Berechnungsschritte 2_{FE} und 4_{FE} und die Fehlerschätzung externe Programme einsetzt (s. Kap. 2.1). Durch ein modulares Programmkonzept mit klar definierten Schnittstellen für diese problemabhängigen Berechnungsschritte ist ARESO für verschiedenste FE-Problemklassen geeignet und erlaubt den Einsatz von Standard FE-Programmen.

Eine numerische Simulation mit ARESO läuft wie folgt ab:

Zunächst wird die von Ingenieuren spezifizierte Beschreibung der rekursiven Unterteilung eines Objekts, der einzelnen Bauteile und des zu lösenden Randwertproblems eingelesen. Dabei erfolgt die Übergabe der Beschreibung in Form von Dateien.

Der Substrukturierungsbaum wird dynamisch aufgebaut. Dazu wird ein ARESO-Prozeß auf allen benötigten Rechnern des Rechnernetzes gestartet und die einzelnen Baumknoten auf den ihnen zugeordneten Rechnern lokal erzeugt.

Die ARESO-Prozesse führen die Berechnungsschritte auf den einzelnen Baumknoten lokal aus. Falls notwendig werden vorher Datenpakete von anderen Rechnern eingelesen und nach Beendigung des Berechnungsschritts an andere Rechner verschickt.

An den Blättern werden in den Schritten 2_{RS} und 4_{RS} problemspezifische Routinen zur Erzeugung der Elementgleichungssysteme und Elementinformationen aufgerufen. Für diese Routinen, auch Hilfsprozesse genannt, gibt es zwei Arten der Realisierung: *externe* und *interne Schnittstellen.* Im ersten Fall übernimmt ein Hilfsprozeß ein Auftragsdatenpaket von ARESO und übergibt ein Ergebnisdatenpaket. Der Hilfsprozeß ruft ein im Rechnernetz vorhandenes Standard FE-Programm auf. Bei internen Schnittstellen wird ein Hilfsdienstmodul zum ARESO-Programm hinzugebunden und direkt von ARESO aufgerufen. In diesem Fall müssen Teile vom Quelltext des Standard FE-Programmes zur Verfügung stehen.

Im Wurzelknoten erfolgt der Aufruf von Routinen zur Fehlerschätzung und Netzverfeinerung . Auch hier besteht die Alternative einer Realisierung über externe oder interne Schnittstellen.

Im Rahmen des adaptiven Berechnungszyklus (s. Abb. 16) wird der Substrukturierungsbaum mehrmals durchlaufen und bei Bedarf neue Baumknoten auf den verschiedenen Rechnern erzeugt.

[2] Der Name *ARESO* steht für "Arbeiten mit rekursiv substrukturierten Objekten".

Nach Beendigung der Berechnung erhält der Ingenieur die Ergebnisse der Simulation. Zur Unterstützung des Datentransfers zwischen dem Ingenieur und dem Programm ARESO bei Ein- und Ausgabe können Pre- und Postprozessoren zur Unterstützung eingesetzt werden, die im Rahmen von Diplomarbeiten entwickelt wurden ([43], [48], [49], [69]).

Mit ARESO sind parallele, adaptive, numerische Simulationen auf einem Netzwerk von Workstations praktisch durchführbar. Die Berechnungen erfolgen gemäß dem in Kapitel 3 beschriebenen Verfahren. ARESO diente als Grundlage dieser Arbeit, um eine Weiterentwicklung der Methode und eine Erweiterung des Anwendungsspektrum zu realisieren. Dabei handelt es sich um die Einführung von hierarchischen Basen und von biquadratischen Ansatzfunktionen sowie die Entwicklung eines iterativen Multilevelverfahren zur Lösung der Systemgleichungen, die in den folgenden Kapiteln beschrieben werden.

4 Ein iteratives Lösungsverfahren

Die in Kapitel 2 und 3 beschriebenen Berechnungsverfahren beruhen auf einer direkten Lösung der Systemgleichungen. Der große Vorteil eines iterativen Verfahrens liegt darin, daß der prinzipiell notwendige Speicherbedarf der Ordnung $O(n)$ bei einem FE-Problem mit n Freiheitsgraden erhalten werden kann.

4.1 Vergleich zwischen direkten und iterativen Verfahren

In der Finite-Element-Methode entstehen i.a. der größte Speicherbedarf und der meiste Rechenaufwand bei der Lösung der anfallenden linearen Gleichungssysteme. Die konkrete Wahl einer Methode zur Lösung eines sehr großen Gleichungssystem mit schwach besetzter Matrix beeinflußt daher entscheidend die Effizienz einer numerischen Simulation basierend auf der FEM. Dabei ist grundsätzlich zwischen *direkten* und *iterativen* Methoden zu unterscheiden.

4.1.1 Direkte Verfahren

Als *direkte Verfahren* bezeichnet man Algorithmen, die nach endlich vielen Rechenschritten die (bis auf Rundungsfehler) exakte Lösung des Gleichungssystems liefern [34]. Die bekanntesten direkten Methoden sind das Gaußsche Eliminationsverfahren und die symmetrische Zerlegung von Cholesky (s. z.B. [67]).

Da direkte Verfahren immer die exakte Lösung liefern, kann die Genauigkeit der Lösung nicht unter Berücksichtigung des Diskretisierungsfehlers gesteuert und der Lösungsvorgang vorzeitig abgebrochen werden.

Direkte Methoden formen die Matrix während des Lösungsprozesses um und bewirken dadurch ein Überschreiben von zuvor unbesetzten Einträgen der Matrix. Dieser Vorgang wird *Auffüllen* (engl. *fill-in*) genannt und führt dazu, daß direkte Methoden die schwach besetzte Matrixstruktur zum Teil zerstören.

Das Auffüllen der Matrix ist bei dem Gauß-Algorithmus oder der Methode von Cholesky auf die Hülle der Matrix beschränkt. Die Hülle einer Matrix umfaßt alle Elemente, die innerhalb der zeilenabhängigen Bandbreite liegen (s. [64]). Mit heuristischen Methoden kann die Hülle von FE-Systemmatrizen auch bei allgemeinen Berechnungsgebieten und ungleichmäßigen Gittern durch eine geeignete Numerierung der Unbekannten reduziert werden (s. z.B. [64]). Auf diesen Matrizen mit Band- bzw. Hüllenstruktur lassen sich hüllenorientierte Band- und Frontlösungsverfahren implementieren ([64]).

Das in den Kapiteln 3.2.2 und 3.2.3 beschriebene Verfahren der rekursiven Substrukturierung als direkte Methode zur Lösung der Systemgleichungen verwendet die gleiche Numerierungsstrategie wie die Methode der fortgesetzten Zerschneidung (*Nested Dissection*) von George ([24], [25]). Für diese Methode konnte nachgewiesen werden, daß sich der fill-in und damit der Rechenaufwand wesentlich verringert.

Im Gegensatz zur Nested Dissection und den hüllenorientierten Verfahren wird bei der rekursiven Substrukturierung kein globales Gleichungssystem für das Gesamtgebiet, sondern ein Baum von Gleichungssystemen aufgestellt. Diese Gleichungssysteme für die Teilgebiete sind nur über die Separatoren, entsprechend der rekursiven Zerlegung, miteinander gekoppelt.

4.1.2 Iterative Verfahren

Bei der iterativen Lösung eines Gleichungssystems

$$Ax = b \qquad\qquad (A \in \mathbf{K}^{I\times I}, b \in \mathbf{K}^{I}) \tag{29}$$

berechnet man ausgehend von einem beliebigen Startvektor $x^{(0)}$ eine Folge von *Iterierten* $x^{(m)}$ für $m = 1, 2, \ldots$:

$$x^{(0)} \mapsto x^{(1)} \mapsto x^{(2)} \mapsto \cdots \mapsto x^{(m)} \mapsto x^{(m+1)} \mapsto \cdots$$

bis eine ausreichend gute Näherungslösung erhalten wird. Dabei bezeichnet $\mathbf{K}^{I}$ den Raum der Vektoren und $\mathbf{K}^{I\times I}$ den Raum der Matrizen zur Indexmenge I [34]. Etwas formaler definiert man:

Definition 4.1 *Ein* **Iterationsverfahren** *ist eine (lineare oder nichtlineare) Abbildung* $\Phi : \mathbf{K}^{I} \times \mathbf{K}^{I} \to \mathbf{K}^{I}$. *Die Folgeglieder der Iteration werden durch die Vorschrift*

$$x^{(m+1)} := \Phi(x^{(m)}, b) \tag{30}$$

aus einem Startwert $x^0 \in \mathbf{K}^{I}$ *erzeugt.*

Definition 4.2 $x^* = x^*(b)$ *heißt* **Fixpunkt** *des Iterationsverfahrens* Φ *zu* $b \in \mathbf{K}^{I}$, *falls*

$$x^* = \Phi(x^*, b). \tag{31}$$

Definition 4.3 *Ein Iterationsverfahren heißt* **konvergent**, *wenn für alle* $b \in \mathbf{K}^{I}$ *ein vom Startwert* $x^{(0)} \in \mathbf{K}^{I}$ *unabhängiger Grenzwert* $x^*(b)$ *der Iterierten (30) existiert.*

Ein iteratives Verfahren muß entscheiden, wann die Iteration abgebrochen wird. Solche Abbruchkriterien sollten feststellen, ob der Fehler

$$e^{(m)} = x^{(m)} - x^* \tag{32}$$

klein genug ist oder die Berechnung eine maximal zulässige Anzahl von Iterationen überschritten hat [7]. Da $e^{(m)}$ im allgemeinen schwer direkt abzuschätzen ist, wird stattdessen das *Residuum* (auch Defekt bezeichnet)

$$r^{(m)} = Ax^{(m)} - b \tag{33}$$

zur Beurteilung der Genauigkeit der aktuellen Iterierten verwendet. Dies ist aber nur eine geeignete Fehlerabschätzung, wenn die Kondition der Matrix klein ist. Es gibt verschiedene Abbruchkriterien, die die Norm des Residuenvektors gewichtet mit normierten Systemgrößen (z.B. Normen von der Systemmatrix, der rechten Seite oder der aktuellen Iterierten) auswerten (s. z.B. [7]). Diese Abbruchkriterien können einzeln oder kombiniert eingesetzt werden.

Als Beispiel für ein Iterationsverfahren betrachten wir das *Gauß-Seidel-Verfahren* (auch *Einzelschrittverfahren* genannt) (s. z.B. [34]). Dieses Verfahren zeigt zwar bei diskretisierten Variationsaufgaben ein schlechtes Konvergenzverhalten, wird aber wegen seiner Glättungseigenschaft oft innerhalb von Multilevel-Verfahren [55] verwendet. Auch das in der vorliegenden Arbeit vorgestellte iterative Verfahren basiert auf einer Gauß-Seidel-Glättung innerhalb eines hierarchisierten Funktionenraums.

Für das Gleichungssystem $Ax = b$ und die eindeutige Zerlegung

$$A = D - E - F, \tag{34}$$

wobei D der Diagonalanteil ($D := \mathrm{diag}(A)$), E eine strikte untere Dreiecksmatrix und F eine strikte obere Dreiecksmatrix ist, lautet das Gauß-Seidel-Verfahren, dargestellt in der *ersten Normalform* [34]:

$$x^{(m+1)} = M^{GS} x^{(m)} + N^{GS} b \tag{35}$$

Dabei sind die Iterationsmatrizen

$$M^{GS} = (D - E)^{-1} F \tag{36}$$

$$N^{GS} = (D - E)^{-1} \tag{37}$$

Die *zweite Normalform* des Gauß-Seidel-Verfahren lautet:

$$x^{(m+1)} = x^{(m)} - (D - E)^{-1}(Ax^m - b) \tag{38}$$

Hier wird deutlich, daß $x^{(m+1)}$ aus $x^{(m)}$ durch eine Korrektur hervorgeht, die sich aus der Multiplikation des Defektes $Ax^m - b$ mit N^{GS} ergibt.

Man kann eine Gauß-Seidel-Iteration auch in komponentenweiser Darstellung beschreiben (s. Abb. 17). Der Aufwand einer Iteration ist vergleichbar mit der Multiplikation einer Matrix mit einem Vektor [68].

Das Gauß-Seidel-Verfahren läuft sequentiell ab und benötigt keine Zwischenspeicherung der neu berechnetet Komponenten der Näherungslösung. Die Iteration hängt von der Anordnung der Indizes ab. Für symmetrisch-definite Gleichungssysteme kann gezeigt werden, daß die Vektorfolge $x^{(m)}$ des Einzelschrittverfahrens

```
proc Gauß-Seidel(A,b,x)
    for  i := 1  to  n  do
```
$$x_i^{(m+1)} := (b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(m+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(m)})/a_{ii}$$
```
endproc
```

Abbildung 17: **Gauß-Seidel-Iteration**

stets gegen die Lösung des Gleichungssystems konvergiert [64].

Viele Iterative Methoden zur Gleichungslösung lassen die Koeffizientenmatrix unverändert und können dadurch die schwache Besetzung der Matrizen voll ausnutzen. Da nur die von Null verschiedenen Matrixelemente gespeichert werden müssen, benötigen sie weniger Speicherbedarf als direkte Methoden. Zudem spielt die Besetzungsstruktur der Matrix im allgemeinen keine zentrale Rolle. Ein weiterer Vorteil ist die Möglichkeit, die Genauigkeit der Lösung zu steuern.

Trotz der Vorteile der iterativen Methoden kommen in den meisten in der Industrie eingesetzten Programmpaketen zur numerischen Simulation basierend auf der Finite-Element-Methode direkte Verfahren zur Lösung der entstehenden Systemgleichungen zum Einsatz [23]. Die Hauptgründe liegen darin, daß direkte Lösungsverfahren

- robust und zuverlässig
- vielseitig einsetzbar
- in der CPU-Zeit vorhersehbar und
- in vielen praktischen Anwendungen schneller als iterative Löser

sind. Ihr Hauptnachteil ist, daß ihre Berechnungskosten bei großen Problemen wesentlich stärker anwachsen als bei effizienten iterativen Verfahren.

Um diesen Nachteil zu beheben, wird in dieser Arbeit der direkte Lösungsalgorithmus der rekursiven Substrukturierung umgewandelt in ein iteratives Lösungsverfahren, wobei alle Vorteile bezüglich Modellierung, Parallelisierbarkeit und Ausrichtung auf verschiedenste FE-Problemstellungen beibehalten werden. Vor der Umwandlung der rekursiven Substrukturierung in ein iteratives Lösungsverfahren ist es zunächst sinnvoll, sich die Besetzungsstruktur der Matrizen etwas genauer anzuschauen.

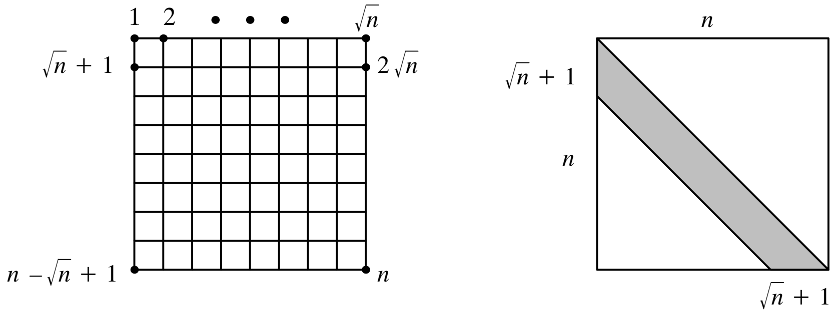


Abbildung 18: (a) uniformes Gitter mit zeilenweiser Numerierung der Knotenvariablen, (b) Bandmatrix (nur unteres Dreieck wegen Symmetrie)

4.2 Die Besetzungsstruktur der Matrizen

In der Methode der Finiten Elemente wird für die gesuchte Lösungsfunktion ein problemgerechter Ansatz gewählt. Bei diesen *Ansatzfunktionen*, auch *Form-* oder *Basisfunktionen* genannt, kommen bei eindimensionalen Elementen vor allem Polynome ersten, zweiten und gelegentlich sogar höheren Grades in Frage. Für zweidimensionale Elemente finden unter anderem lineare, quadratische, bilineare oder biquadratische Ansätze Verwendung [64].

Bei der Wahl der Basisfunktionen spielen das zu behandelnde Problem, die Form der Elemente und bestimmte problemabhängige Stetigkeitsbedingungen eine wesentliche Rolle. Unabhängig vom gewählten Ansatz haben die Basisfunktionen in der Standard-FEM die Eigenschaft gemeinsam, daß sie lokal in jedem Element definiert werden. Dadurch beschränkt sich die Trägermenge einer Ansatzfunktion auf die Fläche eines Finiten Elements. Die Lösungsfunktion für das Gesamtgebiet läßt sich durch eine gewichtete Summe dieser lokalen Ansatzfunktionen darstellen.

Die Beschränkung auf elementlokale Formfunktionen hat einen entscheidenden Einfluß auf die Besetzungsstruktur der Gesamtsteifigkeitsmatrix, die aus den Elementsteifigkeitsmatrizen zusammengesetzt wird. Eine Matrixzeile eines Freiheitsgrades an einem Knoten im FE-Netz hat höchstens Einträge in den Matrixzeilen derjenigen Freiheitsgrade, die zu Knoten in angrenzenden Finiten Elementen gehören. Deshalb sind die entstehenden Systemmatrizen bei der FEM dünn besiedelt und können mit speziellen Lösungsverfahren bearbeitet werden.

Durch die lokale Begrenztheit der Ansatzfunktionen besteht höchstens dann eine direkte Kopplung zwischen zwei Gitterpunkten (d.h. ein Eintrag in der Steifigkeitsmatrix für ihre zugehörigen Freiheitsgrade), wenn beide in demselben Element liegen. Dies hat in der Standard-FEM bei der Verwendung von regelmäßigen Netzen (s. Abb. 18(a)) zur Folge, daß jeder Gitterpunkt im Gebietsinneren zu vier Elementen gehört und acht benachbarte Punkte besitzt. Eine zeilenweise Numerierung der n Knotenvariablen, wie in der Abbildung, führt damit zu einer Matrix mit Bandstruktur und der Bandbreite $\sqrt{n}+1$ (s. Abb. 18(b)). Bei Verwendung ei-

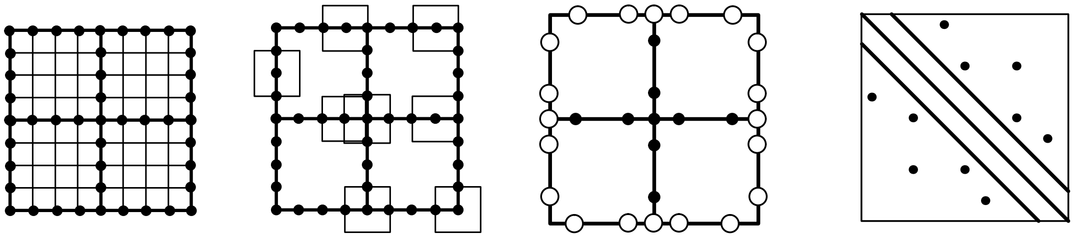


Abbildung 19: Verwendung von Knotenbasen: (a) Verfeinerung eines Teilgebiets, (b) Knotenbasen auf dem Separator und dem Teilgebietsrand, (c) Punkte mit stärkerer Kopplung, (d) Matrixstruktur (tridiagonal)

nes direkten Verfahren zur Gleichungslösung ist zumindest die Hülle der Matrix zu speichern (s. Kap. 4.1.1). Insgesamt entsteht dadurch für die symmetrische Matrix ein Speicherbedarf der Ordnung $O(n^{1.5})$.

Zur Analyse der Besetzungsstruktur der Matrizen bei der rekursiven Substrukturierung betrachten wir die uniforme Vernetzung eines Quadratgebietes in Abb. 19(a). Wir konzentrieren uns speziell auf den obersten Knoten des Substrukturierungsbaumes dieses Teilgebiets (Abb. 19(b)). Aufgrund der Lage der Stützpunkte auf dem Separator und dem Teilgebietsrand grenzen die meisten Gitterpunkte nur an zwei Nachbarpunkte (insbesondere bei stärkerer Verfeinerung). In der Abbildung sind die Grundgebiete, in denen die Basisfunktion eines Gitterpunktes von 0 verschieden ist, für diverse Gitterpunkte eingezeichnet. Wie in Abb. 19(c) dargestellt, existiert unabhängig von der Maschenweite eine konstante Anzahl von Punkten an den Kreuzungspunkten der Separatoren und des Teilgebietsrandes mit drei (weiße Punkte) und vier Nachbarpunkten (schwarze Punkte). Als Konsequenz ergibt sich bei geeigneter Numerierung eine nahezu tridiagonale Steifigkeitsmatrix, die eine konstante Anzahl von Zeilen mit mehr als drei Einträgen (maximal fünf) besitzt.

Während sich bei der Standard-FEM eine Matrix für das Gesamtsystem ergibt, findet man hier einen Baum von Matrizen vor. Da sich die Länge der Separatoren und der Teilgebietsränder von Ebene zu Ebene halbiert, nimmt die Größe der Matrizen im Substrukturierungsbaum von der Wurzel zu den Blättern stark ab.

In Abb. 20 wird verdeutlicht, wie sich die Größe der Matrizen von Ebene zu Ebene verringert. Dabei ist noch nicht berücksichtigt, daß die Randbedingungen dezentral verarbeitet werden, was insbesondere in den obersten Baumebenen eine weitere Reduktion der Matrizengröße bewirkt (s. Kap. 7).

Wenn man den Speicherbedarf dieser dünnbesiedelten Matrizen über den gesamten Baum aufsummiert, erhält man einen Gesamtspeicherbedarf der Ordnung $O(n)$ (s. Kap. 7).

Diese günstige Besetzungsstruktur der Matrizen und der geringe Speicherbedarf können allerdings mit dem direkten Lösungsverfahren der rekursiven Substrukturierung nicht genutzt werden. Bei der Lösung der Gleichungssysteme wer-

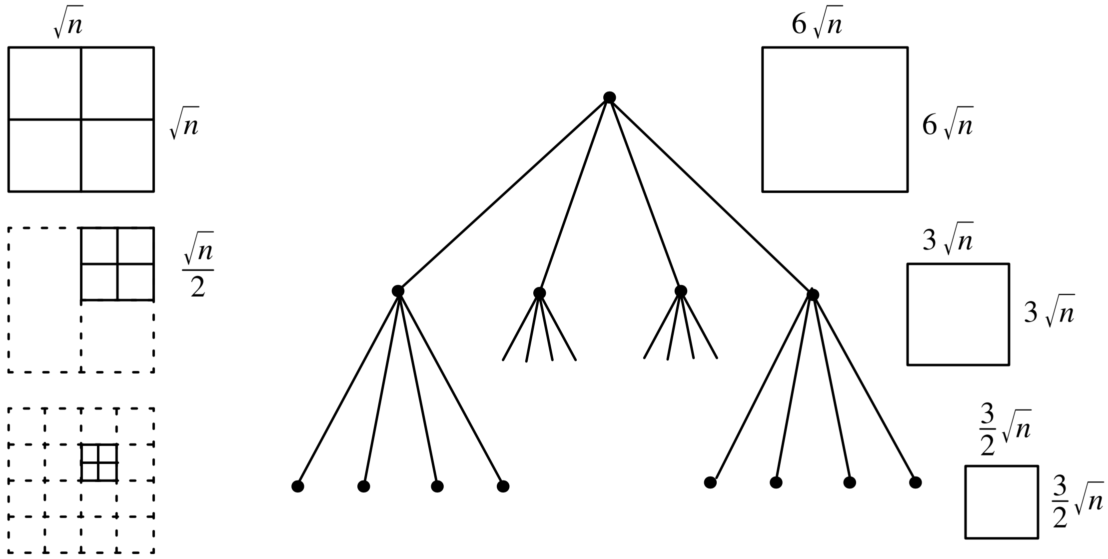


Abbildung 20: Matrixgröße im Substrukturierungsbaum

den in Schritt 2_{RS} die ursprünglich dünnbesiedelten Matrizen durch die statische Kondensation umgeformt und sind danach nahezu vollbesetzt. Man muß also einen Baum betrachten, in dem volle Matrizen gespeichert werden, deren Größe sich von Ebene zu Ebene verringert. Durch die wachsende Anzahl von Baumknoten pro Ebene bleibt der Speicherbedarf pro Baumebene konstant. Der gesamte Speicherbedarf ist abhängig von der Baumtiefe und damit von der Ordnung $O(nlog(n))$.

Um den Speicherbedarf auf die Ordnung $O(n)$ zu reduzieren, muß das direkte Lösungsverfahren durch ein iteratives Verfahren ersetzt werden, das die dünne Besetzungsstruktur der Matrizen gar nicht oder nur unwesentlich stört.

4.3 Der iterative Algorithmus

In diesem Abschnitt wird beschrieben wie in das Berechnungsverfahren der rekursiven Substrukturierung ein iteratives Lösungsverfahren integriert werden kann. Dazu sind die Teilschritte 2_{RS} und 3_{RS} zur Erstellung und Lösung der Knotengleichungssysteme zu modifizieren. Die restlichen Teilschritte 1_{RS}, 4_{RS} sowie der adaptive Berechnungszyklus aus Abb. 16 bleiben unverändert.

Der **Schritt** 2_{RSI} für den iterativen Algorithmus besteht nur noch aus drei Teilschritten:

(1) Aufstellung von Matrix und Lastvektor.

(2) Trennung von inneren und äußeren Freiheitsgraden.

(3) Behandlung der Randbedingungen und der inkompatiblen Punkte.

Im Vergleich zur direkten Lösungsmethode (s. Kap. 3.2.2) wird hier auf den teuersten Schritt bei der Matrixtransformation, die statischen Kondensation, verzichtet. Dadurch wird die strukturelle Information vom Teilgebietsinneren nicht auf den Teilgebietsrand übertragen.

Um trotzdem eine korrekte Lösung des gestellten Problems zu erhalten, muß man die statische Kondensation und den Schritt 3_{RS} zur Lösung der Gleichungssysteme durch ein Iterationsverfahren ersetzen. Es ist aber zu beachten, daß hier ein Baum von Gleichungssystemen zu lösen ist. Man unterscheidet deshalb zwischen zwei Arten von Iteration:

- Iteration auf den Baumknoten
- Iteration zwischen den Baumknoten

Die Iteration auf einem Baumknoten berechnet eine verbesserte Iterierte auf dem Knotengleichungssystem mit einem Glättungsschritt. In unserem Fall wird ein Gauß-Seidel Iterationsschritt (s. Abb. 17) durchgeführt. Die Iteration zwischen den Baumknoten erfolgt durch abwechselnde top-down und bottom-up Baumdurchläufe. Dabei werden die aktuellen Iterierten im Baum nach unten gereicht und anschließend die rechten Seiten, bzw. die Kraftvektoren, von den Blättern bis zur Wurzel korrigiert.

Die Matrizen bleiben während der Iteration unverändert. Dadurch wird die schwache Besetzung der Matrizen, wie in Kapitel 4.2 beschrieben, erhalten. Der Speicherbedarf reduziert sich im Vergleich zum direkten Lösungsalgorithmus von $O(n \cdot log(n))$ auf $O(n)$ (vgl. Kap. 7).

Der **Schritt** 3_{RSI} beim iterativen Lösungsalgorithmus zur Lösung der Gleichungssysteme kombiniert die beiden Arten von Iteration. Er läßt sich in zwei Phasen einteilen:

Phase 1: In einem top-down-Baumdurchlauf wird die aktuelle Lösung vom Vater an die Söhne weitergegeben. In den Söhnen wird mit der Lösung auf dem Gebietsrand eine verbesserte Lösung im Gebietsinneren berechnet.

Phase 2: In einem bottom-up-Baumdurchlauf werden mit der Verbesserung der Lösung im Gebietsinneren die Korrekturen der rechten Seite auf dem Gebietsrand berechnet und an den Vaterknoten übergeben.

Diese beiden Phasen bewirken, daß der W-Zyklus aus Abb. 15 um einen Zyklus von mehreren aufeinanderfolgenden top-down- und bottom-up-Baumdurchläufe ergänzt wird (s. Abb. 21). Dabei werden die beiden Phasen solange wiederholt, bis der Fehler der Iterierten (32) klein genug und damit das Abbruchkriterium für die Iteration erfüllt ist. In den folgenden beiden Kapitel werden die beiden Phasen im Detail beschrieben.

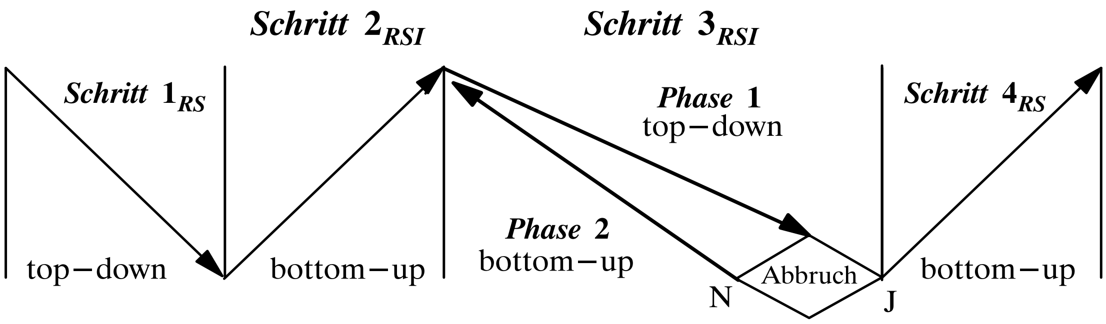


Abbildung 21: Darstellung des iterativen Lösungsverfahrens

4.3.1 Phase 1 (top-down)

Zunächst betrachten wir das Gleichungssystem eines Knotens des Substrukturierungsbaumes. Im Vergleich zum direkten Lösungsalgorithmus handelt es sich hier bei dem Lösungsvektor u und der rechten Seite f um Vektoren, die in jedem Iterationsschritt verändert werden. Außerdem ist in jedem Baumknoten ein zusätzlicher Vektor r notwendig, der die gleichen Dimension wie der Lösungsvektor u besitzt und zur Abspeicherung von Zwischenergebnissen eines Iterationsschrittes verwendet wird. Insgesamt müssen an einem Knoten des Substrukturierungsbaumes eine dünnbesiedelte Matrix sowie drei Vektoren, getrennt nach inneren und äußeren Freiheitsgraden, abgespeichert werden.

$$\begin{pmatrix} K_{aa} & K_{ai} \\ K_{ia} & K_{ii} \end{pmatrix} \begin{pmatrix} u_a^{(m)} \\ u_i^{(m)} \end{pmatrix} = \begin{pmatrix} f_a^{(m)} \\ f_i^{(m)} \end{pmatrix} \qquad \begin{pmatrix} r_a^{(m)} \\ r_i^{(m)} \end{pmatrix} \tag{39}$$

In dem Vektor r stehen Ergebnisse, die an den Vaterknoten weitergereicht werden ($r_a^{(m)}$) und Ergebnisse, die nach einem top-down-Schritt im Baumknoten für den nächsten bottom-up-Schritt des Knotens gespeichert werden ($r_i^{(m)}$).

Am Beginn von Phase 1 im $(m+1)$-ten Iterationsschritt sind die Werte der äußeren Freiheitsgrade $u_a^{(m+1)}$ gegeben. Mit diesen Werten der Unbekannten des Teilgebietsrandes kann die Iteration auf einem Baumknoten durchgeführt werden. Im ersten Iterationsschritt beginnt man dabei mit der Startlösung $u_a^{(0)}$, die aus einem gröberen Netz eines adaptiven Berechnungszyklus interpoliert wird oder eine triviale Lösung ist.

Für die Iteration auf dem Baumknoten werden ein oder mehrere Glättungs- oder Relaxationsschritte in Form von Gauß-Seidel-Iterationen auf dem Knotengleichungssystem durchgeführt. Dabei dürfen nur die inneren Unbekannten des Gleichungssystems $u_i^{(m)}$ verändert werden. Eine Änderung der Unbekannten auf dem Teilgebietsrand $u_a^{(m+1)}$ würde die Stetigkeitsbedingung zu angrenzenden Teilgebieten verletzen.

Bei genauerer Betrachtung der Gauß-Seidel-Iteration (s. Abb. 17) erkennt man, daß zur Relaxation der i-ten Unbekannten die i-te Gleichung nach der i-ten Unbekannten aufgelöst wird. Da sich die Werte von $u_a^{(m+1)}$ während der Iteration auf einem Knoten nicht mehr ändern dürfen, kann ihr Einfluß in einem Schritt auf die rechte Seite übertragen werden. Mit der Kopplungsmatrix K_{ia} zwischen den inneren und äußeren Freiheitsgraden kann die rechte Seite für das Knotengleichungssystem entsprechend modifiziert werden.

$$f_i' = f_i^{(m)} - K_{ia} \cdot u_a^{(m+1)} \tag{40}$$

Dabei handelt es sich aber nur um eine temporäre Änderung von f_i. Mit dieser modifizierten rechten Seite kann man die Gauß-Seidel-Iteration auf die untere Hälfte des Systems von (39) beschränken. Es werden nun eine oder mehrere Gauß-Seidel-Iterationen auf dem Teilgebietsinneren durchgeführt:

$$\text{Gauß-Seidel}(K_{ii}, f_i', u_i^{(m)}) \tag{41}$$

Es könnte auch ein anderes Glättungsverfahren auf dem Gleichungssystem

$$K_{ii} u_i^{(m)} = f_i' \tag{42}$$

zur Bestimmung einer Verbesserung der Lösung im Gebietsinneren verwendet werden. Die Wahl für das Gauß-Seidel-Verfahren liegt in der einfachen Implementierbarkeit in unserem rekursiven Substrukturierungskonzept und in der Möglichkeit einer adaptiven Steuerung der Iteration auf Gitterpunkten, die einen gewissen Mindestbetrag zur Approximation beitragen (s. Kap. 6).

Das Ergebnis $u_i^{(m+1)}$, die $(m+1)$-te Iterierte für die Lösung im Gebietsinneren, überschreibt den Vektor $u_i^{(m)}$. Die Korrektur der Lösung Δu_i wird in dem Vektor r_i gespeichert, da sie in Phase 2 zur Korrektur der rechten Seite benötigt wird.

$$r_i^{(m+1)} = \Delta u_i^{(m+1)} = u_i^{(m+1)} - u_i^{(m)} \tag{43}$$

Nach diesem Schritt ist die Iteration auf einem Baumknoten in Phase 1 abgeschlossen. Als Ergebnis liegen die aktuellen Werte von $u_a^{(m+1)}$, $u_i^{(m+1)}$ und $r_i^{(m+1)}$ vor.

Wie bereits erwähnt, besteht ein Iterationsschritt auch aus einer Iteration zwischen den Baumknoten. In Phase 1 wird dabei der Baum top-down durchlaufen und jeder Knoten übergibt die aktuelle Iterierte $u^{(m+1)}$ an seine Sohnknoten.

Gestartet wird Phase 1 im Wurzelknoten, der keine äußeren Freiheitsgrade besitzt. Dadurch kann auf den Schritt (40) verzichtet und sofort eine Gauß-Seidel-Iteration durchgeführt werden. Ein allgemeiner Knoten im Substrukturierungsbaum bekommt zuerst die $(m+1)$-te Iterierte seines Unbekannten-Vektors auf dem Teilgebietsrand $u_a^{(m+1)}$ von seinem Vaterknoten übergeben. Dann führt er die Schritte (40), (41) und (43) aus. Schließlich übergibt der Knoten seinen Lösungsvektor an die nächsttiefere Baumebene. Phase 1 endet in den Blättern des Substrukturierungsbaumes mit der neuen Lösung $u^{(m+1)}$ und der Korrektur der Lösung $r_i^{(m+1)}$ auf allen Knotengleichungssystemen. In Abb. (22) ist Phase 1 des iterativen Lösungsalgorithmus nochmals komprimiert dargestellt.

(1) falls Knoten äußere FG besitzt

empfange $u_a^{(m+1)}$ vom Vaterknoten

berechne $\quad f_i' = f_i^{(m)} - K_{ia} \cdot u_a^{(m+1)}$

(2) führe ν Glättungsschritte Gauß-Seidel(K_{ii},f_i',$u_i^{(m)}$) aus, um $u_i^{(m+1)}$ zu erhalten

(3) Speichere Änderung der Lösung $\quad r_i^{(m+1)} = u_i^{(m+1)} - u_i^{(m)}$

(4) Schicke $u^{(m+1)}$ an die Söhne

(5) Wende Phase 1 rekursiv auf die Söhne an.

Abbildung 22: **PHASE 1 des iterativen Algorithmus (top down)**

Im Prinzip handelt es sich bei Phase 1 um einen Gauß-Seidel-Iterationsschritt auf den Freiheitsgraden des zugrunde liegenden FE-Netzes. Durch die Kombination mit der rekursiven Substrukturierung ist die Reihenfolge der Bearbeitung der Gitterpunkte anhand der rekursiven Zerlegung des Berechnungsgebiets vorgegeben.

Wir betrachten wieder ein uniform verfeinertes Netz auf einem Quadratgebiet. Das Gauß-Seidel-Verfahren ist abhängig von der Anordnung der Unbekannten. Für ein uniformes Gitter erhält man bei einer zeilenweisen Anordnung das lexikographische Gauß-Seidel-Verfahren [34]. Es sind aber auch komplexere Anordnungsstrategien möglich (z.B. Schachbrett Gauß–Seidel [34]). Die Anordnung der Unbekannten hat Auswirkungen auf das Konvergenzverhalten und auf die praktische Implementierung des Verfahrens.

In unserem Algorithmus werden zuerst die Unbekannten auf dem Separator des Wurzelknotens geglättet. Anschließend werden die Separatoren der nächsten Baumebene behandelt (s. Abb. 23(b)). Durch die rekursive Zerlegung des Berechnungsgebiets erhält man die gleiche Anordnung der Unbekannten wie bei der Nested Dissection [24]. In Abb. 23 sind zwei verschiedene Anordnungsstrategien für ein uniformes Gitter zu sehen. Bei der rekursiven Substrukturierung sind die Unbekannten in drei verschiedene Ebenen eingeteilt. Bei Variablen aus der gleichen Ebene, die in verschiedenen Teilgebieten liegen, ist die Reihenfolge der Glättung untereinander beliebig, da sie nicht direkt voneinander abhängen. Deshalb können alle Teilgebiete einer Ebene parallel geglättet werden. Die Variablen in den Teilgebieten sind dagegen unmittelbar voneinander abhängig. Hier kann die Reihenfolge der Bearbeitung (z.B. in aufsteigender oder absteigender Ordnung) einen Einfluß auf das Ergebnis des Glättungsschrittes haben.

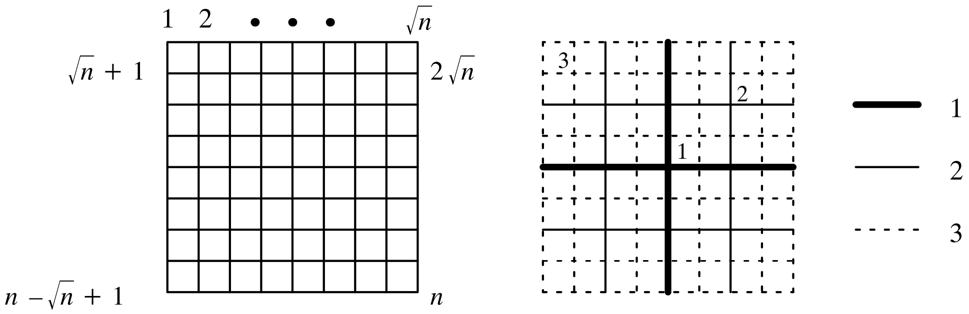


Abbildung 23: Reihenfolge der Glättung: (a) lexikographischer Gauß-Seidel, (b) rekursive Substrukturierung, Nested Dissection

4.3.2 Phase 2 (bottom-up)

Am Ende von Phase 1 hat jeder Knoten die Werte seiner inneren Variablen geändert. Diese Änderungen im Teilgebiet haben aber Auswirkungen auf die Lösung auf dem Teilgebietsrand (d.h. auf die obere Hälfte von (39)). Physikalisch bewirken Änderungen der Lösung im Gebietsinneren eine Kraft auf dem Gebietsrand.

In Phase 2 werden die Änderungen von Phase 1 auf den Teilgebietsrand übertragen. Dabei wird aber nicht die Lösung auf dem Rand geändert, sondern der Kraftvektor f_a des Teilgebietsrandes korrigiert. Änderungen der Lösung würden zu Unstetigkeiten zwischen den Teilgebieten führen, Änderungen des Kraftvektors können dagegen in der nächsthöheren Ebene aufsummiert werden. Es gibt zwei Quellen, die den Vektor f_a eines Baumknotens korrigieren:

- Änderungen der Lösung u_i im Gebietsinneren des Baumknotens.
- heraufgereichte Änderungen der f_a der Söhne, die sowohl f_i als auch f_a des Baumknotens ändern

Betrachten wir zuerst die erste Quelle der Änderung. In einem Baumknoten ist am Beginn der Phase 2 der aktuelle Lösungsvektor $u^{(m+1)}$ und der bereits korrigierte Kraftvektor für das Teilgebietsinnere $f_i^{(m+1)}$ gegeben. Mit der aktuellen Iterierten $u_i^{(m+1)}$ kann das Teilgleichungssystem für den Gebietsrand modifiziert werden:

$$K_{aa} \cdot u_a^{(m+1)} = f_a^{(m)} - K_{ai} \cdot u_i^{(m+1)} \tag{44}$$

Da die Änderungen, verursacht durch die Iterierte $u_i^{(m)}$, bereits im letzten Iterationsschritt abgezogen wurde, ist nur noch die Änderung der Lösung $\Delta u_i^{(m+1)}$, die in $r_i^{(m+1)}$ zwischengespeichert ist, zu berücksichtigen. Damit ergibt sich die Korrektur der rechten Seite des Teilgebietsrandes zu

$$r_a^{(m+1)} = \Delta f_a^{(m+1)} = -K_{ai} \cdot \Delta u_i^{(m+1)} = -K_{ai} \cdot r_i^{(m+1)} \tag{45}$$

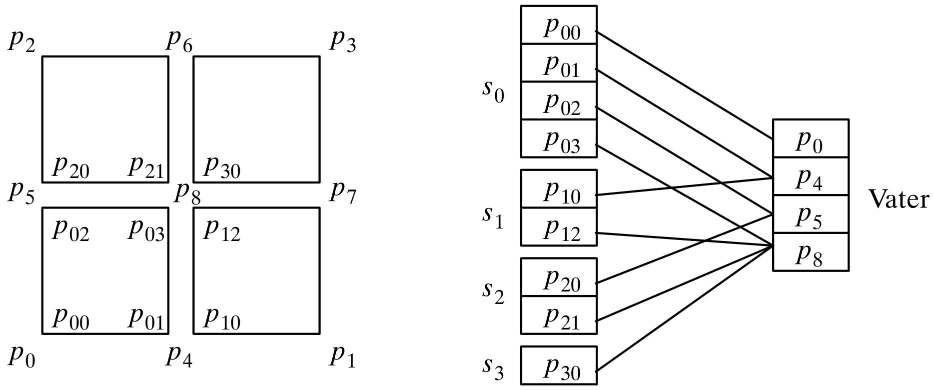


Abbildung 24: Zuordnung der Punkte von Söhnen zu Vater bei Assemblierung

und wird in $r_a^{(m+1)}$ zwischengespeichert. Die Korrektur der rechten Seite (45) aller Baumknoten erfolgt in einem bottom-up-Baumdurchlauf und startet deshalb in den Blattknoten.

Die rechte Seite f_a eines Baumknotens wird auch in anderen Teilgebieten korrigiert. Da frühestens im Vaterknoten alle Korrekturen von f_a bekannt sind, darf die Korrektur der rechten Seite f_a nicht im Knoten selbst, sondern erst in der nächsthöheren Baumebene durchgeführt werden.

In der Ebene des Vaterknotens werden die Korrekturen auf den Rändern der Sohnknoten $r_a^{(m+1)}$ in einem Assemblierungsschritt aufgesammelt und der Kraftvektor tatsächlich modifiziert. Die neue rechte Seite eines Baumknotens ergibt sich aus der alten rechten Seite und einer Summe aus den Korrekturen der K Söhne:

$$f^{(m+1)} = f^{(m)} + \sum_{s=1}^{K} r_a^{(m+1)}(s) \tag{46}$$

Dies ist die zweite Quelle der Korrektur der rechten Seite und korrigiert den gesamten Kraftvektor, d.h. f_i und f_a. Der Korrekturvorgang in (46) ist identisch mit der Assemblierung der rechten Seite in Schritt 2_{RSI}, mit dem Unterschied, daß hier nur die Korrekturen anstelle der gesamten rechten Seiten überlagert werden. Hierzu ist anzumerken, daß es sich in (46) um eine spezielle Summe von Vektoren, die unterschiedliche Längen haben können, handelt. Der Summe liegt eine Abbildungsvorschrift zugrunde, die jedem Freiheitsgrad der Söhne einen Freiheitsgrad im Vater zuordnet. Die Zuordnung der Freiheitsgrade von Sohn zu Vater erfolgt wieder mit Hilfe der Zerlegungsinformation aus Schritt 1_{RS} und den Schnittstellenbäumen aus Schritt 2_{RSI} und ist in Abb. 24 an einem Beispiel verdeutlicht. Dabei sind für vier ausgewählte Stützpunkte des Vaterknotens alle zugehörigen Stützpunkte in den Söhnen angegeben. Für Punkt p_4 sind z.B. die Korrekturen der rechten Seiten von Punkt p_{01} aus Sohn 0 und Punkt p_{10} aus Sohn 1 auf die rechte Seite zu addieren.

(1) falls Knoten Söhne hat

Assembliere neue rechte Seite

$f^{(m+1)} = f^{(m)} + \sum_{s=1}^{K} r_a^{(m+1)}(s)$

(2) Speichere Änderungen von f_a, die von Söhnen heraufgereicht wurde
$r_a^{(m+1)} = f_a^{(m+1)} - f_a^{(m)}$

(3) berechne Korrektur der rechten Seite
$r_a^{(m+1)} = r_a^{(m+1)} - K_{ai} \cdot r_i^{(m+1)}$

(4) Schicke $r_a^{(m+1)}$ an Vaterknoten

(5) Wende Phase 2 auf den Vaterknoten an

Abbildung 25: **PHASE 2 des iterativen Algorithmus (bottom up)**

Durch (46) wird die rechte Seite sowohl im Gebietsinneren wie auch am Gebietsrand modifiziert. Modifikationen am Gebietsrand sind aber erst auf der nächsthöheren Baumebene relevant und werden im Vektor $r_a^{(m+1)}$ gespeichert. Zu diesen Korrekturen von f_a werden die Änderungen der rechten Seite aus (45) addiert. Am Ende von Phase 2 werden die Korrekturwerte $r_a^{(m+1)}$ an den Vaterknoten geschickt.

Die einzelnen Teilschritte von Phase 2 des iterativen Lösungsalgorithmus sind in Abb. (25) nochmals komprimiert dargestellt.

Betrachtet man die Gauß-Seidel-Iteration in Abb. 17, so werden zur Berechnung von $x_i^{(m+1)}$ sowohl Werte aus der neuen Iterierten $x^{(m+1)}$ als auch Werte der letzten Iterierten $x^{(m)}$ berücksichtigt. In einem Baumknoten sind aber die Werte dieser letzten Iterierten von den Sohnknoten nicht sichtbar. Deshalb wird in den Söhnen in Phase 2 durch (45) diese zweite Summe gebildet und durch den Assemblierungsschritt (46) von der rechten Seite des Vaterknotens abgezogen. Der Vater braucht dann nur noch die Teilsumme, die seine eigenen Knotenvariablen betreffen, berechnen.

Zum Abschluß des Kapitels werden beide Phasen, insbesondere der Übergang von einem Vater zu einem Sohn und umgekehrt, in Abb. 26 schematisch dargestellt. In Phase 1 übergibt der Vaterknoten an seinen Sohn die aktuelle Iterierte $u_a^{(m+1)}$ auf dessen Gebietsrand. Damit wird die aktuelle Iterierte im Gebietsinneren des Sohnes berechnet. In Phase 2 erhält der Sohnknoten aus seinen Söhnen die Korrektur seiner rechten Seite Δf_i sowie erste Anteile für die Korrektur von f_a. Als nächstes wird die endgültige Korrektur Δf_a für den Gebietsrand berechnet und an den Vater gesendet. Bei ihm zerfällt $\Delta f_a^{(m+1)}$ wieder in Korrekturen

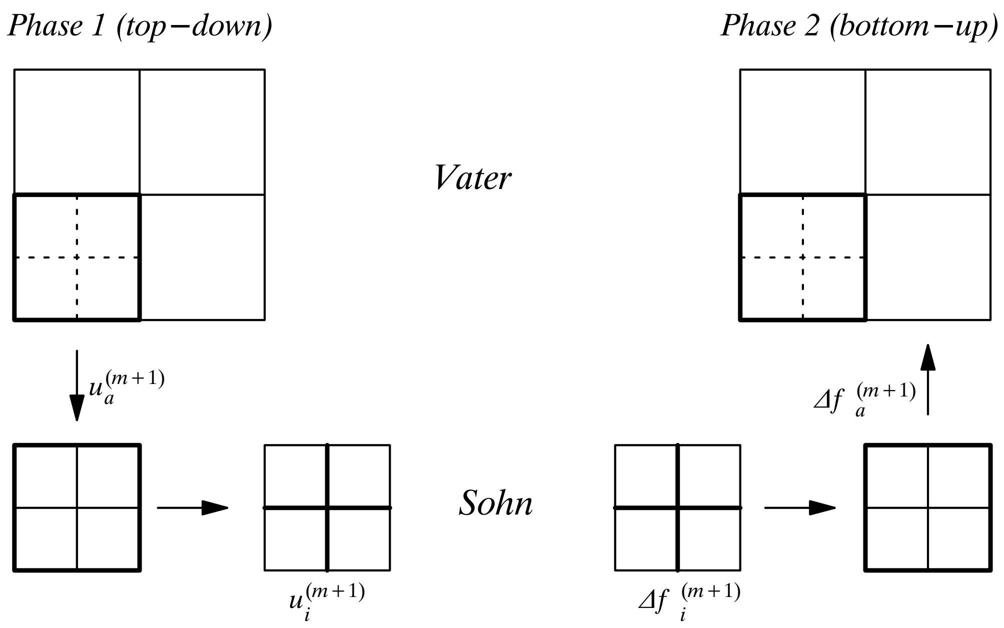


Abbildung 26: Datentransfer beim iterativen Algorithmus

auf Punkte nach außen und Punkte nach innen.

4.4 Das symmetrische Iterationsverfahren

Das iterative Lösungsverfahren kann in ein symmetrisches Verfahren verwandelt werden. Dies ist nützlich, wenn der Algorithmus mit anderen Komponenten kombiniert wird, die ein symmetrisches Verfahren voraussetzen, wie z.B. das *Verfahren der konjugierten Gradienten* (*cg-Verfahren*) nach [37].

In Anlehnung an Kapitel 4.1.2 und [34] definieren wir ein symmetrisches Iterationsverfahren.

Definition 4.4 *Eine Iteration heißt* **symmetrisch** *falls für Hermitesche Matrizen A, auch die Matrix N der zweiten Normalform hermitesch ist.*

In dem Gauß-Seidel-Verfahren ist dies aber nicht der Fall, da der Glättungsschritt abhängig von der Reihenfolge der Variablen ist [34]. Es ist jedoch möglich, ein symmetrisches Gauß-Seidel-Verfahren zu implementieren, indem man in einem Vorwärtsschritt die Unbekannten von 1 bis n (s. Abb. 17) und in einem Rückwärtsschritt von n bis 1 (s. Abb. 27) berechnet.

Die *rückwärts durchgeführte Gauß-Seidel-Iteration* lautet in der zweiten Normalform:

$$x^{(m+1)} = x^{(m)} - (D - F)^{-1}(Ax^m - b) \tag{47}$$

```
proc rückwärts-Gauß-Seidel(A,b,x)
    for  i := n  downto  1  do
```
$$x_i^{(m+1)} := (b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(m)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(m+1)})/a_{ii}$$
```
endproc
```

Abbildung 27: **rückwärts-Gauß-Seidel-Iteration**

Definition 4.5 *Seien Φ^{GS} und Φ^{rGS} die normale bzw. die rückwärts ausgeführte Gauß-Seidel-Iteration. Das Produktverfahren*

$$\Phi^{symGS} := \Phi^{rGS} \circ \Phi^{GS} \tag{48}$$

definiert das **symmetrische Gauß-Seidel-Verfahren**.

Die zweite Normalform des symmetrischen Gauß-Seidel-Verfahrens ist

$$x^{(m+1)} = x^{(m)} - (D - F)^{-1} D (D - E)^{-1} (Ax^m - b). \tag{49}$$

In [34] wird nachgewiesen, daß Φ^{symGS} für positiv definite Matrizen A konvergiert und ein symmetrisches Iterationsverfahren ist.

In unserem baumartigen Iterationsverfahren aus Kapitel 4.3 kann diese Symmetrie erreicht werden, indem im top-down-Durchlauf ein Vorwärtsschritt und zusätzlich im bottom-up-Durchlauf ein Rückwärtsschritt auf den Unbekannten im Gebietsinneren der Baumknoten ausgeführt wird, wobei die Anzahl der Glättungsschritte in beiden Richtungen gleich ist.

Eine Darstellung des symmetrischen Algorithmus ist in Abb. (28) und (29) zu finden. In Phase 1 werden nun die Iterierten $u_i^{(m+\frac{1}{2})}$, $u_a^{(m+\frac{1}{2})}$ und $r_i^{(m+\frac{1}{2})}$ eingeführt, während die Iterierten $u_i^{(m+1)}$ und $r_i^{(m+1)}$ erst nach Phase 2 entstehen. In Phase 2 sind zusätzlich die Schritte (3), (4) und (5) auszuführen.

Durch den zusätzlichen Glättungsschritt in Phase 2 wird die Konvergenzrate im Vergleich zum unsymmetrischen Verfahren verbessert. Dies macht sich besonders bei den praktischen Berechnungen von Elastizitätsproblemen bemerkbar. Allerdings wird der Rechenaufwand pro Iterationsschritt fast verdoppelt.

Jedoch ist durch die Umwandlung des iterativen Algorithmus in ein symmetrisches Iterationsverfahren eine Kombination mit anderen iterativen Lösungsverfahren, z.B. dem cg-Verfahren, möglich. Solche Kombinationen werden von [34] und [62], speziell für Probleme der ebenen Elastizität, bei schlechten Konvergenzraten zur Verbesserung des Konvergenzverhaltens vorgeschlagen. In der vorliegenden Arbeit wird das Konvergenzverhalten durch den Einsatz der adaptiven Relaxation (s. Kap. 6) verbessert und deshalb keine dieser Kombinationen implementiert. Ein Einsatz dieser Verfahren ist aber durch die Symmetrisierung des Verfahren ermöglicht.

(1) falls Knoten äußere FG besitzt

empfange $u_a^{(m+\frac{1}{2})}$ vom Vaterknoten

berechne $\quad f_i' = f_i^{(m)} - K_{ia} \cdot u_a^{(m+\frac{1}{2})}$

(2) führe ν Vorwärts-Glättungsschritte Gauß-Seidel(K_{ii},f_i',$u_i^{(m)}$) aus, um $u_i^{(m+\frac{1}{2})}$ zu erhalten

(3) Speichere Änderung der Lösung $\quad r_i^{(m+\frac{1}{2})} = u_i^{(m+\frac{1}{2})} - u_i^{(m)}$

(4) Schicke $u^{(m+\frac{1}{2})}$ an die Söhne

(5) Wende Phase 1 rekursiv auf die Söhne an.

Abbildung 28: **PHASE 1 des symmetrischen Algorithmus**

(1) falls Knoten Söhne hat

Assembliere neue rechte Seite

$f^{(m+1)} = f^{(m)} + \sum_{s=1}^{K} r_a^{(m+1)}(s)$

(2) Speichere Änderungen von f_a, die von den Söhnen heraufgereicht wurden
$r_a^{(m+1)} = f_a^{(m+1)} - f_a^{(m)}$

(3) berechne $\quad f_i' = f_i^{(m+1)} - K_{ia} \cdot u_a^{(m+\frac{1}{2})}$

(4) führe ν Rückwärts-Glättungsschritte
rückwärts-Gauß-Seidel(K_{ii},f_i',$u_i^{(m+\frac{1}{2})}$) aus, um $u_i^{(m+1)}$ zu erhalten

(5) Speichere Änderung der Lösung $\quad r_i^{(m+1)} = r_i^{(m+\frac{1}{2})} + (u_i^{(m+1)} - u_i^{(m+\frac{1}{2})})$

(6) berechne Korrektur der rechten Seite
$r_a^{(m+1)} = r_a^{(m+1)} - K_{ai} \cdot r_i^{(m+1)}$

(7) Schicke $r_a^{(m+1)}$ an Vaterknoten

(8) Wende Phase 2 auf den Vaterknoten an

Abbildung 29: **PHASE 2 des symmetrischen Algorithmus**

5 Rekursive Substrukturierung unter Verwendung von hierarchischen Basen

In Kapitel 4 wurde gezeigt, daß sich das Verfahren der rekursiven Substrukturierung in ein iteratives Lösungsverfahren umwandeln läßt. Durch den Verzicht auf die statische Kondensation bleiben die Matrizen des Substrukturierungsbaumes schwach besetzt und benötigen, im Vergleich zu dem direkten Lösungsverfahren, deutlich weniger Speicherplatz. Insbesondere in den obersten Knoten des Strukturbaumes, wo die Matrizen sehr groß werden, ist der Gewinn an Speicherplatz beträchtlich.

Um diese Einsparung zu erhalten, müssen im Lösungsprozeß die Matrizen unverändert bleiben. Es dürfen lediglich die Lösungsvektoren und die rechten Seiten korrigiert werden. Bei Verwendung der in der FEM gebräuchlichen Knotenbasen, ist allerdings kein schnell konvergierendes Verfahren zu erzielen. Durch die lokale Begrenztheit der Ansatzfunktionen wirken sich Korrekturen der Lösung nur lokal aus. Deshalb sind mehrere Iterationen notwendig, um die Lösung von den Gebietsrändern in das Gebietsinnere zu übertragen. Diese Tatsache verstärkt sich noch bei wachsender Systemgröße.

Eine deutliche Verbesserung des Konvergenzverhalten kann durch die Einführung von hierarchischen Basen erzielt werden, deren Einsatz in Kombination mit der rekursiven Substrukturierung in diesem Kapitel untersucht wird.

5.1 Hierarchische Basen in der FEM

Wir betrachten ein uniformes Gitter auf dem Einheitsquadrat mit der Maschenweite $h = \frac{1}{2^k}$, das im folgenden als T_k bezeichnet wird. Dieser Fall ist für alle unsere Simulationen relevant, da wir ein zweidimensionales technisches Objekt in bedarfsterminale Bauteile vom Typ Viereck zerlegen (s. Kap. 3.2.1) [3], die sich durch Kongruenzabbildungen auf ein Einheitsquadrat abbilden lassen [9].

Sei G_k die Menge der Gitterpunkte von T_k. Der Raum S_k ist der $(2^k + 1)^2$-dimensionale Raum der stückweise bilinearen Funktionen auf dem Einheitsquadrat.

Die *Knotenbasis* B_k, die den Raum S_k aufspannt, besteht aus Funktionen $u \in S_k$, die jeweils an einem Gitterpunkt den Wert 1 und an allen anderen Gitterpunkten den Wert 0 annehmen.

In Abb. 30 sieht man das Gitter T_1, das sich aus vier Finiten Elementen zusammensetzt. Der Raum S_1 wird von neun Knotenbasen aufgespannt, von denen drei abgebildet sind. N_1 ist eine bilineare Basisfunktion für Viereckselemente. Die Basisfunktionen N_2 und N_3 entstehen durch das Zusammensetzen der vier Elemente in der Abbildung zu einem FE-Netz.

[3] Wir beschränken uns in der Arbeit auf Viereckszerlegungen. Alle Überlegungen in der Arbeit sind aber übertragbar auf Zerlegungen in denen Dreiecke vorkommen.

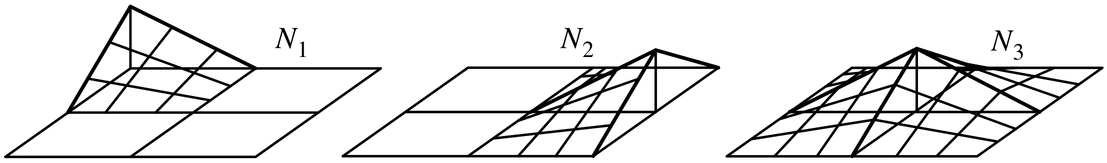


Abbildung 30: Bilineare Knotenbasen

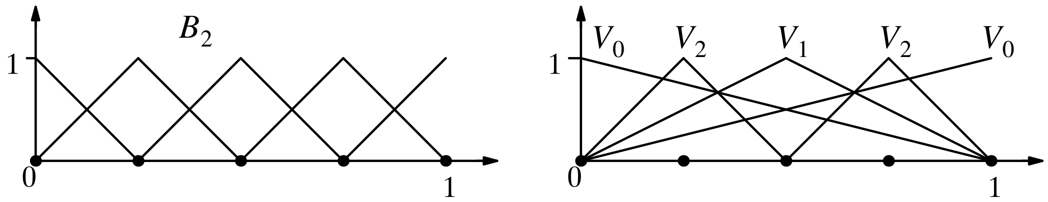


Abbildung 31: Knotenbasis B_2 und hierarchische Basis H_2 für den eindimensionalen Fall

Die hierarchischen Basen werden nun in Anlehnung an [71] und [6] eingeführt. Auf dem Gitter T_k läßt sich die *hierarchische Basis* H_k, die den gleichen Raum aufspannt wie die Knotenbasis B_k, wie folgt definieren:

$$\begin{aligned} H_0 &:= V_0 := B_0 \\ H_i &:= H_{i-1} \cup V_i = \cup_{j=0}^{i} V_j \end{aligned} \tag{50}$$

Dabei ist V_i die Teilmenge der Basisfunktionen aus B_i, die auf den zu S_{i-1} gehörigen Gitterpunkten verschwinden. Man kann H_i auch schreiben:

$$H_i := B_0 \oplus V_1 \oplus \cdots \oplus V_i \tag{51}$$

In Abb. 31 ist für den eindimensionalen Fall die Knotenbasis B_2 und die hierarchische Basis H_2 dargestellt, die den Raum S_2 der stückweise linearen Funktionen auf dem Intervall $[0,1]$ aufspannen.

Für den bei unseren numerischen Simulationen relevanten zweidimensionalen Fall wird in Abb. 32 der Raum S_1 betrachtet. Während sich die Knotenbasis B_1 aus den Basisfunktionen aus Abb. 30 zusammensetzt (vier Basen vom Typ $N1$, vier vom Typ $N2$, eine vom Typ $N3$), werden in der hierarchischen Basis H_1 die Basisfunktionen vom Typ N_1 durch Funktionen vom Typ V_0 ersetzt. Diesen Funktionen liegt als Trägermenge das gesamte Einheitsquadrat zugrunde. Alle anderen Basisfunktionen vom Typ N_2 und N_3 (s. Abb. 30) treten sowohl in B_1 als auch H_1 auf.

Die Einführung von hierarchischen Basen verbessert die Kondition des Gesamtsystems deutlich im Vergleich zur Verwendung von Knotenbasen [71]. Der Nachteil ist eine größere Überlappung zwischen den Basisfunktionen. Dies führt zu Einträgen in die Steifigkeitsmatrix zwischen Punkten, die nicht im selben Element liegen. Ein weiterer Unterschied ist, daß die Verschiebungen nicht mehr als Knotenverschiebungen interpretiert werden können.

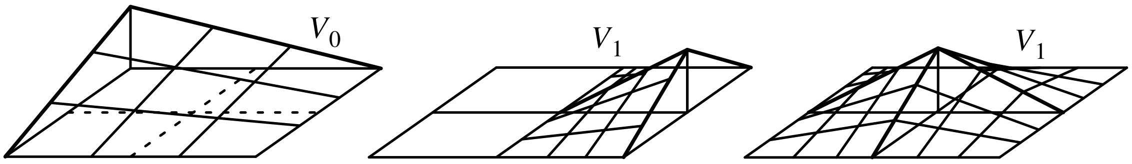


Abbildung 32: Hierarchische Basis H_1 für den zweidimensionalen Fall

Unabhängig von der Wahl der Basisfunktionen kann man die gesuchte Verschiebungsfunktion als Linearkombination der Basisfunktionen N_i darstellen:

$$u_n = \sum_{i=1}^{n} a_i N_i \tag{52}$$

Bei Verwendung von Knotenbasen können die Koeffizienten a_i als Knotenwerte identifiziert werden. Falls man jedoch das FE-Netz verfeinert, wird die Menge der Basisfunktionen, die den alten Lösungsraum aufgespannt haben, durch neue Basisfunktionen ersetzt und ergänzt. Wenn man z.B. m neue Basisfunktionen für m neu entstandene Gitterpunkte einführt, erhält man folgende Darstellung der Verschiebungsfunktion:

$$u_{n+m} = \sum_{i=1}^{n+m} b_i M_i \tag{53}$$

Dabei sind m neue Basisfunktionen einzuführen und ein großer Teil der alten Basisfunktionen durch neue zu ersetzen. Deshalb müssen in einem adaptiven FE-Verfahren die meisten Elementmatrizen und die globale Steifigkeitsmatrix komplett neu berechnet werden [27].

Es gibt jedoch die Möglichkeit, im Rahmen einer hierarchischen Verfeinerung, die Menge der alten Basisfunktionen durch m neue zu ergänzen, ohne die alten zu ändern. Mit einer hierarchische Verfeinerung erhält man

$$u_{n+m} = \sum_{i=1}^{n} c_i N_i + \sum_{i=n+1}^{n+m} c_i N_i = \sum_{i=1}^{n+m} c_i N_i, \tag{54}$$

wobei $N_i \quad (i = n+1, \dots, m)$ die neuen hierarchischen Basisfunktionen sind.

Bezeichnet $I_k : S_j \to S_k$ für $(j \geq k)$ die Interpolation durch (d.h. Projektion auf) S_k, so gilt für $u \in S_j$

$$u = I_j u = I_0 u + \sum_{k=1}^{j} (I_k u - I_{k-1} u). \tag{55}$$

Dadurch zerfällt die Lösung u in Funktionen, die aus verschiedenen Verfeinerungsstufen stammen und auf unterschiedlich großen Trägern definiert sind. Nur

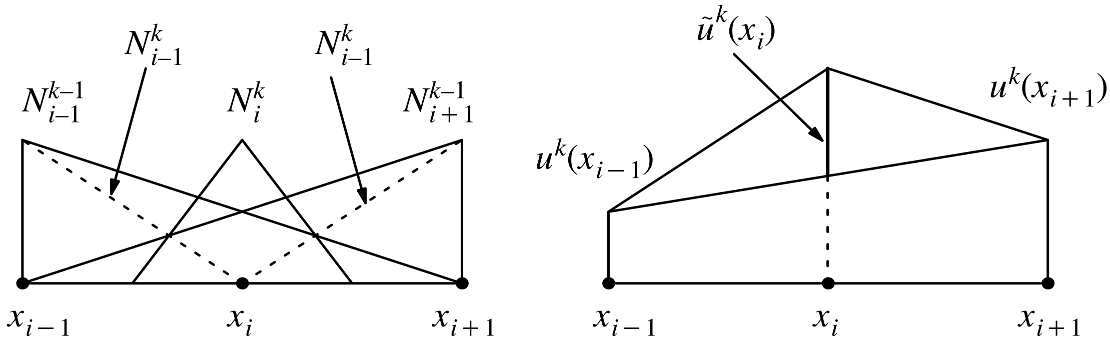


Abbildung 33: Hierarchische Basisfunktionen und hierarchischer Überschuß

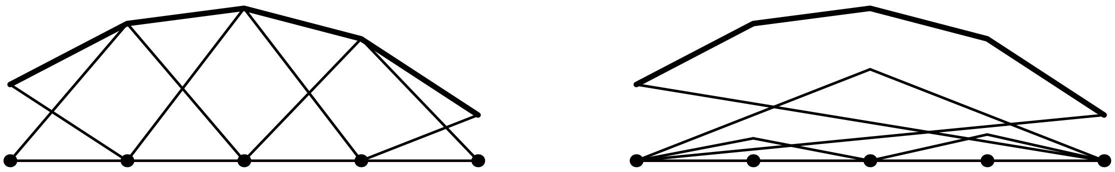


Abbildung 34: Darstellung einer Funktion in der Knotenbasis B_2 und der hierarchischen Basis H_2

die Verschiebungswerte zu Basisfunktionen aus dem Raum S_0 können als Knotenverschiebungen interpretiert werden. Alle anderen Verschiebungswerte sind sogenannte *hierarchische Überschüsse*.

Betrachten wir den eindimensionalen Fall von hierarchischen Basisfunktionen im Intervall $[x_{i-1}, x_{i+1}]$ aus Abb. 33. Die Knotenverschiebung $u^k(x_i)$ am Knoten x_i der Verfeinerungsstufe k setzt sich aus allen hierarchischen Verschiebungswerten $\tilde{u}$ von Basisfunktionen zusammen, die am Knoten x_i von Null verschieden sind.

$$u^k(x_i) = \tilde{u}^k(x_i) \cdot N_i^k(x_i) + \tilde{u}^k(x_{i-1}) \cdot N_{i-1}^{k-1}(x_i) + \tilde{u}^k(x_{i+1}) \cdot N_{i+1}^{k-1}(x_i) \quad (56)$$

Da in unserem Beispiel die hierarchischen Verschiebungswerte an den Intervallgrenzen mit den Knotenverschiebungen identisch sind, führt die Auswertung der Basisfunktionen zu

$$u^k(x_i) = \tilde{u}^k(x_i) + \frac{1}{2}(u^k(x_{i-1}) + u^k(x_{i+1})). \quad (57)$$

Dabei ist $\tilde{u}^k(x_i)$ der *hierarchische Überschuß* im Knoten x_i im Raum S_k. Bei Verwendung von Knotenbasen haben die angrenzenden Basisfunktionen N_{i-1}^k und N_{i+1}^k keinen Einfluß auf den Wert von $u^k(x_i)$.

Zum Vergleich der unterschiedlichen Repräsentation einer Funktion in Abhängigkeit der Basiswahl ist in Abb. 34 für den eindimensionalen Fall eine Funktion in Knotenbasis und hierarchischer Basis dargestellt. Man erkennt, daß die hierarchischen Verschiebungswerte in den Räumen S_i für steigendes i stark abfallen, während die Knotenverschiebungen in allen Gitterebenen die gleiche Größenordnung besitzen.

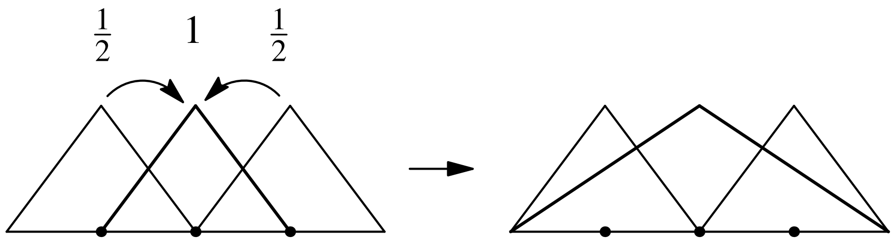


Abbildung 35: Erzeugung der Grobgitterbasen aus den Feingitterbasen

5.2 Hierarchisierung der Systemmatrizen

Während viele Iterationsverfahren die Idee der hierarchischen Basis zur Vorkonditionierung der Systemmatrizen verwenden (z.B. [33]), wird hier die hierarchische Basistransformation explizit durchgeführt. Der Wechsel von der Knotenbasis zur hierarchischen Basis erfolgt durch eine Transformation der Systemmatrizen, die in diesem Abschnitt beschrieben wird. Die Transformation gilt nur bei der Verwendung von bilinearen Basisfunktionen und der Teilungsstrategie der Viertelung. Bei anderen Basisfunktionen oder Teilungsstrategien ist die Transformation entsprechend zu modifizieren.

Wir betrachten nun eine Folge von verschachtelten Gittern, die in einem uniformen oder adaptiven Verfeinerungszyklus entstanden sind:

$$M_0 \rightarrow M_1 \rightarrow M_2 \rightarrow \cdots \rightarrow M_k \tag{58}$$

Sei G_k die Menge der Gitterpunkte dieser Gitter, die in Verfeinerungsstufe $s \leq k$ erzeugt worden sind.

Die Basisfunktion eines sogenannten *Grobgitterpunktes*, d.h. ein Punkt aus der Menge G_{k-1}, wird im eindimensionalen Fall aus drei benachbarten *Feingitterpunkten* aus G_k erzeugt. Dabei werden die Basisfunktionen wie in Abb. 35 mit den Faktoren $\frac{1}{2}$, 1 und $\frac{1}{2}$ gewichtet addiert. Im zweidimensionalen Fall werden die bilineare Basisfunktionen durch ein Tensorprodukt aus den eindimensionalen Basisfunktionen hergeleitet [11]. Deshalb ist auch bei den Gewichtsfaktoren ein Tensorprodukt zu bilden.

$$\begin{pmatrix} \frac{1}{2} \\ 1 \\ \frac{1}{2} \end{pmatrix} \cdot \begin{pmatrix} \frac{1}{2} & 1 & \frac{1}{2} \end{pmatrix} = \begin{pmatrix} \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ \frac{1}{2} & 1 & \frac{1}{2} \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \end{pmatrix} \tag{59}$$

Die Basisfunktion eines Grobgitterpunktes wird somit aus neun benachbarten Feingitterpunkten mit den Faktoren 1, $\frac{1}{2}$ und $\frac{1}{4}$ erzeugt (s. Abb. 36).

Bei der Transformation der Steifigkeitsmatrix von Knotenbasen auf hierarchische Basen ist analog vorzugehen. Zur Definition der Transformationsmatrix ist zunächst eine Nachbarschaftsrelation einzuführen.

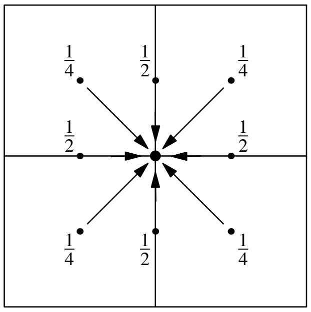


Abbildung 36: Erzeugung der Grobgitterbasen im zweidimensionalen Fall

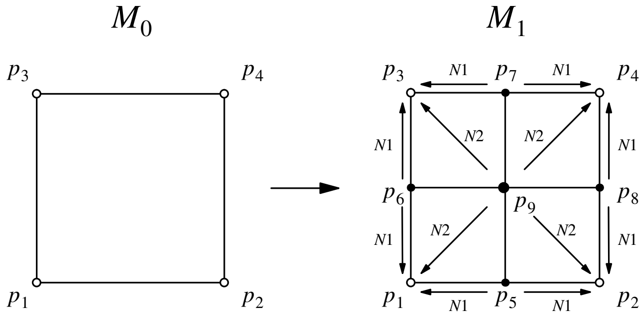


Abbildung 37: Nachbarschaft-Relation zwischen feinen Gitter M_1 und groben Gitter M_0

Eine h-Verfeinerung, wie in Kapitel 2.3.2 beschrieben, erzeugt eine Nachbarschaftsrelation zwischen den Gitterpunkten aus verschiedenen Verfeinerungsstufen. Dabei hat jeder Gitterpunkt, der durch h-Verfeinerung erzeugt wird, Nachbarn im gröberen Netz (s. Abb. 37). Es wird zwischen zwei Arten von Nachbarn unterschieden:

1-Nachbar(M,i): Der neu erzeugte Punkt i liegt auf dem Kantenmittelpunkt eines verfeinerten Vierecks. 1-Nachbar-Punkte sind die zwei Eckpunkte der geteilten Kante.

2-Nachbar(M,i): Der neu erzeugte Punkt i liegt im Mittelpunkt eines verfeinerten Vierecks. 2-Nachbar-Punkte sind die vier Eckpunkte des Vierecks.

In Abb. 37 sind die 1-Nachbarn der Punkte p_5, p_6, p_7 und p_8 sowie die 2-Nachbarn vom Punkt p_9 durch Pfeile gekennzeichnet. So ist z.B.
1-Nachbar$(M_1, p_7) = \{p_3, p_4\}$ und 2-Nachbar$(M_1, p_9) = \{p_1, p_2, p_3, p_4\}$.

Man erhält die hierarchische Matrix $\tilde{K}$ aus der entsprechenden Matrix bezüglich der Knotenbasen K durch eine Multiplikation mit einer Transformationsmatrix T:

$$\tilde{K} = T^T K T. \tag{60}$$

Dabei ist die Transformationsmatrix

$$T = P_k \cdot P_{k-1} \cdots P_1 \tag{61}$$

durch die Matrizen

$$\begin{gathered} P_i = I + R_i \\ R_i(j, 1\text{-Nachbar}(M_i, j)) = \tfrac{1}{2} \quad \text{und} \quad R_i(j, 2\text{-Nachbar}(M_i, j)) = \tfrac{1}{4} \end{gathered} \tag{62}$$

definiert, wobei j ein Gitterpunkt ist, der im Gitter M_i durch Verfeinerung neu entstanden ist. 1-Nachbar(M_i,j) und 2-Nachbar(M_i,j) sind die hierarchischen Nachbarn des Punktes j, die bereits in gröberen Gittern existieren. Alle anderen Einträge der Matrix R_i sind Null.

Auch die rechte Seite des Gleichungssystems $K \cdot u = f$ wird entsprechend transformiert und es ergibt sich insgesamt das System:

$$T^T K T \tilde{u} = T^T f \tag{63}$$

$$\tilde{K} \tilde{u} = \tilde{f} \tag{64}$$

Dabei repräsentiert $\tilde{u}$ die Verschiebungen in hierarchischer Darstellung. Falls man die Knotenverschiebungen berechnen will, muß man nach der Lösung des Gleichungssystems eine Rücktransformation (*Enthierarchisierung*) durchführen:

$$u = T \tilde{u} \tag{65}$$

Im Falle der Standard-FEM, wo ein Gesamtgleichungssystem aufgestellt wird, können die Gitterpunkte in Mengen eingeteilt werden, die zur selben hierarchischen Ebene (d.h. zur selben Verfeinerungsstufe) gehören. Anschließend werden die Mengen von groben zu feinen Netzen in aufsteigender Reihenfolge angeordnet und man erhält eine globale Transformationsmatrix T mit Blockstruktur (s. [27])

In dem Verfahren der rekursiven Substrukturierung ist keine globale Transformationsmatrix zur Hierarchisierung notwendig. Die Transformation auf die hierarchischen Basen kann in den einzelnen Baumknoten am Ende von Schritt 2_{RSI} nach dem Erstellen der Teilgebietsgleichungssysteme dezentral durchgeführt werden. Dadurch zerfällt die Transformation wie in (61) in k Transformationen, wobei jede Transformation einer Baumebene vollkommen parallel in den Baumknoten dieser Ebene durchgeführt werden kann.

Gestartet wird die Hierarchisierung in den Blattknoten. Da bei bilinearen Ansatzfunktionen jedes Element nur vier Knotenpunkte enthält, gibt es hier keine hierarchischen Nachbarpunkte und es erfolgt keine Hierarchisierung. In der nächsten Baumebene werden jeweils vier Elemente zu einem Baumknoten zusammengesetzt und man erhält ein FE-Netz wie in Abb. 37. Hier existieren alle Nachbarschaftsrelationen zwischen den Kantenmittelpunkten, dem Teilgebietsmittelpunkt und den Eckpunkten. Dadurch kann die Transformationsmatrix entsprechend (62) aufgestellt und die Matrixtransformation (60) durchgeführt werden.

Für die in Abb. 37 angegebene Numerierung der Knotenvariablen ergibt sich folgende Transformationsmatrix:

$$T = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 1 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & 0 & 1 & 0 \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{66}$$

Ein Baumknoten aus den oberen Baumebenen hat zusätzlich zu den bisher betrachteten neun Punkten weitere Stützpunkte auf dem Teilgebietsrand und auf den Separatoren. Diese Stützpunkte wurden bereits in den tieferen Baumebenen hierarchisch transformiert. Zur Hierarchisierung des aktuellen Baumknotens sind wieder nur die Kantenmittelpunkte, der Teilgebietsmittelpunkt und die Eckpunkte zu betrachten. Deshalb hat die Transformationsmatrix die gleiche Anzahl an außerdiagonalen Einträgen die von Null verschieden sind wie (66). Durch die Transformation (60) werden nur die Zeilen und Spalten der vier Eckpunkte geändert. Daher ist der Aufwand für diesen Schritt proportional zur Anzahl der Knotenstützpunkte d.h. von der Ordnung $O(n_i + n_a)$.

Die Transformation der Matrix nach dem Zusammenfügen von Teilgebieten beschränkt sich, unabhängig von der Größe der Teilgebiete, auf eine konstante Anzahl von Stützpunkten. Diese Stützpunkte und die zugehörigen Transformationen werden in den beiden Abbildungen 38 und 39 verdeutlicht.

In Abb. 38 werden zwei eindimensionale Elemente am Knoten p_2 zusammengesetzt. Durch die gewichtete Addition der Anteile der Steifigkeitsmatrix von Punkt p_2 auf seine hierarchischen Nachbarknoten entstehen die zwei Basisfunktionen für die Randpunkte, die sich über die gesamte Strecke von p_1 bis p_3 erstrecken. Lediglich die zu Punkt p_2 gehörige Knotenbasis bleibt erhalten. Nach der Lösung der Systemgleichung erhält man hierarchische Verschiebungswerte. Zur Ermittlung der Knotenverschiebungen muß die *Enthierarchisierung* (65)

$$u(p_2) = \tilde{u}(p_2) + \frac{1}{2}(u(p_1) + u(p_3)) \tag{67}$$

durchgeführt werden.

Im zweidimensionalen Fall sind zusätzlich zu den 1-Nachbarn der Kantenmittelpunkte die 2-Nachbarn des Teilgebietsmittelpunkts in die Transformation mit einzubeziehen (s. Abb. 39). Nach der gewichteten Addition der Matrixzeilen

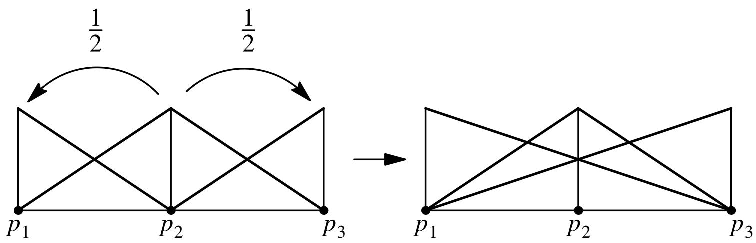


Abbildung 38: Transformation der Knotenbasis auf hierarchische Basis für den eindimensionalen Fall

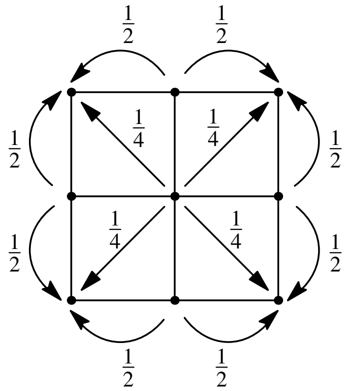


Abbildung 39: Transformation der Knotenbasis auf hierarchische Basis für den zweidimensionalen Fall

von Feingitterpunkten auf Grobgitterpunkte entstehen in den Eckpunkten Basisfunktionen, deren Trägermenge das gesamte Teilgebiet umfaßt, wie z.B. die Basisfunktion vom Typ V_0 in Abb. 32.

Die Einführung von hierarchischen Basen hat keinen Einfluß auf die in Kap. 3.2.6 beschriebene Schnittstelle von ARESO zur Erzeugung der Elementmatrizen. Diese Matrizen basieren nach wie vor auf Knotenbasen. Sobald die Finiten Elemente in der nächsten Baumebene zusammengesetzt werden, verwandelt die hierarchische Transformation einige dieser Knotenbasen in hierarchische Basen mit einem vergrößerten Träger.

Die hierarchischen Basen werden bottom-up erzeugt, da Schritt 2_{RSI} in einem bottom-up-Baumdurchlauf erfolgt. Insgesamt wird ausgehend von den neu entstandenen Stützpunkten des letzten Verfeinerungsschrittes bis zu den Gitterpunkten der Startdiskretisierung ein Finite-Element-Raum bestehend aus hierarchischen Basen erzeugt. Dieser Raum entspricht dem in [71] vorgestellten FE-Raum. Die dezentrale Hierarchisierung setzt keine regelmäßigen Gitter voraus und ist deshalb sehr gut für unser adaptives Verfahren geeignet.

5.3 Das iterative Verfahren mit hierarchischen Basen

Die Einführung von hierarchischen Basen bewirkt einige zusätzliche Operationen in dem iterativen Verfahren aus Kapitel 4. Die Transformation der Knotengleichungssysteme findet am Ende von Schritt 2_{RSI} statt, der damit aus vier Teilschritten besteht:

(1) Aufstellung von Matrix und Lastvektor.

(2) Trennung von inneren und äußeren Freiheitsgraden.

(3) Behandlung der Randbedingungen und der inkompatiblen Punkte.

(4) Transformation der Matrix $(T^T \cdot K \cdot T)$ und der rechten Seite $(T^T \cdot f)$.

Aus Effizienzgründen wird die Transformationsmatrix T nicht explizit aufgestellt. Bei der Assemblierung wird eine sogenannte *Hierarchisierungstabelle* erzeugt, in der alle an der Hierarchisierung beteiligten Punkte eingetragen sind. Diese Tabelle hat eine feste Größe, da in jedem Baumknoten maximal neun Punkte beteiligt sind. Für das Beispiel aus Abb. 37 ergibt sich folgende Tabelle. In der

p_1	p_5	p_6	p_9
p_2	p_5	p_8	p_9
p_3	p_6	p_7	p_9
p_4	p_7	p_8	p_9

Tabelle 1: Hierarchisierungstabelle

(1-3) s. Abb. 22, aber ersetze u durch die Verschiebungen in hierarchischer Darstellung $\tilde{u}$

(3.5) enthierarchisiere Lösungsvektor: $u^{(m+1)} = T\tilde{u}^{(m+1)}$

(4-5) s. Abb. 22

Abbildung 40: **PHASE 1 des iterativen Algorithmus mit hierarchischen Basen**

ersten Spalte enthält die Tabelle die Grobgitterpunkte, in der zweiten und dritten Spalte Punkte, die mit dem Faktor $\frac{1}{2}$ und in der letzten Spalte Punkte die mit dem Faktor $\frac{1}{4}$ addiert werden. Damit können sämtliche hierarchische Transformationen ohne explizite Matrixmultiplikationen durch gewichtete Additionen der betreffenden Zeilen, Spalten und Komponenten der beteiligten Matrizen und Vektoren, durchgeführt werden.

Der letzte Teilschritt von 2_{RSI} wird bei allen Baumknoten außer dem Wurzelknoten durchgeführt. Der Wurzelknoten entspricht dem Startnetz der Berechnung, das aus einem oder mehreren viereckigen disjunkten Teilgebieten besteht. Er kann nur hierarchisiert werden, wenn das Berechnungsgebiet eine rechteckige Form besitzt (s. z.B. Abb. 3). Bei komplizierteren Berechnungsgebieten (s. z.B. Abb. 4) ist eine Hierarchisierung nicht möglich oder zumindest nicht mit einem allgemeinen Schema, wie in Kapitel 5.2 beschrieben, realisierbar. Deshalb wird in diesen Fällen auf eine Hierarchisierung des Wurzelknotens verzichtet.

Bei Blattknoten, in denen es keine hierarchische Nachbarbeziehung zwischen den Stützpunkten gibt, entspricht die Transformationsmatrix T der Einheitsmatrix.

Nach Schritt 2_{RSI} haben die Knotengleichungssysteme eine hierarchische Form. In dem anschließenden Zyklus von einigen Baumdurchläufen in Schritt 3_{RSI} muß bei jedem Wechsel zwischen den Baumebenen die Lösung hierarchisiert bzw. enthierarchsiert werden. Zudem ist jede Korrektur der rechten Seite der entsprechenden hierarchischen Transformation zu unterwerfen.

In den Abbildungen 40 und 41 sind die Unterschiede des Algorithmus unter Verwendung von hierarchischen Basen bezüglich der Grundversion aus Kapitel 4.3 zu sehen.

Alle Operationen in Phase 1 finden auf dem hierarchisierten Knotengleichungssystem $\tilde{K}\tilde{u} = \tilde{f}$ statt. Deshalb sind für die ersten drei Teilschritte, Verschiebungen in hierarchischer Darstellung zu verwenden. Am Ende von Phase 1 wird die Lösung an die Söhne weitergegeben, in denen die hierarchische Transformation des Vaterknotens nicht durchgeführt wurde. Darum ist der Lösungsvektor vor der Übergabe zu enthierarchisieren:

(0) hierarchisiere Lösungsvektor: $\tilde{u}^{(m+1)} = T^{-1}u^{(m+1)}$

(1) falls Knoten Söhne hat

Assembliere neue rechte Seite

$\Delta f^{(m)} = \sum_{s=1}^{K} r_a^{(m+1)}(s)$

$f^{(m+1)} = f^{(m)} + T^T \Delta f^{(m)}$

(2-5) s. Abb. 25

Abbildung 41: **PHASE 2 des iterativen Algorithmus mit hierarchischen Basen**

In dem Beispiel aus Abb. 37 und mit Tabelle 1 sind z.B. die hierarchischen Nachbarpunkte vom Punkt p_7 aus der ersten Spalte der letzten beiden Tabellenzeilen ersichtlich. Daraus ergibt sich die Enthierarchisierung

$$u(p_7) = \tilde{u}(p_7) + \frac{1}{2}(u(p_3) + u(p_4)). \tag{68}$$

Dieser Schritt muß am Anfang von Phase 2 wieder invertiert werden durch

$$\tilde{u} = T^{-1}u, \tag{69}$$

um auf dem Knotengleichungssystem wieder mit Verschiebungen in hierarchischer Darstellung rechnen zu können. Dazu ist für alle Kantenmittelpunkte und Teilgebietsmittelpunkte der hierarchische Überschuß zu bestimmen. Für den Punkt p_7 in unserem Beispiel berechnet sich der hierarchische Überschuß aus

$$\tilde{u}(p_7) = u(p_7) - \frac{1}{2}(u(p_3) + u(p_4)). \tag{70}$$

Die Matrix T^{-1} zur Hierarchisierung des Lösungsvektor ergibt sich aus T, indem man alle Einträge außerhalb der Diagonalen mit -1 multipliziert. Dadurch ist keine explizite Invertierung der Transformationsmatrix notwendig und auch die Hierarchisierung von u in (69) kann mit Hilfe der Tabelle 1 durchgeführt werden.

Der zweite zusätzliche Berechnungsschritt in Phase 2 betrifft die rechten Seiten. Auch die Korrekturen der rechten Seite Δf müssen derselben hierarchischen Transformation unterworfen werden wie die rechte Seite in Schritt 2_{RSI}:

$$\Delta\tilde{f} = T^T \Delta f \tag{71}$$

Im Vergleich zur Grundversion ohne hierarchische Basen kommen in beiden Phasen eine konstante Anzahl von Operationen pro Transformation des Verschiebungsvektors bzw. der rechten Seite hinzu.

5.4 Kondition der hierarchisierten Matrizen

Die Kondition der Systemmatrizen ist entscheidend für die Konvergenzgeschwindigkeit von iterativen Lösungsverfahren. Zudem folgt bei einer niedrigen Kondition aus einem kleinem Residuum der Iterationslösung, daß es sich um eine gute Näherungslösung handelt.

Eine Verwendung von hierarchischen Basisfunktionen in dem Verfahren der rekursiven Substrukturierung hat sowohl Auswirkungen auf die Kondition als auch auf die Besetzungsstruktur der Matrizen des Substrukturierungsbaumes. Durch die Hierarchisierung der Matrizen verlieren sie zwar ihre nahezu diagonale Gestalt, aber die Nichtdiagonaleinträge sind so schwach, daß sich die Kondition des Gesamtsystems sogar verbessert [27].

Die Steifigkeitsmatrix eines FE-Ansatzraumes der aus Knotenbasen besteht, hat eine Kondition, die bei zunehmender Verfeinerung des Netzes stark anwächst. Bei einem zweidimensionalen uniformen Netz mit der Gitterweite h wächst die Konditionszahl mit $O((\frac{1}{h})^2)$ [71].

Verwendet man dagegen die in Kapitel 5.2 vorgestellten hierarchischen Basen als FE-Ansatzfunktionen, so ist im zweidimensionalen Fall die Kondition der hierarchischen Matrix mit $O(log(\frac{1}{h})^2)$ beschränkt [71]. Man kann die Kondition der Matrix auch in Bezug zu der Anzahl der Gitterebenen, bzw. zu der Anzahl der Ebenen des Substrukturierungsbaumes setzen. Bei einem uniformen Netz mit j Ebenen ist die Kondition der hierarchisierten Matrix mit $O(j^2)$ beschränkt [28].

Unter geeigneten Voraussetzungen läßt sich damit ein iterativer Algorithmus mit dem cg-Verfahren implementieren, der ein uniform verfeinertes Gesamtsystem mit n Unbekannten bei einer gegebenen Genauigkeit in $O(n \cdot log(n))$ Operationen löst (s. [71]). Zwar bedeutet dies ein logarithmisches Wachstum für die Zahl der benötigten Iterationen, aber im Gegensatz zu anderen schnelleren iterativen Verfahren, z.B. Mehrgitterverfahren, werden hier keine gravierende Einschränkungen bezüglich des Problems und der Diskretisierung gemacht. In den Beweisen von [71] werden keine starke elliptische Regularität, kein geometrisches Wachstum der Unbekannten von Gitterebene zu Gitterebene und keine Quasiuniformität der Netzverfeinerung gefordert und zusätzlich geometrisch komplexe Gebiete erlaubt. Deshalb ist das Verfahren für adaptiv verfeinerte FE-Netze und komplizierte technische Objekte sehr gut geeignet.

Die günstige Kondition des hierarchisierten Systems läßt auch die Verwendung des Gauß-Seidel-Verfahrens zur Lösung der FE-Gleichungssysteme zu. In [6] wird das Gauß-Seidel-Verfahren auf das hierarchische System angewandt und analysiert. Es wird gezeigt, daß unter gewissen Voraussetzungen, die Zahl der Iterationen, um das Gleichungssystem bis auf eine vorgegebene Genauigkeit zu lösen, von der Ordnung $O(log(n)^2)$ ist. Damit ist die gesamte Anzahl an Rechenoperationen von $O(n \cdot log(n)^2)$.

Da wir in unserem iterativen Algorithmus das Gauß-Seidel-Verfahren auf hierarchisierte Systemmatrizen anwenden, folgt aus den obigen Aussagen über die

Kondition der Systemmatrizen, bei uniformer Verfeinerung von zweidimensionalen FE-Problemen, ein logarithmisches Wachstum der Iterationszahlen. Bei einem Problem mit n Unbekannten sind $O(log(n)^2)$ Baumdurchläufe erforderlich. Mit einem Speicherbedarf von $O(n)$ (s. Kap. 7) ergibt sich damit die gesamte Anzahl von arithmetischen Operationen zu $O(n \cdot log(n)^2)$. Diese Zahlen werden in den praktisch durchgeführten numerischen Simulationen bestätigt (s. Kap. 9).

Ein logarithmisches Wachstum der Kondition der Systemmatrizen ermöglicht iterative Lösungsverfahren die deutlich geringeren Rechenaufwand als direkte Methoden benötigen. Die optimale Anzahl von Iterationen mit $O(1)$, die z.B. mit Mehrgitterverfahren für bestimmte Anwendungen erzielt werden kann, wird aber nicht erreicht. In [29] wird die hierarchische Basis erweitert zu einem Erzeugendensystem . Dieses System führt zu Matrizen deren Kondition mit $O(1)$ unabhängig von der Maschenweite des FE-Netzes beschränkt sind. Die Matrizen können zwar singulär werden, was jedoch bei iterativen Lösungsverfahren wie dem cg-Verfahren oder dem Gauß-Seidel-Verfahren keine Probleme bereitet. Das Erzeugendensystem eignet sich auch in Verbindung mit Gebietszerlegungsverfahren ([29]) und scheint ein geeigneter Zugang zu einem iterativen Algorithmus in der rekursiven Substrukturierung mit optimaler Konvergenz. Dies konnte aber in dieser Arbeit nicht mehr berücksichtigt werden und es müssen noch genauere Untersuchungen stattfinden.

Die theoretisch erreichbaren Komplexitäten der Algorithmen können in der Praxis nur erreicht werden, wenn die Transformation der Matrizen und die Glättungsoperationen auf den Matrizen mit dem Aufwand $O(n)$ durchführbar sind. Für die Transformationsmatrizen T muß die Anzahl der Einträge beschränkt sein. Das gilt, wenn die Anzahl der hierarchischen Nachbarpunkte eines Stützpunktes unabhängig von der Anzahl der Unbekannten des Gesamtsystems ist [27]. Diese Art von Lokalität wird bei bei unserem Verfahren erfüllt (s. Kap. 5.2).

5.5 Inkompatible Knoten und Randbedingungen

Durch Verwendung von hierarchischen Basen gibt es gewisse Unterschiede bei der Behandlung der inkompatiblen Knoten und der Randbedingungen im Vergleich zum direkten Lösungsalgorithmus (s. Kap.3.2.2).

Behandlung der inkompatiblen Knoten

Es besteht ein sehr enger Zusammenhang zwischen der Hierarchisierung der Knotengleichungssysteme und der Behandlung der inkompatiblen Stützpunkte, die sich in zwei Teilschritte gliedern läßt.

(1) Transformation der Matrix

(2) Elimination der inkompatiblen Punkte

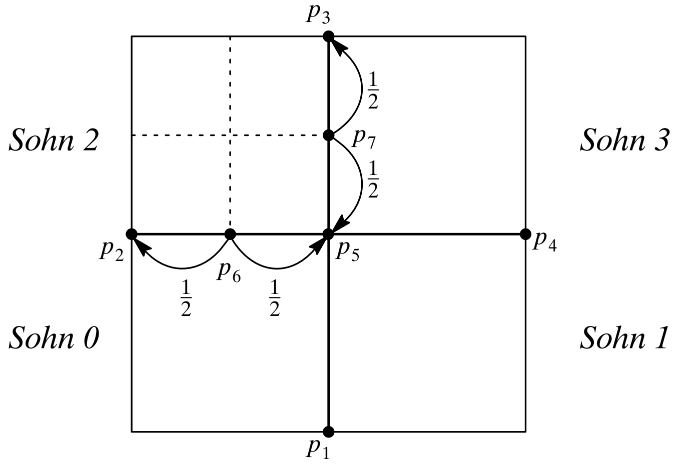


Abbildung 42: Behandlung der inkompatiblen Punkte

Der erste Teilschritt kann wie die Hierarchisierung der Matrizen als Transformation des Gleichungssystems betrachtet werden:

$$K' = T_i^T K T_i \tag{72}$$

$$f' = T_i^T f \tag{73}$$

Dabei ist die Transformationsmatrix T_i für einen Baumknoten der Ebene k wie folgt definiert:

$$\begin{aligned} T_i &= I + S_k \\ S_k(j, 1\text{-Nachbar}(M_{k+1}, j)) &= \tfrac{1}{2}, \end{aligned} \tag{74}$$

wobei j ein inkompatibler Stützpunkt ist, der im Gitter M_{k+1}, d.h. in der Baumebene $k+1$, erzeugt wurde und nun in der Baumebene k eliminiert wird.

Das Beispiel in Abb. 42 zeigt einen Baumknoten, der aus vier Söhnen zusammengesetzt wird und die zwei inkompatiblen Punkte p_6 und p_7 enthält. Die angegebene Numerierung der Knotenvariablen führt zu folgender Transformationsmatrix:

$$T_i = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 & \frac{1}{2} & 1 & 0 \\ 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & 1 \end{pmatrix} \tag{75}$$

Der Transformationsschritt (72) überträgt die strukturelle Information der inkompatiblen Stützpunkte im Teilgebietsinneren auf die angrenzenden Nachbarpunkte. Anschließend ist im zweiten Teilschritt der inkompatible Stützpunkt zu

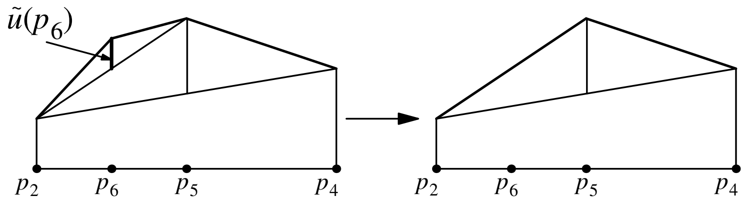


Abbildung 43: Behandlung des inkompatiblen Punktes p_6 auf der Kante von Punkt p_2 nach p_4

eliminieren, d.h. die zugehörige Matrixzeile und Matrixspalte aus dem Knotengleichungssystem K' zu löschen.

Betrachten wir den inkompatiblen Punkt p_6 in Abb. 42. Die Transformation (72) addiert die Matrixzeilen und -spalten von Punkt p_6 mit dem Faktor $\frac{1}{2}$ auf die angrenzenden kompatiblen Punkte p_2 und p_5. Dadurch enstehen zwei Basisfunktionen, die sich entlang der Kante von p_2 bis p_5 erstrecken und der Punkt p_6 kann aus dem Knotengleichungssystem entfernt werden. Als Ergebnis liegt ein konformer Funktionenraum entlang der Kante zwischen Sohn 0 und Sohn 2 vor, der die Stetigkeit der Lösung garantiert.

Der erste Teilschritt zur Behandlung der inkompatiblen Stützpunkte entspricht exakt der Basistransformation von der Knotenbasis auf die hierarchische Basis. Diese Transformation ist bei Verwendung von hierarchischen Basen bereits in den Sohnknoten vollzogen worden. Deshalb erfordern die inkompatiblen Punkte bei Verwendung von hierarchischen Ansatzräumen keine zusätzlichen Matrixtransformationen sondern lediglich ihre Elimination.

Anschaulich bedeutet ein inkompatibler Punkt, daß seine Verschiebung durch die angrenzenden Nachbarknoten vorgeschrieben und der Punkt auf der Kante zwischen diesen Punkten festgehalten wird. In unserem Beispiel gilt

$$u(p_6) = \frac{1}{2}(u(p_2) + u(p_5)). \tag{76}$$

Nach (57) ist dies gleichbedeutend damit, daß der hierarchischer Überschuß $\tilde{u}(p_6)$ gleich Null ist.

In Abb. 43 ist die hierarchische Darstellung der Lösung entlang der Kante von Punkt p_2 nach p_5, vor und nach der Behandlung des inkompatiblen Punktes p_6 zu sehen.

Behandlung der Randbedingungen

Die Verwendung von hierarchischen Basen hat auch einen Unterschied in der Behandlung der Randbedingungen im Vergleich zum direkten Lösungsalgorithmus (s. Kap. 3.2.2) zur Folge. Wir betrachten wieder nur den Fall, daß die Randbedingungen auf den geometrischen Rändern des Gesamtgebietes vorliegen.

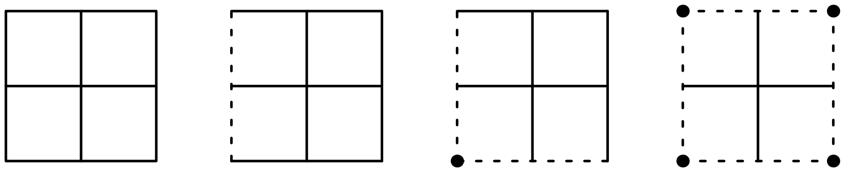


Abbildung 44: Randknoten als zusätzliche Stützpunkte nach außen bei Verwendung von hierarchischen Basen

Die Eckpunkte eines Teilgebiets sind Träger von hierarchischen Basisfunktionen, die das gesamte Teilgebiet überdecken. Sie sind von entscheidender Bedeutung für die Konvergenz des Verfahrens und müssen an die nächsthöhere Baumebene weitergereicht werden. Das erfolgt durch eine Zuordnung der Freiheitsgrade der Eckpunkte zu den äußeren Freiheitsgraden der Baumknoten. Falls für den Eckpunkt eine Neumann-Randbedingung spezifiziert ist, wird sie zwar dezentral in das Gleichungssystem integriert, der Stützpunkt aber nicht dezentral im Baumknoten eliminiert. Eckpunkte mit Dirichlet Randbedingungen werden solange an den Vaterknoten gereicht bis sie im Teilgebiet des Vaterknotens keine Eckpunkte mehr sind. In dieser Baumebene werden die Randbedingungen behandelt und die Freiheitsgrade der entsprechenden Stützpunkte eliminiert.

Betrachtet man beispielsweise die Situation in Abb. 14 dann werden die Stützpunkte, die bisher in Ebene 3 eliminiert worden sind, bei der hierarchischen Version bis zum Wurzelknoten nach oben gereicht.

In Abb. 44 sind die vier verschiedenen Typen von Baumknoten zu sehen, die in einem Substrukturierungsbaum auftreten. Es gibt Baumknoten im Gesamtgebietsinneren, Baumknoten mit einer Teilgebietskante auf dem Gesamtgebietsrand, Baumknoten mit zwei Teilgebietskanten auf dem Gesamtgebietsrand sowie den Wurzelknoten. Die besonders hervorgehobenen Punkte sind die Stützpunkte, die jetzt erst zentral im Wurzelknoten eliminiert werden, die gestrichelten Linien stellen den Gesamtgebietsrand dar. Dabei zeigt sich für die ersten beiden Fälle, daß sowieso alle Eckpunkte als äußere Freiheitsgrade behandelt und zum Vaterknoten weitergereicht werden. In den beiden anderen Fällen gibt es einen bzw. vier Stützpunkte, die nicht mehr dezentral eliminiert werden. Das führt pro Baumknoten zu maximal einer zusätzlichen Unbekannten nach außen und im Wurzelknoten zu vier zusätzlichen Unbekannte (falls es sich um Neumann-Randbedingungen handelt). Dies kann in seiner Auswirkung auf den Speicher- und Rechenaufwand des Gesamtproblems vernachlässigt werden.

5.6 Hierarchisierung bei biquadratischen Ansatzfunktionen

Das bisherige Verfahren unter Verwendung von hierarchischen Basen basiert auf bilinearen Basisfunktionen. Eine Erweiterung auf biquadratische Basisfunktionen

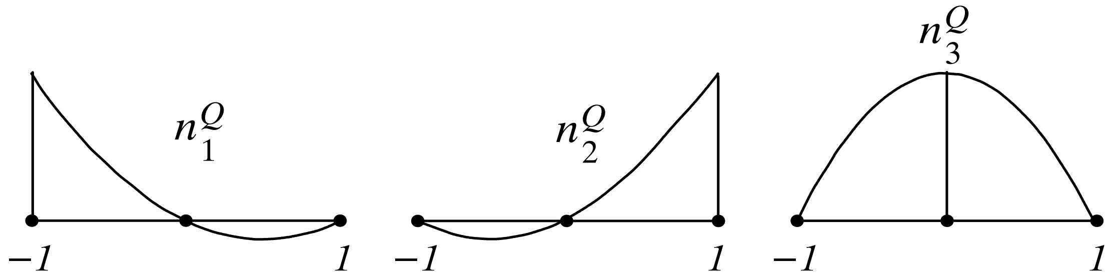


Abbildung 45: Quadratische Basisfunktionen für den eindimensionalen Fall

ist aber möglich und wird in diesem Kapitel behandelt.

Wir betrachten zunächst den eindimensionalen Fall. Als Basisfunktionen werden die eindimensionalen Lagrangeschen Interpolationspolynome [42] verwendet (s. Abb. 45), die im Intervall [-1,1] wie folgt definiert sind.

$$\begin{aligned} n_1^Q(x) &= \frac{1}{2}(x^2 - x) \\ n_2^Q(x) &= \frac{1}{2}(x^2 + x) \\ n_3^Q(x) &= (1 - x^2) \end{aligned} \tag{77}$$

Im Unterschied zu eindimensionalen Elementen mit linearen Ansatzfunktionen besteht ein Finites Element hier aus drei Knotenpunkten und drei Basisfunktionen.

Für die Hierarchisierung dieser Basisfunktionen gibt es zwei Varianten. Die erste Möglichkeit ist analog zum linearen Fall (s. Abb. 35). Bei der Einführung eines neuen Knotenpunkts durch Netzverfeinerung werden die Basisfunktionen der Grobgitterpunkte durch Addition von benachbarten Feingitterbasen erzeugt [11]. Eine kurze Berechnung ergibt die Gewichtsfaktoren aus Abb. 46, mit denen die Grobgitterbasen der Punkte p_1, p_3 und p_5 aus den Feingitterbasen erzeugt werden.

Bei quadratischen Basisfunktionen ändert sich sowohl die Anzahl der hierarchischen Nachbarknoten als auch die Gewichtsfaktoren. In Abb. 46 sieht man die drei hierarchischen Nachbarknoten p_1, p_3 und p_5 des Knotenpunkts p_2 und die Faktoren mit denen die hierarchische Transformation durchgeführt wird. Wenn die Basis des Feingitterpunktes p_4 ebenfalls mit den Faktoren $-\frac{1}{8}$, $\frac{3}{4}$ und $\frac{3}{4}$ auf die hierarchischen Nachbarknoten p_1, p_3 und p_5 addiert wird, ergibt sich als Ergebnis eine hierarchische Basis, die ausschließlich aus quadratischen Ansatzfunktionen besteht (s. Abb. 46, untere Hälfte).

Die zweite Möglichkeit besteht darin, zuerst die quadratische Basis in einen linearen und einen quadratischen Anteil zu splitten [11]. Die linearen Basisfunktionen $n_1^L(x)$ und $n_2^L(x)$ lassen sich durch die quadratischen Basisfunktionen aus-

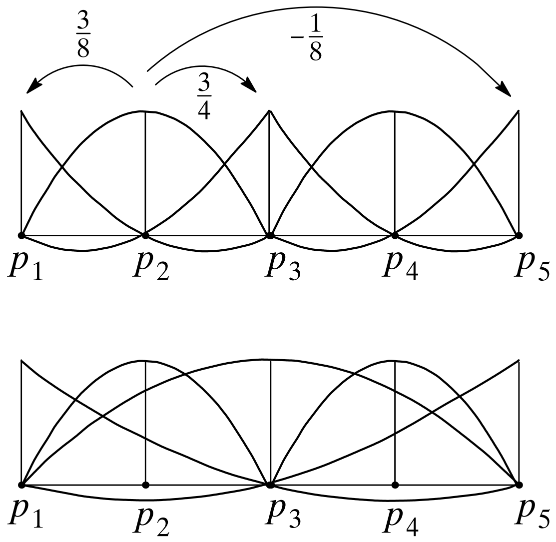


Abbildung 46: Hierarchische Transformation der quadratischen Basisfunktionen

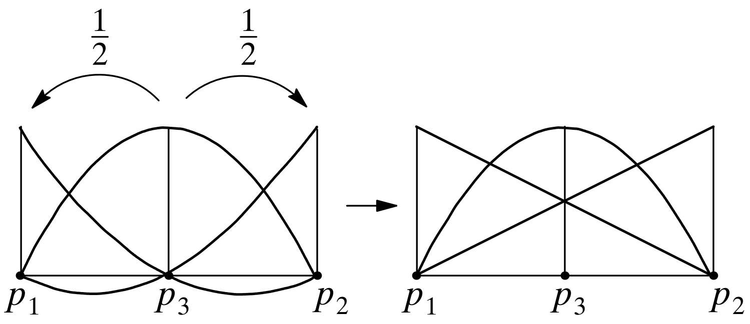


Abbildung 47: Transformation der quadratischen Basis auf Elementebene

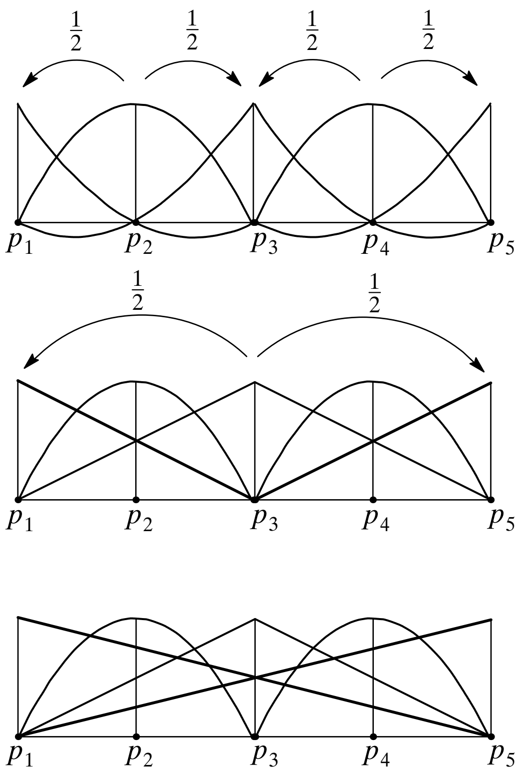


Abbildung 48: Transformation der quadratischen Basis in zwei Stufen auf eine hierarchische Basis mit linearen und quadratischen Basisfunktionen

drücken. Z.B. ist

$$
\begin{aligned}
n_1^L(x) &= \frac{1}{2}(-x+1) \\
&= \frac{1}{2}(x^2-x)+\frac{1}{2}(1-x^2) = n_1^Q(x)+\frac{1}{2}n_3^Q(x). \qquad (78)
\end{aligned}
$$

(79)

Bei dieser Variante wird die Basis bereits auf Elementebene in zwei lineare ($n_1^L(x)$ und $n_2^L(x)$) und eine quadratische Ansatzfunktion ($n_3^Q(x)$) transformiert (s. Abb. 47). In der nächsthöheren Hierarchieebene werden nur noch die linearen Basisfunktionen transformiert und somit deren Träger vergrößert.

In Abb. 48 ist dargestellt, wie bei der zweiten Variante zwei Finite Elemente zusammengesetzt und hierarchisiert werden. Zuerst werden die quadratischen Basen der Elemente durch eine gewichtete Addition der zu p_2 bzw. p_4 gehörigen Basisfunktionen auf die hierarchischen Nachbarpunkte in lineare und quadratische Basisfunktionen aufgeteilt. Anschließend werden die beiden Elemente zusammengesetzt und die zu p_1, p_3 und p_5 gehörigen Basisfunktionen hierarchisiert. Als Ergebnis erhält man eine Basis bestehend aus zwei quadratischen und drei linearen Basen.

Die erste Möglichkeit hat den Nachteil, daß sich die Hierarchisierung auf Punkte aus benachbarten Elementen auswirkt. Die Transformation der Basen von p_1, p_2 und p_3 in Abb. 46 wirkt sich z.B. auf die Basis von p_5 aus, die im angrenzenden Element liegt. In dem Verfahren der rekursiven Substrukturierung herrscht aber in jedem Baumknoten nur eine lokale Sichtweise. Benachbarte Teilgebiete auf gleicher Baumebene können deshalb nicht gegenseitig ihre Knotengleichungssysteme modifizieren. Dadurch ist die erste Variante in unserem Verfahren nicht implementierbar. Die zweite Variante dagegen beschränkt ihre Transformation nur auf Punkte eines Elements oder eines Teilgebiets. Diese Transformationen können in jedem Baumknoten problemlos durchgeführt werden.

Im zweidimensionalen Fall betrachten wir deshalb nur noch die zweite Variante der Hierarchisierung. Während bei bilinearen Basisfunktionen ein Viereckelement aus vier Stützpunkten besteht, besitzen biquadratische Viereckselemente neun Stützpunkte und neun Basisfunktionen. Die neun Basisfunktionen im Einheitsquadrat $[-1,1]\times[-1,1]$ erhält man durch Produktbildung aus den drei eindimensionalen Lagrangeschen Interpolationspolynomen (77). Im folgenden sind vier dieser Funktionen, einschließlich der entsprechenden eindimensionalen Funktionen, aus deren Produkt sie gebildet sind, angeführt.

$$\begin{aligned}
N_9^Q(x,y) &= (1-x^2)(1-y^2) = n_3^Q(x)\cdot n_3^Q(y)\\
N_5^Q(x,y) &= \frac{1}{2}(1-x^2)(1+y) - \frac{1}{2}N_9^Q(x,y) = n_3^Q(x)\cdot n_2^Q(y)\\
N_8^Q(x,y) &= \frac{1}{2}(1-y^2)(1+x) - \frac{1}{2}N_9^Q(x,y) = n_3^Q(y)\cdot n_2^Q(x)\\
N_1^Q(x,y) &= \frac{1}{4}(1+x)(1+y) - \frac{1}{2}N_5^Q(x,y) - \frac{1}{2}N_8^Q(x,y) - \frac{1}{4}N_9^Q(x,y)\\
&= n_2^Q(x)\cdot n_2^Q(y) \qquad (80)
\end{aligned}$$

Dabei gibt es drei verschiedene Typen, die in den Abb. 49, 50 und 51 zu sehen sind: Basisfunktionen von Eckpunkten (z.B. N_1^Q), von Seitenmittelpunkten (z.B. N_5^Q,N_8^Q) und vom Elementmittelpunkt (N_9^Q).

Der erste Schritt der Hierarchisierung ist die Aufspaltung der Basisfunktionen in bilineare und in biquadratische Anteile und findet auf Elementebene statt. Für die bilineare Basisfunktion $N_1^L(x,y)$ in dem Eckpunkt $(1,1)$ ergibt sich aus (80):

$$\begin{aligned}
N_1^L(x,y) &= \frac{1}{4}(1+x)(1+y)\\
&= N_1^Q(x,y) + \frac{1}{2}N_5^Q(x,y) + \frac{1}{2}N_8^Q(x,y) + \frac{1}{4}N_9^Q(x,y) \qquad (81)
\end{aligned}$$

Das bedeutet, daß sich die bilineare Basisfunktion eines Eckpunktes aus der biquadratischen Basisfunktion des Eckpunktes und der gewichteten Summe der Basisfunktionen der angrenzenden Seitenmittelpunkte sowie des Elementmittelpunktes erzeugen läßt. Die an der Transformation beteiligten Punkte sind genau

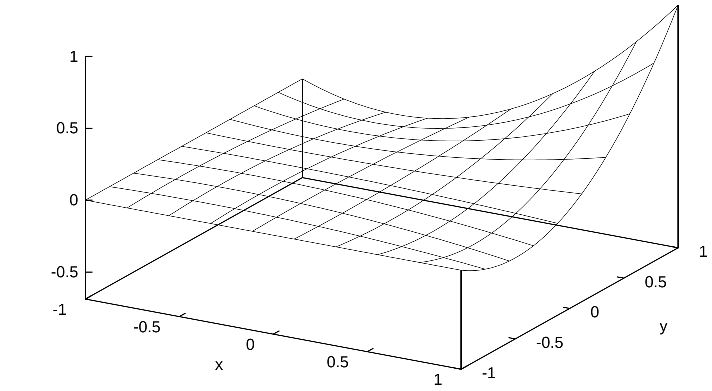


Abbildung 49: Biquadratische Basisfunktion $N_1^Q(x,y)$

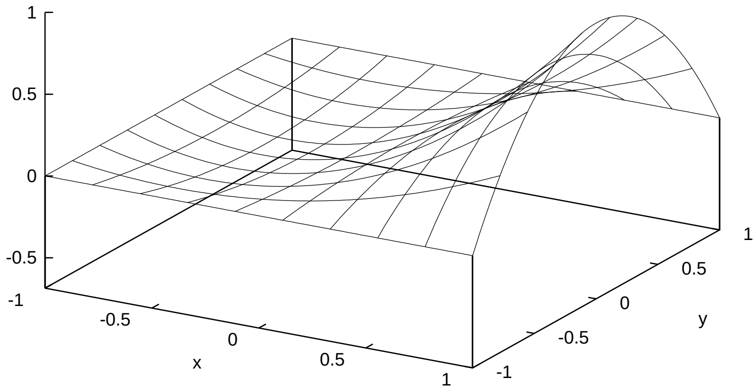


Abbildung 50: Biquadratische Basisfunktion $N_8^Q(x,y)$

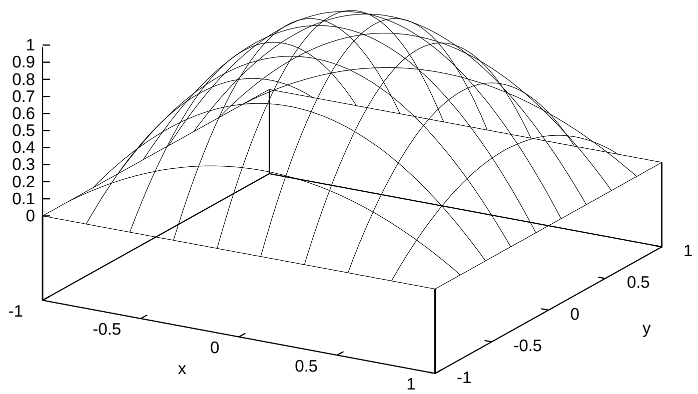


Abbildung 51: Biquadratische Basisfunktion $N_9^Q(x,y)$

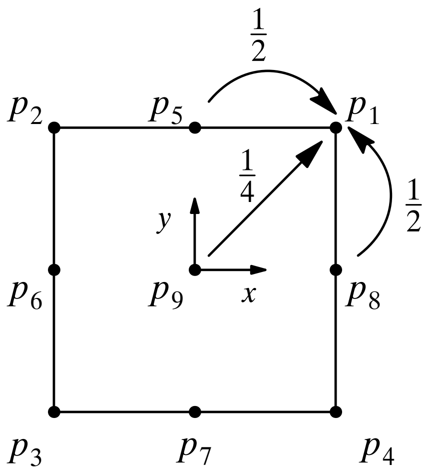


Abbildung 52: Transformation einer biquadratischen Basisfunktion auf Elementebene

die hierarchischen Nachbarpunkte entsprechend Kapitel 5.2. Auch die Gewichtsfaktoren sind identisch zu der Transformation bei bilinearen Basisfunktionen.

In Abb. 52 sind die beteiligten Punkte sowie die Gewichtsfaktoren für die Basisfunktion des Punktes p_1 eingezeichnet. Um die Basis des gesamten Elements in einen linearen und einen quadratischen Anteil zu verwandeln, ergeben sich die gleichen Transformationen wie in Abb. 39. Dieser Schritt findet auf der Blattebene des Substrukturierungsbaumes statt. Ab der nächsten Baumebene werden nach dem Assemblierungsschritt nur noch die bilinearen Basisfunktionen hierarchisiert. Dabei sind die gleichen Stützpunkte wie bei der Verwendung eines bilinearen Ansatzraumes beteiligt und die Transformation eines beliebigen Baumknotens läuft analog zu Abb. 39 ab.

Behandlung der inkompatiblen Punkte

Ein Unterschied der quadratischen Basisfunktionen zu den linearen ergibt sich bei der Behandlung der inkompatiblen Punkte.

Bei den linearen Ansatzfunktionen entspricht die Matrixtransformation für die inkompatiblen Punkte genau der hierarchischen Transformation. Dadurch ist im hierarchischen System keine zusätzliche Transformation für die inkompatiblen Stützpunkte durchzuführen. Es müssen lediglich die entsprechenden Zeilen und Spalten aus den Knotengleichungssystemen eliminiert werden (s. Kap.5.5). Dagegen ist bei den quadratischen Basisfunktionen zusätzlich zur hierarchischen Transformation eine Transformation für die inkompatiblen Stützpunkte auszuführen.

Betrachten wir den Unterschied an dem Beispiel in Abb. 53, wo zwei Elemente E_1 und E_2 entlang der Kante zwischen p_1 und p_3 zusammengesetzt werden. Die hierarchische Transformation im Element E_2 erzeugt die hierarchischen linearen

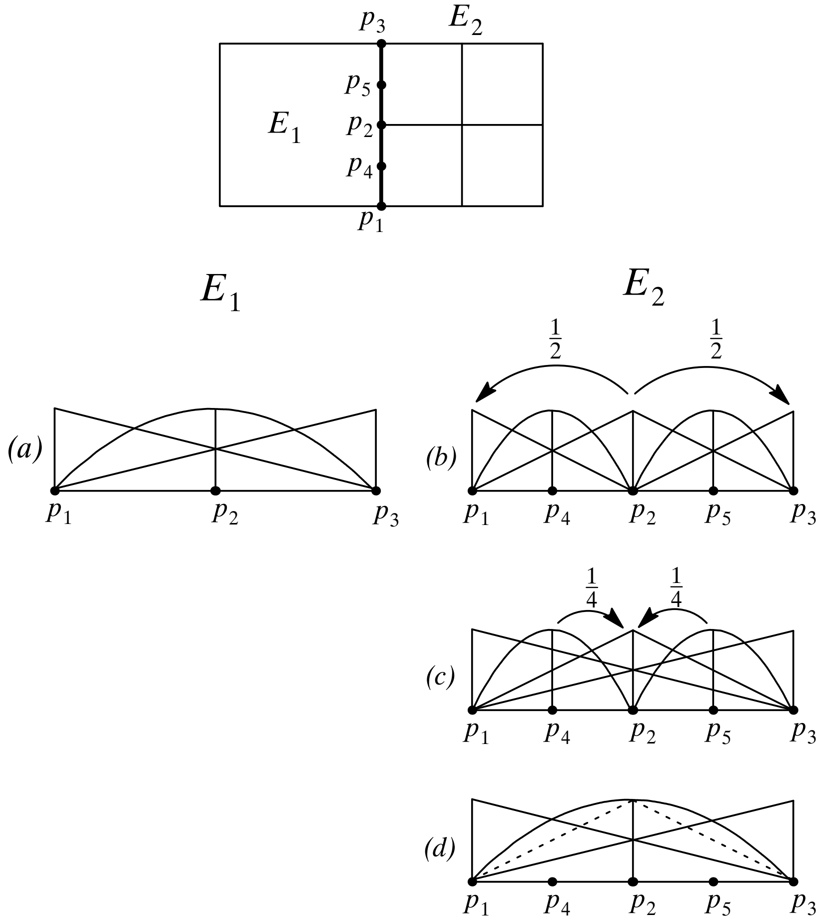


Abbildung 53: Behandlung von inkompatiblen Punkten bei quadratischen Basisfunktionen

Basen entlang der Kante (s. Abb. 53(b) und (c)). Eine Elimination der beiden inkompatiblen Punkte p_4 und p_5 (d.h. der beiden quadratischen Ansatzfunktionen) ohne vorherige zusätzliche Transformation läßt auf der Kante in Element E_2 nur einen stückweise linearen Funktionsverlauf zu. Auf der anderen Seite ist im Element E_1 ein quadratischer Funktionsverlauf möglich (s. Abb. 53(a)) was zu einer Inkonsistenz führt.

Es ist aber möglich, aus einer linearen und zwei quadratischen Basisfunktionen eine quadratische Basisfunktion mit einem größeren Träger zu erzeugen (s. Abb. 54 und [11]). Mit dieser Umformung kann die Inkonsistenz vermieden werden, indem vor der Elimination der inkompatiblen Punkte p_4 und p_5 ihre Beiträge gewichtet mit dem Faktor $\frac{1}{4}$ auf den Punkt p_2 addiert werden (s. Abb. 53(c)). So wird eine quadratische Basisfunktion erzeugt, die sich entlang der Kante erstreckt (s. Abb. 53(d)).

Die Behandlung der inkompatiblen Punkte erfolgt in drei Schritten: hierarchische Transformation, Transformation bezüglich der inkompatiblen Punkte und

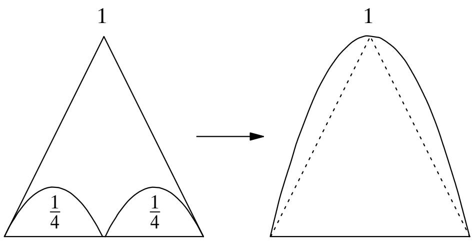


Abbildung 54: Erzeugung einer groben quadratischen Basisfunktion

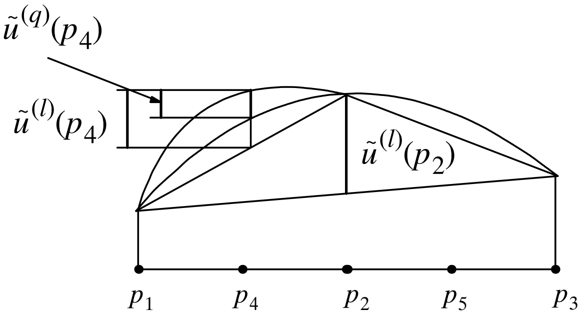


Abbildung 55: Linearer und quadratischer hierarchischer Überschuß

schließlich die Elimination der inkompatiblen Punkte aus dem Knotengleichungssystem.

Zur Berechnung des Funktionswertes eines inkompatiblen Punktes gilt wie im linearen Fall in (76), daß der hierarchische Überschuß des Punktes gleich Null ist. In [11] wird gezeigt, daß sich der *quadratische hierarchische Überschuß* $\tilde{u}^{(q)}$ eines Punktes aus dem linearen hierarchischen Überschuß des Punktes und dem linearen hierarchischen Überschuß des angrenzenden Punktes ergibt (s. Abb 55). In unserem Beispiel erhalten wir

$$\tilde{u}^{(q)}(p_4) = \tilde{u}^{(l)}(p_4) - \frac{1}{4}\tilde{u}^{(l)}(p_2). \qquad (82)$$

Unter der Bedingung, daß der quadratische hierarchische Überschuß eines inkompatiblen Punktes verschwinden muß, berechnet sich der linearen hierarchischen Überschuß aus

$$\tilde{u}^{(l)}(p_4) = \frac{1}{4}\tilde{u}^{(l)}(p_2). \qquad (83)$$

Die Knotenverschiebung des Punktes p_4 ergibt sich schließlich aus dem linearen hierarchischen Überschuß und der Interpolation aus den angrenzenden Nachbarpunkten

$$u(p_4) = \frac{1}{4}\tilde{u}^{(l)}(p_2) + \frac{1}{2}(u(p_1) + u(p_2)). \qquad (84)$$

Durch die Analogie zur hierarchischen Transformation der Knotengleichungssysteme bei bilinearen Ansatzfunktionen läßt sich unser Verfahren problemlos auf Finite Elemente mit biquadratischen Ansatzfunktionen erweitern. Als zusätzliche Berechnungsschritte sind lediglich die Transformation der Elementmatrizen und die Behandlung der inkompatiblen Punkten erforderlich. Damit wurde gezeigt, daß das iterative Lösungsverfahren auch auf Finite Elemente mit einem höheren Ansatzgrad übertragen werden kann.

6 Adaptive Relaxation

Zur Verringerung der Berechnungskosten wird der iterative Lösungsalgorithmus mit dem Verfahren der adaptiven Relaxation kombiniert. Dadurch können die Glättungsschritte auf die Teilgebiete konzentriert werden, die wesentlich zur Verbesserung der Lösung beitragen. Solch ein adaptiver Baumdurchlauf besucht deshalb nicht mehr alle Baumknoten. Zudem kann mit der Methode der adaptiven Relaxation eine adaptive Netzverfeinerung gesteuert werden.

6.1 Das Prinzip der adaptiven Relaxation

In [60] und [61] wird als Ergänzung zu elliptischen, iterativen Lösern in Kombination mit adaptiven Verfeinerungstechniken die sogenannte *fully adaptive multigrid method* (FAMe) vorgestellt. Diese Methode nutzt im Gegensatz zu den meisten Lösern sowohl die (hierarchische) Matrixstruktur als auch den aktuellen Stand der Iteration im Lösungsprozeß aus. Der Kern der FAMe bildet ein adaptives Relaxationsschema mit einer Strategie von aktiven Mengen, die anzeigen, wo die Iteration den Fehler am stärksten reduziert.

Das Verfahren kann auf eine breite Klasse von elliptischen, partiellen Differentialgleichungen zweiter Ordnung angewendet werden. Da es die Rechenarbeit auf die Punkte konzentriert, die den Fehler entscheidend reduzieren, verbessert es sowohl die Effizienz als auch die Robustheit von klassischen Multilevelmethoden (s. z.B [54]). Ein weiterer Vorteil ist, daß die Fehlerschranken, die den Abbruch der Iteration steuern, auch als Fehlerindikatoren für eine Netzverfeinerung herangezogen werden können.

Wir betrachten das dünnbesiedelte symmetrische lineare Gleichungssystem, das aus unserer Finite-Element Diskretisierung abgeleitet worden ist:

$$Ax = b \qquad (A \in \mathbf{K}^{I\times I}, b \in \mathbf{K}^I, x \in \mathbf{K}^I) \tag{85}$$

Auf diesem Gleichungssystem kann man die folgenden Definitionen einführen.

Definition 6.1

$$\theta_i = \theta_i(x) = a_{i,i}^{-1} e_i^T (b - Ax) \tag{86}$$

wird das **aktuelle skalierte Residuum** *der Gleichung i genannt, wobei e_i der i-te Einheitsvektor, d.h. die i-te Spalte der Einheitsmatrix, ist.*

Definition 6.2 *Ein* **elementarer Relaxationsschritt** *der i-ten Gleichung besteht aus einer Änderung des Vektors x durch*

$$x \longleftarrow x + \theta_i(x) e_i. \tag{87}$$

Nun haben wir die Freiheit zu entscheiden, in welcher Reihenfolge die Relaxationsschritte erfolgen. In den meisten Anwendungen werden entweder alle Komponenten gleichzeitig relaxiert (Gauß-Jacobi-Iteration) oder sequentiell in einer festen Reihenfolge (Gauß-Seidel-Iteration). In [61] wird vorgeschlagen, die Relaxation adaptiv durchzuführen.

Definition 6.3 *Das* **Residuum** *eines Vektors x ist definiert durch*

$$r = b - Ax, \tag{88}$$

und das **skalierte Residuum** *ist definiert durch*

$$\bar{r} = D^{-1}(b - Ax), \tag{89}$$

wobei $D = diag(A)$ der Diagonalanteil von A ist.

Ein elementarer Relaxationsschritt reduziert den Fehler einer Approximationslösung. Dieser Effekt wird durch das folgende Lemma beschrieben. Dabei betrachten wir den Fehler einer Näherungslösung x im Bezug zu der exakten Lösung $x^* = A^{-1}b$.

Lemma 6.1 *Sei $x - x^*$ der Fehler vor und $x' - x^*$ der Fehler nach einem elementaren Relaxationsschritt von Gleichung i, dann wird die Energienorm des Fehlers reduziert entsprechend*

$$\|x - x^*\|_E^2 - \|x' - x^*\|_E^2 = a_{i,i}\theta_i(x)^2. \tag{90}$$

Ein Beweis dieses Lemmas ist in [61] zu finden. Das Lemma besagt, daß die Fehlerreduktion nur schnell ist, wenn Gleichungen mit großen Residuen relaxiert werden.

Damit ist die beste Wahl der Relaxationsreihenfolge bei jedem Schritt die Gleichung mit dem größten Residuum zu bestimmen (Gauß-Southwell-Methode [66]). Da dies bei großen Gleichungssystemen zu teuer ist, klassifizieren wir die Residuen entsprechend ihrer Größe und führen Mengen von Gleichungen mit großen Residuen ein.

Definition 6.4 *Die* **strenge aktive Menge** *$S(\theta, x)$ ist definiert durch*

$$S(\theta, x) := \{i| \quad \|\theta_i(x)\| \geq \theta\}. \tag{91}$$

Wenn man die Relaxation auf die strenge aktive Menge beschränkt, garantiert Lemma 6.1 eine untere Schranke für die Reduktion der Energienorm des Fehlers in einem elementaren Relaxationsschritt. Leider ist die Bestimmung der strengen aktiven Menge immer noch zu teuer und wir beschränken uns deshalb auf eine Näherung von ihr.

```
proc AdaptiveRelaxation(θ, x, S̃)
      solange ( S̃ ≠ ∅ )
            begin
            wähle i ∈ S̃
            S̃ ← S̃\{i}
            falls |θ_i(x)| > θ dann
                  begin
                  x ← x + θ_i(x)e_i
                  S̃ ← S̃ ∪ Conn(i)
                  end
            end
endproc
```

Abbildung 56: **Adaptive Relaxation**

Definition 6.5 *Eine* **approximierende aktive Menge** $\tilde{S}$ *ist definiert durch*

$$\tilde{S}(\theta, x) \supset S(\theta, x). \tag{92}$$

Bei dieser Menge handelt es sich um eine erweiterte aktive Menge, die inkrementell während der Relaxation berechnet werden kann.

Die Kernidee des Algorithmus der adaptiven Relaxation ist:
Verändere keine Unbekannte mit kleinem skalierten Residuum.

Dies führt zu einem effizienten Schema die aktive Menge zu aktualisieren (s. Abb. 56). In dem Algorithmus wird der Begriff der Verbindungen (engl. connections) verwendet.

Definition 6.6 *Für jeden Index* $1 \leq i \leq n$*, definieren wir die Menge der* **Verbindungen** *durch*

$$Conn(i) = \{j | j \neq i, a_{i,j} \neq 0\}. \tag{93}$$

Die Verbindungen entsprechen den Einträgen ungleich Null in der Systemmatrix, mit Ausnahme der Diagonalelemente.

Innerhalb der Hauptschleife der Prozedur in Abb. 56 findet ein elementarer Relaxationsschritt und eine Aktualisierung der aktiven Menge statt. Dabei enthält die aktive Menge im allgemeinen auch Indizes von Gleichungen mit kleinen Residuen. Dies muß in Kauf genommen werden, wenn man mit den approximierenden aktiven Mengen arbeitet.

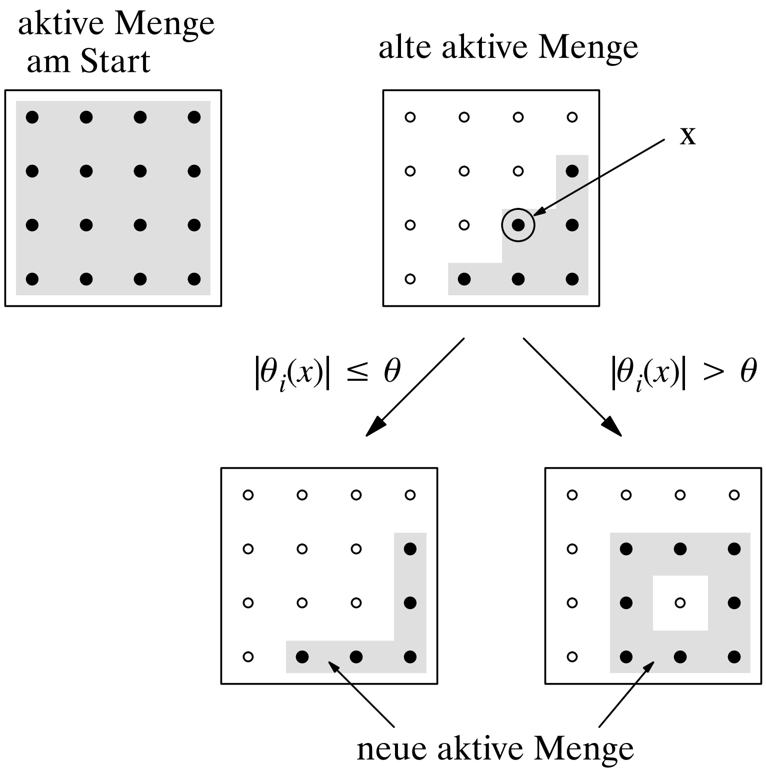


Abbildung 57: Aktualisierung der aktiven Menge bei der Relaxation

In Abb. 57 ist ein einzelner Relaxationsschritt veranschaulicht. Am Beginn der Relaxation wird die aktuelle Menge mit allen Gitterpunkten initialisiert. Nach einer Reihe von elementaren Relaxationsschritten reduziert sie sich auf die Punkte mit großen Residuen. Aus dieser Menge wird ein Punkt x ausgewählt und sein aktuelles skaliertes Residuum berechnet. Falls sein Residuum kleiner als die Toleranz ist, wird sein Wert nicht aktualisiert und der Punkt aus der aktiven Menge entfernt. Falls sein Residuum groß ist, wird sein Wert aktualisiert. Nach der Aktualisierung ist sein Residuum 0 und er wird ebenfalls aus der aktiven Menge entfernt. Da aber die Aktualisierung seines Wertes Einfluß auf seine Nachbarpunkte hat, müssen die Residuen dieser Punkte überprüft werden. Dies wird erreicht, indem die Verbindungen des Punktes in die aktiven Menge eingefügt werden.

Der Algorithmus aus Abb.56 funktioniert nur, wenn $\tilde{S}$ eine Obermenge von S ist. Die Terminierung der adaptiven Relaxation ist garantiert durch die positive Definitheit von A [61].

Das Verfahren der adaptiven Relaxation spart deutlich an Arbeitsaufwand gegenüber der normalen Gauß-Seidel Iteration (s. [61], [54]). Weitere Verbesserungen erzielt man durch eine dynamische Anpassung von θ, d.h. man startet mit großen Werten für θ und reduziert es langsam auf Werte von $O(h^2\varepsilon)$. Aber trotz dieser Einsparungen ist die Ordnung des Rechenaufwandes asymptotisch von der gleichen Ordnung wie bei der normalen Gauß-Seidel Iteration. Deshalb ist ein Einsatz der adaptiven Relaxation nur in Verbindung mit einem hierarchischen Verfahren sinnvoll [61].

6.2 Adaptive Relaxation und adaptive Netzverfeinerung

Normalerweise verwenden adaptive numerische Verfahren in der FEM eine adaptive Netzverfeinerung, aber einen festen Lösungsalgorithmus. Mit dem Verfahren der adaptiven Relaxation kann ein vollständig adaptives Verfahren implementiert werden, indem sowohl der Lösungsalgorithmus als auch die Netzgenerierung adaptiv sind.

In [61] wird gezeigt, daß die adaptive Relaxation in einem Multilevelverfahren auch zur Steuerung der Netzverfeinerung verwendet werden kann. Wenn die skalierten Residuen in allen Hierarchieebenen eines Multilevelverfahrens klein sind, dann ist der Gesamtfehler der Näherungslösung klein. Der algebraische Fehler läßt sich unter bestimmten Voraussetzungen durch die Summe der skalierten Residuen der verschiedenen Hierarchieebenen sowohl in der L_2-Norm als auch in der Energienorm beschränken (s. [61]). Als Konsequenz ergibt sich für ein hierarchisches System, daß die Relaxation die Residuen der Unbekannten auf allen Hierarchieebenen bis zur geforderten Genauigkeit reduzieren muß. Die Beschränkung des algebraischen Fehlers durch die Residuen gilt nicht bei Knotenbasen, da die Residuen mit der Kondition der Matrix multipliziert werden müssen, um Fehlerschranken zu liefern [60].

Die skalierten Residuen beschränken sowohl den algebraischen Fehler der Gleichungslösung als auch den Diskretisierungsfehler des FE-Netzes. Dabei verstehen wir unter *algebraischen Fehler* den Fehler des Gleichungslösers bezüglich der korrekten Lösung des linearen Gleichungssystems. Der *Diskretisierungsfehler* ist der Fehler der (korrekten) algebraischen Lösung bezüglich der wahren Lösung der Differentialgleichung. Die einheitliche Interpretation der Fehler im Verfahren der adaptiven Relaxation wird ermöglicht durch das Prinzip des virtuellen globalen Netzes und führt zu einem Netzverfeinerungskriterium.

Beim *virtuellen globalen Netz* ist das FE-Problem eingebettet in eine unendliche Folge von verfeinerten Netzen. Die Lösung ist auf allen Verfeinerungsebenen durch Interpolation aus den gröberen Ebenen definiert. Der Algorithmus rechnet aber nur auf den Ebenen, die tatsächlich zum Fehler der Lösung beitragen.

Im virtuellen globalen Netz gibt es *Geister-Knoten*, an denen die Genauigkeit der Lösung ausreichend ist und die durch Interpolation bestimmt werden können. Geister-Knoten müssen nicht explizit gespeichert werden. Alle anderen Knoten sind *lebende Knoten*, die gespeichert und im Relaxationsprozeß berechnet werden. Die Residuen der lebenden Knoten tragen zum algebraischen Fehler der Gleichungslösung bei, während die Geister-Knoten zum Diskretisierungsfehler beitragen.

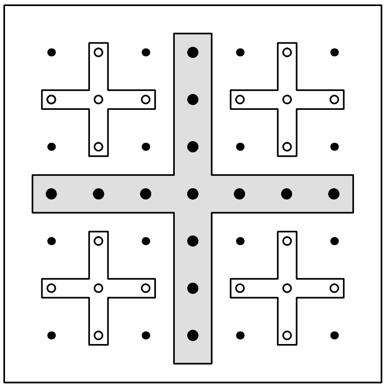


Abbildung 58: Gruppierung der Unbekannten entsprechend der rekursiven Zerlegungsstruktur

Als Adaptionskriterium ergibt sich ein adaptiver Relaxationsschritt auf der feinsten Ebene. Sobald eine Unbekannte auf der feinsten Ebene relaxiert wird, müssen die Residuen der nächst tieferen Ebene überprüft werden. Dazu wird das Netz lokal verfeinert und neue lebende Knoten erzeugt.

Dies führt zu einem Wechselspiel zwischen Iteration und Verfeinerung. Ermöglicht wird dies durch die einheitliche Interpretation der algebraischen und der Diskretisierungsfehler. Beide haben dieselbe Ursache und werden mit denselben Mittel reduziert: Die Relaxation von Unbekannten mit einem starken Residuum.

6.3 Adaptive Relaxation in der rekursiven Substrukturierung

Die Idee der Adaptiven Relaxation läßt sich auf das iterative Verfahren der rekursiven Substrukturierung übertragen. Eine Vorgehensweise mit approximierenden Mengen, wie in Kapitel 6.1, ist aber aufgrund der getrennten Datenhaltung im Substrukturierungsbaum nicht praktikabel. Als Lösung wird eine heuristische Vorgehensweise vorgeschlagen, die durch Sicherheitsmaßnahmen gewährleistet, daß keine zu großen Residuen unbemerkt bleiben.

Modifiziertes Konzept der aktiven Mengen

Während bisher jeder Freiheitsgrad einzeln betrachtet wurde, fassen wir nun mehrere Freiheitsgrade zu einer Gruppe zusammen. Dies wurde bereits in [61] für Gebietszerlegungsverfahren vorgeschlagen. Die Gruppierung der Unbekannten ergibt sich bei uns automatisch durch die rekursive Zerlegung des Berechnungsgebiets. Alle Unbekannten eines Separators bilden eine Gruppe (s. Abb. 58) und werden gemeinsam aktiv oder inaktiv gesetzt.

Nach einem Relaxationsschritt auf einem Separator werden die aktiven Mengen nicht durch alle Punkte ergänzt die eine Verbindung (93) mit den Unbekannten des Separators haben. Es werden lediglich die Separatoren der vier Sohnknoten in die aktive Menge aufgenommen. Die restlichen Punkte in Abb. 58, die durch die Verwendung von hierarchischen Basen ebenfalls eine Kopplung mit den Punkten des obersten Separators besitzen, werden nicht berücksichtigt. Dabei handelt es sich um eine Verletzung des Prinzips der approximierenden aktiven Menge, die eine Obermenge der strengen aktiven Menge sein muß. Diese Inkonsistenz muß durch Sicherheitsmaßnahmen, die später beschrieben werden, kompensiert werden.

Die aktive Menge verändert sich entsprechend der rekursiven Zerlegungsstruktur unserer Gebiete und nicht, wie in Kapitel 6.1, entsprechend der Menge der Verbindungen. Der Grund dafür ist, daß durch die dezentrale Datenhaltung keine Verbindungen zwischen den Freiheitsgraden aus verschiedenen Teilgebieten existieren. Außerdem würde das strenge Konzept der approximierenden aktiven Menge bei der Relaxation des Wurzelknotens einen kompletten Baumdurchlauf erfordern, da die hierarchischen Basisfunktionen das gesamte Gebiet überdecken.

Das Prinzip der adaptiven Relaxation im rekursiven Substrukturierungsbaum funktioniert aber trotz der Inkonsistenz aufgrund der Verwendung von hierarchischen Basen. In dem hierarchischen Ansatzraum wird die Lösungsfunktion hierarchisch dargestellt. Dabei ist der hierarchische Verschiebungswert abhängig von der Verfeinerungsstufe des Punktes und reduziert sich bei glatten Funktionen pro Verfeinerungsstufe auf 1/4 (s. Abb. 55). Diese Tatsache erlaubt es, Punkte die mehr als eine Ebene tiefer liegen (d.h. ca. 1/16 des maximalen Residuums der aktuellen Ebene liefern) zu vernachlässigen. Lokale Effekte bei denen die Gesetzmässigkeit der hierarchischen Darstellung unter Umständen verletzt ist, können z.B. durch einen vollständigen Baumdurchlauf nach Abbruch der Iteration erfaßt werden.

Mit dem modifizierten Konzept der aktiven Mengen und der Gruppierung der Unbekannten läßt sich der Algorithmus für die adaptive Relaxation im Substrukturierungsbaum formulieren (s. Abb. 59).

Die Relaxation wird im Wurzelknoten W gestartet und führt einen adaptiven Baumdurchlauf mit der Prozedur *AdapBaum* aus. Falls am Ende diese Baumdurchlaufs die Wurzel noch in der aktiven Menge $\tilde{S}$ liegt, wird ein weiterer Durchlauf gestartet.

Die Prozedur *AdapRelaxKnoten* berechnet das aktuelle skalierte Residuum für alle Unbekannten eines Baumknotens. Die skalierten Residuen entsprechen den Korrekturen des Lösungsvektors in einem Gauß-Seidel-Schritt und fallen im iterativen Lösungsalgorithmus automatisch an. Es muß nur zusätzlich überprüft werden, ob die Korrekturen tatsächlich durchzuführen sind. Falls eines dieser Residuen die zulässige Schranke θ überschreitet, wird der Relaxationsschritt auf allen Unbekannten des Baumknotens durchgeführt und der Knoten bleibt in der aktiven Menge. Falls an einem Knoten keine Unbekannte ein skaliertes Residuum über der zulässigen Schranke besitzt, wird die Lösung des Knotens nicht korrigiert und der Knoten aus der aktiven Menge entfernt.

proc AdapRelaxSchritt(θ, x, W)
  $\tilde{S} = \{W\}$
  $\hat{S} = \emptyset$
  solange ($W \in \tilde{S}$)
      AdapBaum$(\theta, x, \tilde{S}, \hat{S}, W)$
endproc

proc AdapBaum$(\theta, x, \tilde{S}, \hat{S}, K)$
  AdapRelaxKnoten$(\theta, x, \tilde{S}, \hat{S}, K)$
  falls $(K \in \tilde{S})$ dann
    für alle Söhne s_j von K
      AdapBaum$(\theta, x, \tilde{S}, \hat{S}, s_j)$
endproc

proc AdapRelaxKnoten$(\theta, x, \tilde{S}, \hat{S}, K)$
  berechne $\theta_i(x)$ für alle $i \in K$
  falls $(\max_{i \in K} |\theta_i(x)| > \theta)$ dann
    begin
    $\hat{S} \leftarrow \hat{S} \cup K$
    für alle $i \in K$
      $x \leftarrow x + \theta_i(x) e_i$
    für alle Söhne s_j von K
      $\tilde{S} \leftarrow \tilde{S} \cup s_j$
    end
  sonst
    $\tilde{S} \leftarrow \tilde{S} \backslash \{K\}$
endproc

Abbildung 59: **Adaptive Relaxation im Substrukturierungsbaum**

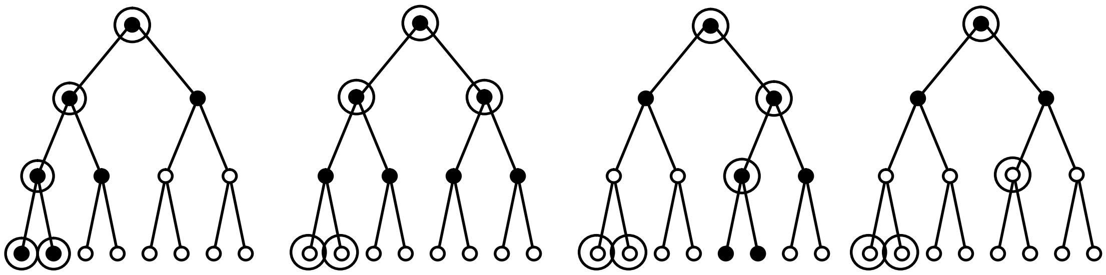


Abbildung 60: Folge von adaptiven Baumdurchläufen auf einem Substrukturierungsbaum

Die Routine *AdapBaum* sorgt nach einem durchgeführten Relaxationsschritt dafür, daß die Residuen aller Söhne des Knotens überprüft werden.

Im weiteren Text verstehen wir unter einem *adaptiven Baumdurchlauf* einen Aufruf der Routine *AdapBaum* für den Wurzelknoten. Ein *adaptiver Relaxationsschritt* ist eine Folge von adaptiven Baumdurchläufen und wird durch die Routine *AdapRelaxSchritt* ausgeführt. Schließlich bezeichnen wir als *adaptiven Relaxationszyklus* eine Folge von adaptiven Relaxationsschritten mit unterschiedlichen Abbruchschranken θ.

In dem Algorithmus treten zwei Mengen $\tilde{S}$ und $\hat{S}$ auf.

- $\tilde{S}$ = Menge aller Baumknoten, deren Werte im letzten adaptiven Baumdurchlauf geändert worden sind, sowie Knoten die in früheren Baumdurchläufen geglättet, aber im letzten Durchlauf nicht mehr kontrolliert wurden.
- $\hat{S}$ = Menge aller Baumknoten, die während eines adaptiven Relaxationsschrittes mindestens einmal relaxiert worden sind.

Die Menge $\tilde{S}$ dient zur Steuerung des Abbruchs der Iteration. Sobald $\tilde{S}$ den Wurzelknoten nicht mehr enthält, d.h alle Unbekannten des Separators des Wurzelknotens ein skaliertes Residuum unterhalb der Genauigkeitsschranke besitzen, wird der adaptive Relaxationsschritt abgebrochen. Die Menge $\hat{S}$ dient zur Sicherheitskontrolle. Nach Abbruch der Iteration werden alle Knoten, die mindestens einmal korrigiert wurden, nochmal besucht und falls notwendig relaxiert. Auch die Söhne der Knoten von $\hat{S}$ werden in diesem Baumdurchlauf überprüft, um sicherzustellen, daß die nächste Baumebene keinen Einfluß auf die Darstellung der Lösung hat.

In Abbildung 60 ist eine Folge von adaptiven Baumdurchläufen auf einem Substrukturierungsbaum basierend auf Bisektion zu sehen. Diese Teilungsstrategie wurde nur zur Veranschaulichung der verschiedenen Mengen gewählt, da sich mit ihr tiefere Bäume graphisch besser darstellen lassen. In der Abbildung sind die Baumknoten die im aktuellen Baumdurchlauf besucht werden schwarz markiert. Die aktive Menge $\tilde{S}$ wird durch die umkreisten Knoten angedeutet. In den umkreisten schwarzen Knoten wird die Relaxation tatsächlich durchgeführt. Die umkreisten weißen Knoten sind Elemente von $\tilde{S}$ die im aktuellen Baumdurchlauf nicht mehr besucht werden, aber in früheren Schritten relaxiert wurden. Die Vereinigung aller umkreister Knoten der vier Bäume ergibt die Menge $\hat{S}$.

Dynamische Anpassung von θ

Durch die Wahl von θ wird eine absolute Schranke für das zulässige skalierte Residuum festgelegt. Die Größe des Residuums ist aber abhängig von den Ausmaßen, den Struktureigenschaften und den Randbedingungen des zu berechnenden Objekts. Deshalb muß sich die Wahl eines geeigneten θ an der Norm des Verschiebungsvektors aus einer groben Diskretisierung orientieren. Bei starken Änderungen der Lösung muß θ im Laufe der Berechnung mittels dieser Norm korrigiert werden. Mit der Wahl

$$\theta = \varepsilon \cdot \|u\|_\infty \tag{94}$$

erhält man einen relativen Fehler der Größe $O(\varepsilon)$.

Die bisherigen Überlegungen gehen von einer einheitlichen Abbruchschranke θ für die gesamte Berechnung aus. Eine effizientere Methode ist aber die dynamische Anpassung von θ während der Iteration. Gestartet wird ein adaptiver Relaxationsschritt auf dem Substrukturierungsbaum mit einer oberen Schranke θ_0 für θ. Wenn der adaptive Relaxationsschritt terminiert, wird θ_0 schrittweise auf die geforderte Genauigkeit reduziert und jeweils ein neuer Relaxationsschritt gestartet. Die Wahl von θ_0 ist nicht kritisch, da für ein zu großes θ_0 keine Relaxation durchgeführt wird.

Die Vorgehensweise der schrittweise Anpassung von θ wird in [61] vorgeschlagen und liegt näher an der Methode von Gauß-Southwell, da die Arbeit zuerst auf die großen Residuen konzentriert wird.

Zusätzlich zur dynamischen Anpassung der Abbruchschranke von Relaxationsschritt zu Relaxationsschritt wird θ auch innerhalb eines Baumdurchlaufs an die jeweilige Baumebene angepaßt. Dabei gibt es zwei Grundregeln:

(1) In den groben Basisfunktionen wird genauer gerechnet wie in den feinen.

(2) Bei einer Verfeinerung des Netzes wird die Abbruchschranke reduziert.

Da die Basisfunktionen mit großen Trägern einen wesentlichen Anteil zur Lösung beitragen lohnt, es sich auf den zugehörigen Freiheitsgraden mehr Arbeit zu investieren. Fehler von Verschiebungen der groben Basisfunktionen summieren sich im hierarchischen Konzept auf. Durch eine schärfere Schranke für Baumknoten der oberen Ebenen durchläuft die adaptive Relaxation zuerst nur die oberen Baumknoten und besucht die Blattknoten erst in den letzten adaptiven Relaxationsschritten. Dadurch wird die Fehlerfortpflanzung entscheidend verringert und man erhält die gewünschte Genauigkeit mit insgesamt weniger Rechenoperationen, da die Anzahl der teuren kompletten Baumdurchläufe reduziert wird.

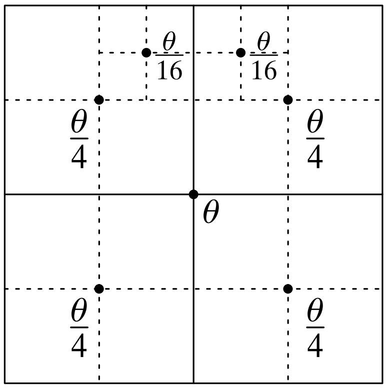


Abbildung 61: Stufenweise Anpassung der Abbruchschranke an die Diskretisierung

Eine ähnliche Idee wurde bereits in [10] für Mehrgitterverfahren vorgeschlagen. Dabei ist bei adaptiven Verfahren immer die zu erwartende Reduktion des Fehlers in Bezug zur dafür notwendigen Rechenarbeit zu setzen. Damit entsteht ein Optimierungsproblem zwischen dem Fehler und der zu leistenden Arbeit, bei dem die Arbeit proportional zum Fehler sein sollte. Es lohnt sich nur feinere Gitter zu erzeugen oder feinere Gitter zu besuchen wenn eine entsprechende Reduktion des Fehlers zu erwarten ist.

Die zweite Grundregel hat ihre Motivation darin, daß eine Verfeinerung des Netzes das Ziel hat, den Diskretisierungsfehler zu reduzieren. Da das aktuelle Netz bereits eine gewisse Genauigkeit erfüllt, muß für das verfeinerte Netz die Abbruchschranke verschärft werden. In Abb. 61 sieht man ein Netz mit einem Separator, der kein skaliertes Residuum über θ besitzt. Wenn eine Verbesserung der Genauigkeit erzielt werden soll, muß das Netz verfeinert werden und in den neu entstehenden Blättern (gestrichelt gezeichnet) eine schärfere Abbruchschranke $\theta/4$ gelten. In der nächsten Verfeinerungsstufe wird die Abbruchschranke auf $\theta/16$ reduziert.

Die Kombination der beiden Regeln führt zu einer Folge von Baumdurchläufen mit variierenden Abbruchschranken (s. Abb. 62). Nach einem adaptiven Relaxationsschritt i mit einer gegebenen Abbruchschranke auf einer Baumebene k wird die Schranke verschärft. Dabei wird im Schritt $i+1$ für die nächste Verfeinerungsstufe $k+1$ die Genauigkeit $1/4$ gefordert. Aufgrund der Regel, daß eine gröbere Basisfunktion genauer gerechnet wird (in unserem Fall mit Genauigkeit $1/4$), bedeutet dies eine Reduktion der Schranke in einer Ebene um $1/16$ zwischen zwei adaptiven Relaxationsschritten. In Abb. 62 sieht man für eine Folge von Relaxationsschritten die Abbruchschranken der verschiedenen Baumebenen.

Die Abbruchschranke eines Baumknotens der Ebene k im i-ten Relaxations-

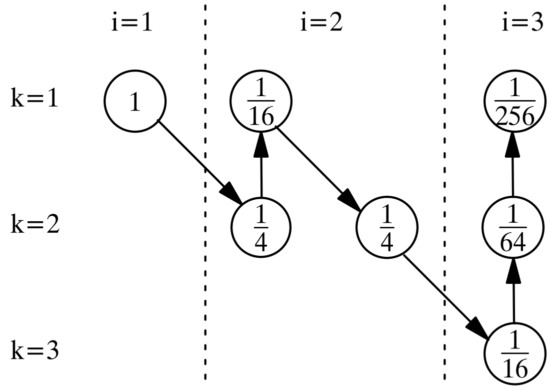


Abbildung 62: Dynamische Anpassung der Abbruchschranke an die Iterationsschritte und die Baumebenen

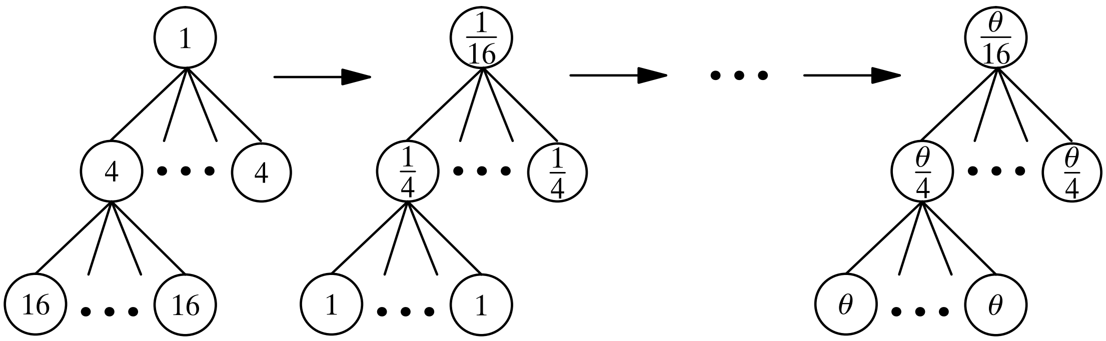


Abbildung 63: Schrittweise Reduktion der Abbruchschranke im Substrukturierungsbaum

schritt ergibt sich somit zu

$$\theta_i^k = \frac{\theta_{i-1}}{16} \cdot 4^k. \tag{95}$$

Insgesamt erhält man eine Folge von Baumdurchläufen, die mit einer sehr groben Abbruchschranke im Wurzelknoten startet (z.B. $\theta_0^0 = 1$) und die Schranken solange reduziert, bis in den Blättern des Substrukturierungsbaumes die geforderte Genauigkeit θ erreicht ist (s.Abb. 63). Da die Abbruchschranke von Baumebene zu Baumebene modifiziert wird, ist die Abbruchschranke im Wurzelknoten abhängig von der Baumtiefe. Am Ende eines adaptiven Relaxationszyklus gilt: Die skalierten Residuen in allen Blättern des Substrukturierungsbaumes liegen unter der vorgegebenen Abbruchschranke θ.

Die eben beschriebene Strategie zur dynamischen Anpassung von θ kann beliebig variiert werden. Dabei kann sowohl die Anpassung zwischen zwei Relaxationsschritten als auch die Anpassung bei einem Wechsel der Baumebenen mit anderen Faktoren durchgeführt werden. Z.B. führt die Vorgabe, grobe Basen nur doppelt so genau wie feine Basen zu rechnen zu einer Reduktion der Schranke in einer

Ebene um 1/8 zwischen zwei adaptiven Relaxationsschritten ($\theta_i^k = \theta_{i-1}/8 \cdot 2^k$). Beide Strategien führen bei unseren berechneten Objekten zu ähnlichen Diskretisierungen bei vergleichbarer Genauigkeit und Rechenaufwand.

Entscheidend ist, daß in den oberen Baumebenen genauer gerechnet und eine schrittweise Reduktion der Genauigkeitsanforderung durchgeführt wird, bis in den Blättern die Abbruchschranke θ erreicht wird.

Adaptionskriterium

Wie bereits in Kapitel 6.2 erwähnt, können die skalierten Residuen als Schranke für den Diskretisierungsfehler interpretiert werden und liefern damit ein Verfeinerungskriterium.

Sobald ein Separator eine Unbekannte mit zu großem skalierten Residuum besitzt, werden alle Unbekannten des Separators relaxiert. Dadurch ändern sich die Randbedingungen für die Teilgebiete der Söhne, die durch die Verschiebungen des Vaterknotens gegeben sind. Deshalb werden alle Söhne in die aktive Menge eingefügt und deren Residuen kontrolliert.

Falls ein Knoten keine Söhne besitzt, d.h. ein Blattknoten ist, hat ein Relaxationsschritt Auswirkungen auf die Geister-Knoten der nächst tieferen Baumebene. Diese Geister-Knoten liefern somit einen Beitrag zum Diskretisierungsfehler, der über der zulässigen Schranke liegt und müssen in lebende Knoten umgewandelt werden. Dies geschieht indem der Blattknoten verfeinert wird und vier neue Söhne erzeugt werden.

Ein vollständiger adaptiver Relaxationszyklus besteht somit aus mehreren aufeinanderfolgenden adaptiven Relaxationsschritten mit dynamischer Anpassung der Abbruchschranke. Sobald in einem Blattknoten eine Korrektur über der zulässigen Abbruchschranke durchgeführt wird, wird nach Beendigung des Relaxationsschrittes der Blattknoten verfeinert. Seine Werte werden durch Interpolation aus den gröberen Baumknoten bestimmt. Da eine Verfeinerung Auswirkungen auf den Pfad im Substrukturierungsbaum von den neuen Blättern bis zur Wurzel hat, wird der Relaxationszyklus erneut gestartet. Dieser Vorgang wird solange wiederholt bis in einem adaptiven Relaxationszyklus keine einzige Variable auf einem Blattknoten geändert werden muß.

Wenn man die adaptiven Baumdurchläufe in Abbildung 60 betrachtet, so müßten am Ende des Zyklus die beiden umkreisten Blätter des ersten Baumes verfeinert werden. Alle anderen Blätter wurden entweder gar nicht besucht oder nicht relaxiert.

Als Sicherheitsmaßnahme benötigt man im vollständig adaptiven Fall nicht die Menge $\hat{S}$. Da nur Baumknoten erzeugt werden, die zum Fehler der Lösung einen Beitrag liefern, wird nach jeder Verschärfung der Abbruchschranke im Wurzelknoten der komplette Baum durchlaufen und alle Baumknoten überprüft. Die Menge aller Baumknoten eines adaptiven Baumes entspricht der Menge $\hat{S}$ eines uniform verfeinerten Baumes.

Das Verfahren der adaptiven Relaxation im Rahmen der rekursiven Substrukturierung bewirkt eine deutliche Einsparung an Rechenoperationen im Vergleich zum Verfahren mit vollen Baumdurchläufen (s. Kap. 9). Es berücksichtigt die Tatsache, daß eine Lösung der Systemgleichungen nur bis zu einem algebraischen Fehler sinnvoll ist der in der Größenordnung des Diskretisierungsfehlers liegt. Zusätzlich ergibt sich ein lokaler Indikator zur Netzverfeinerung und damit ein Wechselspiel zwischen Gleichungslösung und Netzverfeinerung. Dieser Indikator benötigt nicht mehr die globale Sichtweise der Ergebnisse des Gesamtgebiets, wie die Indikatoren in Kapitel 2.3.1. Dadurch erfolgt auch die Fehlerschätzung parallel in den Blättern des Substrukturierungsbaumes.

Eine weitere Verbesserung des Verfahrens ist möglich, wenn man die Unbekannten innerhalb einer Gruppierung ebenfalls hierarchisch anordnet und einen Glättungsschritt auf einem Separator adaptiv durchführt. Da ein Separator Freiheitsgrade aus verschiedenen Verfeinerungsstufen enthält, könnte eine baumartige Anordnung der Freiheitsgrade die Relaxationen auf einem Separator auf die Unbekannten mit großen Basisfunktionen konzentrieren. Bisher werden alle Unbekannten eines Separators gleichwertig behandelt, da die Implementierung von ARESO die Stützpunkte eines Separators in Listen und dabei ausschließlich nach geometrischen Aspekten anordnet. Wie sich eine adaptive Glättung der Separatoren auf das Gesamtverfahren auswirkt, müßte aber noch untersucht werden.

7 Berechnungskosten

In diesem Kapitel wird ein Speicherbedarf von $O(n)$ und ein Aufwand an arithmetischen Operationen von $O(n \cdot log(n)^2)$ für das iterative Verfahren an einem Modellbeispiel nachgewiesen. Die Werte werden mit anderen Verfahren, insbesondere dem direkten Verfahren der rekursiven Substrukturierung, verglichen.

Für die Analyse der Berechnungskosten betrachten wir den Spezialfall eines quadratischen Grundgebiets mit uniformen Gitter wie in [39] und [63]. Dieser Fall ist aussagekräftig für das Verfahren, da die rekursive Zerlegung dieses Gebildes lange Separatoren auf den obersten Ebenen des Substrukturierungsbaumes erzeugt. An diesen obersten Knoten entsteht bei der rekursiven Substrukturierung der größte Bedarf an Speicher und Rechenoperationen. Sie bilden einen Flaschenhals für die sequentielle und vor allem für die parallele Version des Verfahrens der rekursiven Substrukturierung.

Einen weiteren Vorzug der Wahl des Beispiels liefert die Tatsache, daß in dieser Arbeit zweidimensionale Objekte bei Finite-Element-Berechnungen in Vierecke zerlegt werden. Bei der Zerlegung eines beliebigen Objekts in möglichst wenige Vierecke entstehen in der zweiten Ebene des Substrukturierungsbaumes Teilbäume, die dem gewählten Beispiel sehr ähnlich sind. Obwohl wir uns in allen Beispielen dieser Arbeit auf Vierecksvernetzungen beschränken, sind sämtliche Überlegungen aber auch auf Dreiecksnetze übertragbar.

Für eine adaptive Verfeinerung sind keine allgemeinen Abschätzungen machbar. Im Vergleich zu einem uniformen Gitter mit gleicher Anzahl von Unbekannten liegen i.a. weniger Stützpunkte auf den Separatoren der oberen Baumknoten. Dadurch sind der Speicherbedarf und die Berechnungskosten dieser Knoten deutlich niedriger. Da die oberen Ebenen des Substrukturierungsbaumes einen wesentlichen Anteil zu den gesamten Kosten beitragen, benötigt ein adaptives Netz bei gleicher Anzahl von Unbekannten weniger Speicher und arithmetische Operationen als ein uniformes Gitter.

Wir betrachten im folgenden das Einheitsquadrat mit $2^m \cdot 2^m$ Elementen und

$$n = (2^m + 1)^2, \quad n, m \in I\!N_0. \tag{96}$$

Gitterpunkten. Zudem nehmen wir an, daß pro Gitterpunkt ein Freiheitsgrad existiert. Eine Erweiterung auf mehrere Freiheitsgrade pro Gitterpunkt ist leicht möglich.

Außerdem ist zu beachten, daß im Substrukturierungsbaum die Randbedingungen dezentral (d.h. möglichst weit unten) in die Gleichungssysteme integriert werden. Dadurch können die Gitterpunkte auf dem Gesamtgebietsrand früher kondensiert werden, was insbesondere in den oberen Knoten die Anzahl der Unbekannten stark reduziert [63].

Zur Analyse des Speicherbedarfs und des Rechenaufwands unter Verwendung des Prinzips der rekursiven Substrukturierung benötigt man die Anzahl der Freiheitsgrade pro Baumknoten, getrennt nach inneren und äußeren Freiheitsgraden.

- n_{FG} = Anzahl der FG des Knotens,
- n_i = Anzahl der inneren FG des Knotens,
- n_a = Anzahl der äußeren FG des Knotens.

In Tabelle 2 ist ein Überblick über die Berechnungskosten eines Baumknotens für das direkte und iterative Verfahren mit rekursiver Substrukturierung gegeben. Die Kosten sind nach den einzelnen Berechnungsschritten aufgeschlüsselt. Dabei

	OP^G	SP^G	OP^Z	SP^Z
Schritt 1_{RS}	-	-	$O(1)$	$O(1)$
Schritt 2_{RS}	$\frac{n_i^3}{6} + n_a n_i^2 + \frac{n_a^2 n_i}{2}$	$\frac{(n_i+n_a)^2}{2}$	$O(n_{FG})$	$O(n_{FG})$
Schritt 3_{RS}	$n_a n_i$	$n_i + n_a$	$O(n_{FG})$	-
Schritt 2_{RSI}	$O(n_{FG} \cdot log(n_{FG}))$	$O(n_{FG} \cdot log(n_{FG}))$	$O(n_{FG})$	$O(n_{FG})$
Schritt 3_{RSI}	$O(n_{FG} \cdot log(n_{FG}) \cdot log(n)^2)$	$2 \cdot (n_i + n_a)$	$O(n_{FG})$	-
Schritt 4_{RS}	-	-	$O(1)$	$O(1)$

Tabelle 2: Berechnungskosten eines Baumknotens

bezeichnen OP^G die Anzahl der Gleitpunktoperationen und SP^G den Speicherbedarf an Gleitpunktzahlen zur Bearbeitung der Gleichungssysteme. OP^Z und SP^Z bezeichnen die zusätzlichen Bearbeitungskosten, z.B zur Erstellung der Strukturinformationen. Die Kosten zur Ermittlung der Elementmatrizen und der Fehlerschätzung sind problemabhängig und deshalb nicht in der Tabelle berücksichtigt. Solange die Kosten für diese Berechnungsschritte, wie in allen unseren durchgeführten numerischen Simulationen, von der Ordnung $O(n)$ oder $O(n \cdot log(n))$ sind, können sie vernachlässigt werden. Für genauere Details zum Aufwand von Schritt 1_{RS} und Schritt 4_{RS} wird der Leser an [63] verwiesen.

Da der Hauptaufwand asymptotisch in der Erstellung und Lösung der Gleichungssysteme liegt, werden im folgenden die Berechnungskosten für diese Berechnungsschritte genauer analysiert und obige Formeln hergeleitet. Der Aufwand von Berechnungsschritte mit geringerer Ordnung wird vernachlässigt, wie es in der Literatur über vergleichbare numerische Verfahren üblich ist.

7.1 Analyse von Speicherbedarf und Rechenoperationen für den direkten Lösungsalgorithmus

Bei dem direkten Lösungsalgorithmus mit statischer Kondensation lautet die Formel für den Speicherbedarf eines Knotens des Substrukturierungsbaumes bei symmetrischen Matrizen

$$SP_{RS}(\text{Knoten}) = \frac{(n_i + n_a)^2}{2}. \tag{97}$$

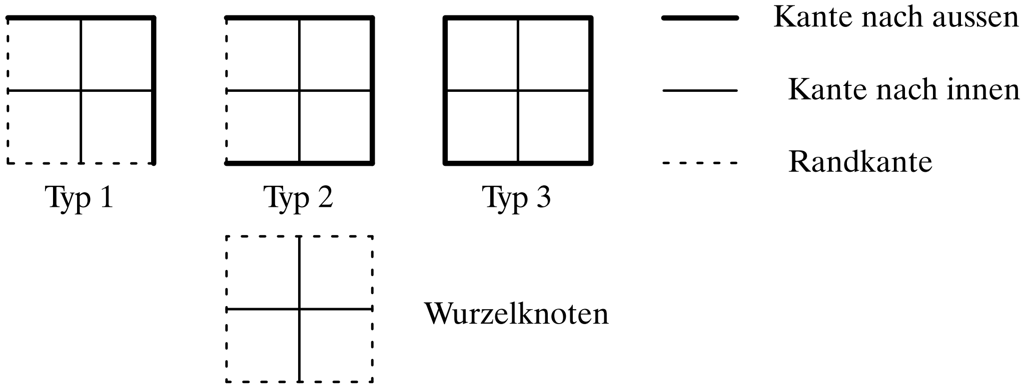


Abbildung 64: Verschiedene Typen von Baumknoten im Substrukturierungsbaum

Die Anzahl der Rechenoperationen im Schritt s_{RS}, bedingt durch eine LU-Zerlegung und zwei Matrix-Matrix Multiplikationen, ergibt sich gemäß (21), (22) und (23) zu

$$OP_{RS}(\text{Knoten}) = \frac{{n_i}^3}{6} + n_a {n_i}^2 + \frac{{n_a}^2 n_i}{2}. \tag{98}$$

Dabei gilt eine Rechenoperation als eine Addition und eine Multiplikation. Alle anderen Matrixoperationen des Teilschrittes können asymptotisch vernachlässigt werden. Die entscheidenden Größen sind also die Anzahl der Freiheitsgrade auf dem Teilgebietsrand und auf den Separatoren im Teilgebietsinneren. Diese Anzahl ist wiederum von der gewählten Teilungsstrategie abhängig.

Wir betrachten nun die Strategie der Viertelung (s. Abb. 9) etwas genauer. Der Substrukturierungsbaum zu einem uniform verfeinerten Netz von $2^m \cdot 2^m$ Elementen hat $m+1$ Ebenen. Jede Ebene s enthält 4^s Knoten. Der Separator teilt ein Element in vier Vierecke und hat die Form eines Kreuzes. Dadurch ergeben sich für die Anzahl der FG auf dem Separator für einen Knoten der Ebene s:

$$n_i = 2 \cdot 2^{-s} \cdot \sqrt{n}, \qquad 0 \leq s \leq m-1 \tag{99}$$

Somit halbiert sich n_i von Baumstufe zu Baumstufe.

Für die äußeren FG eines Knoten ist zu beachten, ob der Teilgebietsrand auf dem Gesamtgebietsrand liegt. Dabei gibt es drei Fälle:

$$n_a = \begin{cases} n_i & \text{falls zwei Teilgebietsränder auf dem Gesamtgebietsrand liegen} \\ \frac{3}{2} n_i & \text{falls ein Teilgebietsrand auf dem Gesamtgebietsrand liegt} \\ 2 n_i & \text{falls alle Teilgebietsränder im Gebietsinneren liegen} \end{cases} \tag{100}$$

Sonderfälle sind der Wurzelknoten, wo keine äußeren FG existieren und die erste Ebene, in der nur Knoten von Fall 1 auftreten (s. Abb 64). Die Werte von n_a und n_i sind obere Schranken für die tatsächlichen Werte, da die Kreuzungspunkte zwischen Separatoren und Teilgebietsränder teilweise doppelt gezählt werden. Tabelle 3 enthält die entsprechenden Werte von n_a und n_i für die obersten fünf Baumstufen nach den drei Fällen aufgeschlüsselt. Zusätzlich ist der

Eb.	Typ	Anz. Kn.	$\frac{n_i}{\sqrt{n}}$	$\frac{n_a}{\sqrt{n}}$	$\frac{OP_{RS}(Kn.)}{\sqrt{n^3}}$	$\frac{OP_{RS}(Eb.)}{\sqrt{n^3}}$	$\frac{SP_{RS}(Kn.)}{n}$	$\frac{SP_{RS}(Eb.)}{n}$
0	W	1	2	-	1.333	1.333	2.0	2.0
1	1	4	1	1	1.667	6.667	2.0	8.0
2	1	4	1/2	2/4	0.2083	5.708	0.5	12.75
	2	8	1/2	3/4	0.3490		0.781	
	3	4	1/2	4/4	0.5208		1.125	
3	1	4	1/4	2/8	0.0260	3.495	0.125	15.31
	2	24	1/4	3/8	0.0436		0.195	
	3	36	1/4	4/8	0.0651		0.281	
4	1	4	1/8	2/16	0.0033	1.913	0.0313	16.64
	2	56	1/8	3/16	0.0055		0.0488	
	3	196	1/8	4/16	0.0081		0.0703	

Tabelle 3: Speicherbedarf und Anzahl der Gleitpunktoperationen in den Knoten der oberen fünf Ebenen des Substrukturierungsbaumes

Speicherbedarf (SP_{RS}(Knoten)) und die Anzahl der arithmetischen Operationen (OP_{RS}(Knoten)) für die jeweiligen Knoten angegeben. Schließlich sind die Summe der Speicheranforderungen und der Operationen über alle Knoten einer Ebene gebildet.

Wenn man die Werte aller Baumebenen aufsummiert und berücksichtigt, daß ab einer Ebene $s \geq 5$ über 87% der Knoten im Gebietsinneren liegen und man sich daher in den unteren Baumebenen auf Typ 3 beschränken kann, erhält man die Anzahl an arithmetischen Operationen für den kompletten Substrukturierungsbaum (s. [63])

$$OP_{RS}(\text{Baum}) \leq 21.2 \cdot n^{\frac{3}{2}}, \tag{101}$$

und den gesamten Speicherbedarf des Baums

$$SP_{RS}(\text{Baum}) \leq (54.7 + 9 \cdot (ld(n) - 8)) \cdot n. \tag{102}$$

Als Ergebnis liefern die Abschätzungen (101) und (102), daß für ein Randwertproblem mit n Freiheitsgraden unter Verwendung der rekursiven Substrukturierung und der Teilungsstrategie der Viertelung $O(n^{\frac{3}{2}})$ arithmetische Operationen und $O(n \cdot ld(n))$ Speicher benötigt wird.

Für die Analyse der Berechnungskosten ist die Teilungsstrategie von entscheidender Bedeutung. Bei der Strategie der Bisektion (s. Abb. 7) erhält man als Abschätzung für die Anzahl der Gleitpunktoperationen

$$OP_{BS}(\text{Baum}) \leq 12.28 \cdot n^{\frac{3}{2}} \tag{103}$$

sowie

$$SP_{BS}(\text{Baum}) \leq (86.6 + 12.5 \cdot (ld(n) - 10)) \cdot n. \tag{104}$$

für den gesamten Speicherbedarf des Baums (s. [63]). Der Vergleich der Werte mit den Schranken für die Strategie der Viertelung zeigt, daß sich der Speicherbedarf des Baumes etwa um ein Viertel erhöht. Die Anzahl der arithmetischen Operationen verringert sich dagegen um etwa 40 %. Der Grund für den erhöhten Speicherbedarf liegt in der größeren Baumtiefe $(2m + 1)$, die mehr Datenredundanzen zur Folge hat. Die reduzierten Berechnungskosten sind durch eine stärkere Strukturierung der Freiheitsgrade und die kürzeren Separatoren bedingt.

In [63] werden auch verschiedene Kombinationen der Teilungsstrategien sowie die Auswirkung des Zeitpunkts der Elimination der Randpunkte diskutiert. Dabei wird festgestellt, daß eine Einbringung der Randbedingung erst im Wurzelknoten eine Verdreifachung der Rechenoperationen und zusätzlichen Speicherbedarf von $50n$ bewirken würde. Der Hauptnachteil ist aber, daß ca. 55% aller Gleitpunktoperationen im Wurzelknoten stattfinden, der etwa um den Faktor 20 größer ist als der maximale Knoten bei der verbesserten Behandlung der Randbedingungen.

7.2 Berechnungskosten für andere direkte Lösungsverfahren

Die Standard-FEM kann als Spezialfall der rekursiven Substrukturierung betrachtet werden. Das gesamte Berechnungsgebiet wird in einem Schritt in die Finiten Elemente zerlegt. Der entsprechende Substrukturierungsbaum hat nur zwei Ebenen. Seine Blattknoten sind Finite Elemente. Sein Wurzelknoten enthält ein sehr großes lineares Gleichungssystem, das es zu lösen gilt.

Die Matrix des Gleichungssystems ist schwach besetzt und erhält durch eine geeignete Numerierung der Knotenvariablen eine Bandstruktur. Dies ermöglicht den Einsatz sogenannter Band- und Frontlösungsmethoden (s. z.B. [64]), deren Berechnungskosten die Ordnung $O(n \cdot b^2)$ und deren Speicherbedarf die Ordnung $O(n \cdot b)$ aufweisen. Dabei ist b die Bandbreite der Matrix, die mit heuristischen Minimierungsverfahren optimiert werden kann (s. z.B. [64]).

In unserem Beispiel mit dem quadratischen Grundgebiet erhält man bei einer zeilenweisen Numerierung der Variablen die Bandbreite $\sqrt{n}$ und damit folgende Werte für die Berechnungskosten und den Speicherbedarf:

$$OP_{FL} \approx \frac{n^2}{2} \tag{105}$$

$$SP_{FL} \approx n^{\frac{3}{2}} \tag{106}$$

Im Vergleich zur rekursiven Substrukturierung sind die Kosten um eine Größenordnung höher. Während bei den Rechenoperationen die rekursive Substrukturierung schon bei kleinen Problemen günstiger ist, tritt eine Speicherersparnis erst bei größeren Problemen auf. Durch eine Parallelisierung unter Einsatz von wenigen Rechnereinheiten kann dieser Nachteil ausgeglichen werden [63].

Das Verfahren der Nested Dissection von Alan George ([24], [25]) benutzt eine andere Numerierungs-Strategie als die Frontlösungsmethode. Die Idee der Nested Dissection basiert auf einer Unterteilung des Berechnungsgebiets durch Separatoren in möglichst gleichgroße, voneinander unabhängige Teilgebiete. Anschließend werden die Freiheitsgrade entsprechend dieser Unterteilung geordnet. Durch rekursive Anwendung dieses Prinzips erhält man eine Matrix, deren fill-in bei der direkten Gleichungslösung minimal ist.

Mit der Nested Dissection konnten zum ersten Mal für die von uns betrachteten Randwertprobleme gezeigt werden, daß ein direktes Lösungsverfahren der Ordnung $O(n^{\frac{3}{2}})$ für die Gleitpunktoperationen und der Ordnung $O(n \cdot ld(n))$ für den Speicherbedarf realisierbar ist.

Zwischen der Nested Dissection und der rekursiven Substrukturierung besteht ein sehr enger Zusammenhang. Beide Verfahren benutzen die gleiche Reihenfolge zur Numerierung der Knotenvariablen und kommen dadurch zu einer identischen Verarbeitungsreihenfolge der Unbekannten. Bei unserem Modellproblem ergeben sich nach [26] und [63] folgende obere Schranken für die Kosten der Berechnung:

$$OP_{ND} \leq 10 \cdot n^{\frac{3}{2}} \tag{107}$$

$$SP_{ND} \leq 4 \cdot n \cdot ld(n) \tag{108}$$

Der Rechenaufwand ist somit bei der Nested Dissection etwas geringer als bei der rekursiven Substrukturierung (s. (101), (102)), da das Verfahren am Modellproblem entwickelt und optimiert wurde.

Bei allgemeinen Problemen mit unregelmäßigeren Geometrien der Berechnungsgebiete wird aber bei der rekursiven Substrukturierung das Berechnungsgebiet vom Ingenieur durch Separatoren in Teilgebiete zerlegt. Dieser erzeugt im Normalfall kompakte Schnittstellen, die optimal für die Gleichungslösung sind. Bei der Nested Dissection sind graphentheoretische Überlegungen zu einer geeigneten Wahl der Separatoren notwendig, die die Modellbildung nicht berücksichtigen ([25], [26]). Deshalb ist bei praktischen Ingenieurproblemen das Verfahren der rekursiven Substrukturierung überlegen.

Beim Speicherbedarf ist der Unterschied dadurch bedingt, daß zum einen gemeinsame Freiheitsgrade in verschiedenen Knoten einer Baumebene existieren und zum anderen gewisse Freiheitsgrade in mehreren Baumebenen vorkommen. Deshalb treten die meisten Freiheitsgrade mehrmals in dem Substrukturierungsbaum auf, während in der Nested Dissection jeder Freiheitsgrad nur einmal existiert. Dieser Nachteil tritt auch bei komplizierten Berechnungsgebieten und bei größeren Netzen auf. Er kann aber durch die einfache Parallelisierbarkeit des Verfahrens unter Einsatz von wenigen Rechnern kompensiert werden.

Die rekursive Substrukturierung ist in den Berechnungskosten dem Verfahren der Nested Dissection gleichwertig und besser als das Frontlösungsverfahren. Sowohl die Nested Dissection als auch das Frontlösungsverfahren haben den Nachteil, daß sie sich nur sehr schlecht parallelisieren lassen, insbesondere im Hinblick auf eine Skalierbarkeit auf mehrere Prozessoren [23]. Hier ist die rekursive Substrukturierung deutlich im Vorteil.

7.3 Berechnungskosten für das iterative Verfahren

Wie in den Kapiteln 7.1 und 7.2 gezeigt wurde, ist die rekursive Substrukturierung bei Finite-Element-Berechnungen der effizienteste direkte Lösungsalgorithmus für die Systemgleichungen.

Der entstehende Baum von Gleichungssystemen hat aber einen entscheidenden Nachteil. Die größten Knoten liegen in den obersten Ebenen und wachsen bei weiterer Netzverfeinerung stark an. Zum Beispiel befinden sich bei der Strategie der Viertelung der größte Knoten mit einem Speicherbedarf von $2 \cdot n$ in den Ebenen 0 und 1 (s. Tab. 3). In den unteren Ebenen bleibt zwar der Speicherbedarf pro Baumebene nahezu konstant, dafür reduziert sich aber der Speicher des größten Knotens pro Ebene auf ein Viertel.

Bei den Operationen ist die Konzentration auf die obersten Ebenen noch viel ausgeprägter. In unserem Beispiel ergibt bei der Strategie der Viertelung die Summe der Operationen der ersten vier Ebenen $19.1 \cdot n^{\frac{3}{2}}$ (s. Tab. 3), d.h. ca. 90 % der notwendigen Operationen. Bei der Bisektion erhält man als Summe über die ersten acht Ebenen $86.4 \cdot n^{\frac{3}{2}}$ Operationen (s. [63]), d.h. ca. 86% aller Operationen.

Im Hinblick auf eine Parallelisierung des Lösungsprozesses und die Berechnung größerer Probleme erweist sich dieser Engpaß in den oberen Baumknoten als großer Nachteil des Verfahrens.

Eine Verbesserung dieses Engpasses bringt die Umstellung des Algorithmus auf einen iterativen Löser. Wie in Kapitel 4 erwähnt wurde, bleiben durch den Verzicht auf die statische Kondensation die Matrizen dünn besiedelt und ermöglichen eine kompakte Speicherung. Diese Speicherersparnis wirkt sich in den oberen Baumknoten besonders stark aus.

Ausschlaggebend für den Speicherbedarf von dünnbesetzten Matrizen ist die Anzahl der Einträge. Eine kompakt gespeicherte Matrix muß die Matrix-Operationen des iterativen Algorithmus, wie z.B. eine Matrix-Vektor Multiplikation, in $O(E)$ ermöglichen, wobei E die *Anzahl der Einträge der Matrix* die von Null verschieden sind, repräsentiert. Nur in diesem Fall macht sich die schwache Besetzungsstruktur auch in der Anzahl der arithmetischen Operationen bemerkbar. Durch diese Forderung sind bei einer kompakten Speichertechnik für dünnbesiedelte Matrizen zusätzliche Datenstrukturen notwendig.

In der Implementierung der vorliegenden Arbeit wurde eine Datenstruktur verwendet, die einen zeilen- und spaltenweisen Zugriff auf die Matrixelemente erlaubt und einen Speicherbedarf von drei Gleitpunktzahlen pro Matrixeintrag erfordert. Dies ermöglicht einen schnellen Zugriff auf die Matrixelemente in allen Berechnungsschritten, z.B. auch bei der Assemblierung. Die arithmetischen Operationen zur iterativen Gleichungslösung erfordern lediglich Matrix-Vektor Multiplikation. Für diese Operationen können noch kompaktere Datenstrukturen verwendet werden. Solche Datenstrukturen sind entweder zeilen- oder spaltenweise strukturiert und erfordern etwa einen Speicherbedarf von zwei Gleitpunktzahlen pro Matrixeintrag (s. [7]). Da es sich bei ARESO um einen Prototypen handelt, ist diese Optimierung aber noch nicht durchgeführt worden. Im weiteren Verlauf der Arbeit werden pro Matrixeintrag von kompakt gespeicherten Matrizen drei Gleitpunktzahlen als Speicherbedarf angenommen.

7.3.1 Abschätzung des Speicherbedarfs

In diesem Kapitel wird ein Speicherbedarf der Ordnung $O(n)$ für den iterativen Lösungsalgorithmus nachgewiesen. Der Speicherbedarf ist damit eine Größenordnung günstiger als bei dem direkten Lösungsverfahren, wird aber erst bei sehr großen Systemen spürbar. Es wird sich zeigen, daß sich der Hauptaufwand von den oberen Baumknoten in die unteren Baumebenen verlagert und damit die Parallelisierung der Berechnung verbessert.

Im hierarchisierten System haben viele Stützpunkte nicht nur eine direkte Kopplung mit angrenzenden Knoten, wie es bei der Verwendung von Knotenbasen der Fall ist (s. Abb. 19). Deswegen müssen zur Analyse des Speicherbedarfs die Stützpunkte in verschiedene Hierarchieebenen eingeteilt werden, die sich in dem Ausmaß ihrer Kopplung mit anderen Gitterpunkten unterscheiden.

In Abb. 65 ist am Beispiel eines Baumknotens vom Typ 3 eine Unterteilung der Stützpunkte in die Hierarchieebenen 0 bis 3 vorgenommen worden. Die zugehörigen Basisfunktionen von Stützpunkten dieser Ebenen werden mit B_0 bis B_3 bezeichnet. Basisfunktionen vom Typ B_0 überdecken das gesamte Teilgebiet, vom Typ B_1 das halbe, vom Typ B_2 ein Viertel, usw. Die tatsächliche Überdeckung ist etwas geringer, da ein Teil der Basisfunktionen bis in angrenzende Teilgebiete reicht und erst in höheren Baumebenen zum tragen kommt (s. z.B. Basisfunktion B_2 in Abb. 65 rechts oben).

Die oberste Hierarchieebene enthält Punkte, die mit allen Punkten des Teilgebiets koppeln und volle Matrixzeilen erzeugen. Von Hierarchieebene zu Hierarchieebene reduziert sich die Kopplung der Punkte etwa auf die Hälfte, während sich die Anzahl der Gitterpunkte pro Hierarchieebene etwa verdoppelt. Im weiteren Text wird zwischen Ebenen des Substrukturierungsbaumes, sogenannten *Baumebenen*, und Ebenen in der Hierarchie der Gitterpunkte, sogenannten *Hierarchieebenen*, unterschieden.

Zur genaueren Analyse der Besetzungsstruktur der Matrizen muß man wie bei der Analyse des direkten Verfahrens die Lage der Baumknoten zum Gesamtgebietsrand (s. Abb. 64) berücksichtigen. Zusätzlich ist die Zerlegung der Matrizen in die vier Teilmatrizen K_{ii}, K_{ia}, K_{ai} und K_{aa} (s. Abb. 66) zu beachten. Der

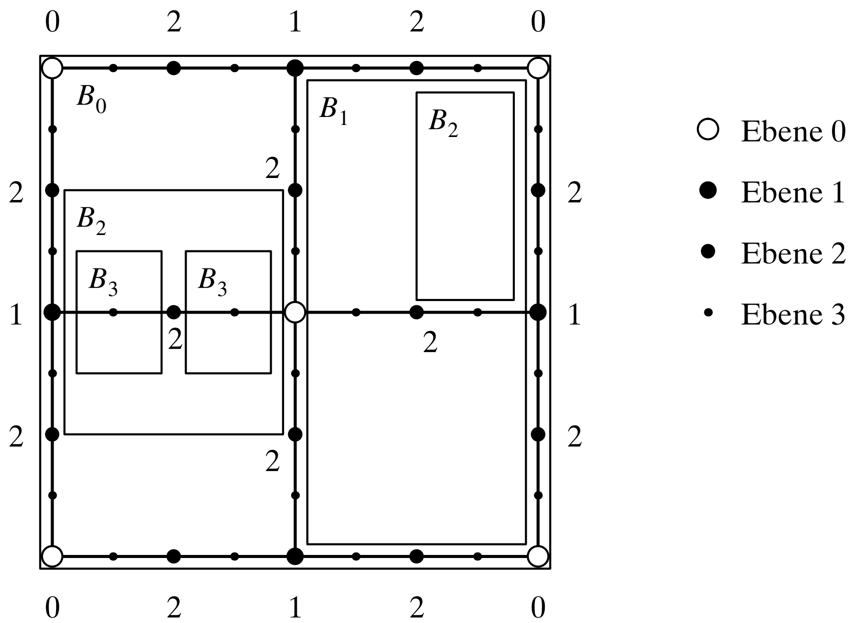


Abbildung 65: Einteilung der Stützpunkte eines Baumknotens in Hierarchieebenen

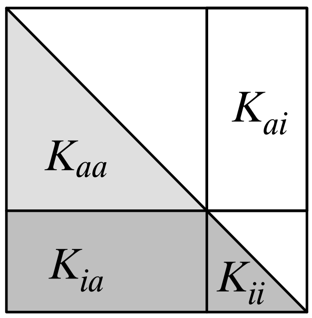


Abbildung 66: Matrixstruktur (nur unteres Dreieck gespeichert)

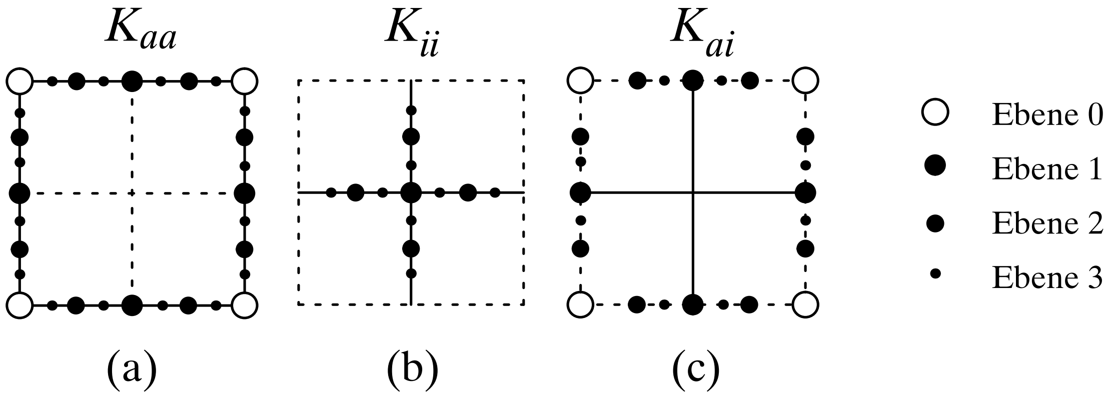


Abbildung 67: Stützpunkte eines Baumknotens vom Typ 3 getrennt nach Hierarchieebenen und Teilmatrizen

Speicherbedarf setzt sich aus dem unteren Dreieck der Matrix, die Anzahl der arithmetischen Operationen aus den Teilmatrizen K_{ii}, K_{ia} und K_{ai} zusammen (s. Kap. 7.3.2).

Wir behandeln zuerst den Fall eines Baumknotens vom Typ 3 (s. Abb. 64). Dieser Typ entspricht Teilgebieten, die im Gebietsinneren liegen und tritt im Substrukturierungsbaum am häufigsten auf.

Wir beginnen mit der Teilmatrix K_{aa}. Bei diesen Baumknoten gibt es vier Eckpunkte, die mit allen n_a Stützpunkten nach außen koppeln, und vier Seitenmittelpunkte, die mit $n_a/2$ Stützpunkten koppeln (s. Abb. 67(a)). In den weiteren Hierarchieebenen verdoppelt sich die Anzahl der Stützpunkte und halbiert sich etwa der Grad der Kopplung. In Tabelle 4 wird die Besetzungsstruktur der Matrix K_{aa} genauer aufgeschlüsselt. Es ist die Anzahl der Punkte pro Hierarchieebene,

H-Eb.	Anz. Punkte	$\frac{E(pkt)}{n_a}$	$\frac{E(H-Eb)}{n_a}$
0	4	1	4
1	4	1/2	2
2	8	3/16	1.5
3	8 8	3/32 2/32	1.25
4	8 24	3/64 2/64	1.125

Tabelle 4: Speicherbedarf für die Matrix K_{aa} eines Baumknotens vom Typ 3

die Anzahl der von Null verschiedenen Matrixeinträge pro Punkt einer Hierarchieebene $E(pkt)$ sowie die Summe der Matrixeinträge aller Punkte einer Ebene $E(\text{H-Eb})$ aufgelistet. Dabei ist die Symmetrie der Matrizen berücksichtigt, indem nur Kopplungen von Punkten mit groben Basen zu Punkten mit feinen Basen gezählt werden, nicht jedoch die umgekehrte Kopplung. Da einige Kopplungen zwischen Punkten aus gleicher Hierarchieebene doppelt gezählt werden, handelt es sich bei den folgenden Abschätzungen um obere Schranken.

Die Anzahl der Matrixeinträge von K_{aa} für ein Teilgebiet vom Typ 3 mit $(2^m + 1)^2$ Gitterpunkten ergibt sich durch die Summe der Werte von E(H-Eb) von den Hierarchieebene 0 bis m:

$$E_{HB}^{T3}(K_{aa}) \leq (6 + m) \cdot n_a. \tag{109}$$

Wir betrachten nun die Teilmatrix K_{ii}. Die Stützpunkte der verschiedenen Hierarchieebenen sind in Abb. 67(b) gekennzeichnet. In Tabelle 5 ist die Besetzungsstruktur der Matrix K_{ii}, aufgeteilt in die Hierarchieebenen, zu sehen. Um

H-Eb.	Anz. Pkt.	$\frac{E(pkt)}{n_i}$	$\frac{E(H-Eb)}{n_i}$
1	1	1	1
2	4	1/2	2
3	4 4	2/8 1/8	1.5
4	4 12	2/16 1/16	1.25
5	4 28	2/32 1/32	1.125

Tabelle 5: Speicherbedarf für die Matrix K_{ii} eines Baumknotens vom Typ 3

H-Eb.	Anz. Pkt.	$\frac{E(pkt)}{n_a}$	$\frac{E(H-Eb)}{n_a}$
0	4	1	4
1	4	3/4	3
2	8	1/8	1
3	8	1/16	0.5
4	8	1/32	0.25

Tabelle 6: Speicherbedarf für die Matrix K_{ai} eines Baumknotens vom Typ 3

die Anzahl der Matrixeinträge von K_{ii} für ein Teilgebiet vom Typ 3 mit $(2^m+1)^2$ Gitterpunkten zu berechnen, ist die Summe über die m Hierarchieebenen zu bilden:

$$E_{HB}^{T3}(K_{ii}) \leq (2 + m) \cdot n_i. \tag{110}$$

Als letzte Teilmatrizen folgen K_{ai} und K_{ia}, die aufgrund der Symmetrie die gleiche Besetzungsstruktur haben. Die beiden Matrizen enthalten die Kopplungen zwischen den äußeren und inneren Freiheitsgraden und entstehen durch Basisfunktionen auf dem Teilgebietsrand, die das Separatorkreuz im Teilgebietsinneren überdecken. In Abb. 67(c) sind die betroffenen Stützpunkte und in Tabelle 6 die Besetzungsstruktur der Matrix K_{ai} zu sehen. Die Summe über die $m + 1$ Hierarchieebenen ergibt für die die Anzahl der Matrixeinträge von K_{ai}:

$$E_{HB}^{T3}(K_{ai}) \leq 9 \cdot n_i. \tag{111}$$

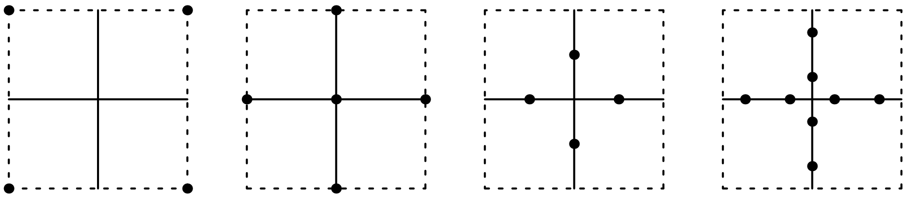


Abbildung 68: Verschiedene Hierarchieebenen der Stützpunkte des Wurzelknotens

Da die bisher betrachteten Baumknoten von Typ 3 sind, gilt

$$n_a = 2 \cdot n_i = 4 \cdot n_m, \tag{112}$$

wobei n_m die Länge eines Separators des Knotens ist und davon abhängt, wieviel Verfeinerungsstufen das uniforme Gitter des entsprechenden Teilgebiets enthält.

$$n_m = 2^m = \sqrt{n} - 1 \tag{113}$$

Damit ergibt sich die Anzahl der Matrixeinträge des gesamten Knotengleichungssystems mit (109), (110) und (111) zu

$$\begin{aligned} E_{HB}^{T3}(\text{Knoten}) &= E(K_{aa}) + E(K_{ai}) + E(K_{ii}) \\ &= (23 + 3m) \cdot n_i = (46 + 6m) \cdot n_m. \end{aligned} \tag{114}$$

Als nächstes betrachten wir den Wurzelknoten, in dem schon alle Randbedingungen aufgebracht sind und der nur noch aus dem Separatorkreuz sowie den vier Eckpunkten des Gesamtgebiets besteht (s. Abb. 68). Da der Wurzelknoten nur innere Freiheitsgrade besitzt, sind die Matrizen K_{aa}, K_{ai} und K_{ia} nicht vorhanden und es ist lediglich die Matrix K_{ii} zu speichern. Die Anzahl der Matrixeinträge von K_{ii} entspricht etwa (110). Durch die vier Eckpunkte und die vier Seitenmittelpunkte entstehen aber noch einige zusätzliche Einträge:

$$\begin{aligned} E_{HB}^{W}(K_{ii}) &= (2 + 2m + 4 \cdot 1 + 4 \cdot \frac{3}{4}) \cdot n_i \\ &= (9 + m) \cdot n_i = (18 + 2m) \cdot n_m. \end{aligned} \tag{115}$$

In der zweiten Baumebene treten vier Knoten mit zwei Teilkanten auf dem Gesamtgebietsrand auf (s. Abb. 69). Dadurch sind die Freiheitsgrade dieser beiden Kanten bereits in tieferen Baumebenen eliminiert worden.

Die genauen Werte für die Anzahl der Matrixeinträge von K_{aa} und K_{ai}, aufgeschlüsselt nach den Hierarchieebenen der Gitterpunkte, sind in den Tabelle 7 und 8 aufgelistet. Die Summe aller Hierarchieebenen ergibt die Anzahl der Matrixeinträge für einen Knoten mit zwei Teilkanten auf dem Gebietsrand.

$$E_{HB}^{T1}(K_{aa}) = (5 + m) \cdot n_a = (10 + 2m) \cdot n_m \tag{116}$$

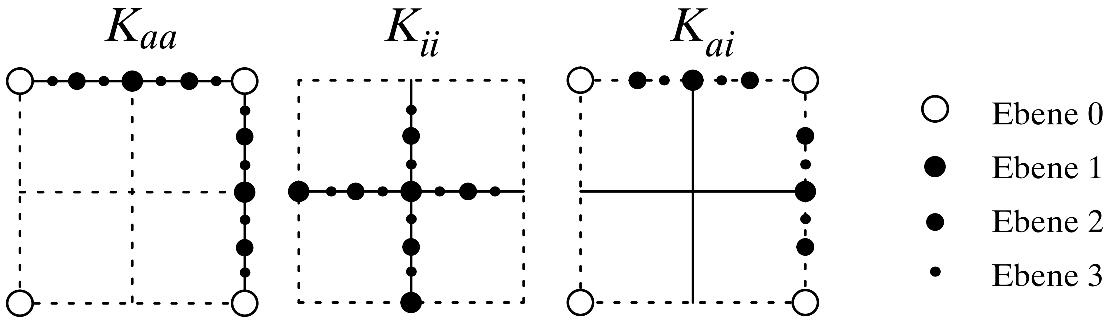


Abbildung 69: Stützpunkte eines Baumknotens vom Typ1 getrennt nach Hierarchieebenen und Teilmatrizen

H-Eb.	Anz. Pkt.	$\frac{E(pkt)}{n_a}$	$\frac{E(H-Eb)}{n_a}$
0	4	1	4
1	2	3/4	1.5
2	2	3/8	1.250
	2	2/8	
3	2	3/16	1.125
	6	2/16	

Tabelle 7: Speicherbedarf für die Matrix K_{aa} eines Baumknotens vom Typ 1

H-Eb.	Anz. Pkt.	$\frac{E(pkt)}{n_i}$	$\frac{E(H-Eb)}{n_i}$
0	4	1	4
1	2	3/4	1.5
2	4	1/8	0.5
3	4	1/16	0.25

Tabelle 8: Speicherbedarf für die Matrix K_{ai} eines Baumknotens vom Typ 1

$$E_{HB}^{T1}(K_{ai}) = 6.5 \cdot n_a = 13 \cdot n_m \tag{117}$$

Bei der Matrix K_{ii} sind im Vergleich zu Knoten vom Typ 3 noch zwei Stützpunkte auf den Seitenmittelpunkten zu berücksichtigen (s. Abb. 69(b)).

$$E_{HB}^{T1}(K_{ii}) = (2 + m + 2 \cdot \frac{3}{4}) \cdot n_i = (3.5 + m) \cdot n_i = (7 + 2m) \cdot n_m \tag{118}$$

Die Summe der drei Teilmatrizen ergibt die Anzahl der Matrixeinträge für einen Baumknoten vom Typ 1:

$$E_{HB}^{T1}(Knoten) = (30 + 4m) \cdot n_m. \tag{119}$$

Ab der dritten Baumebene treten alle drei verschiedenen Typen von Baumknoten aus Abb. 64 auf. Die Typen 1 und 3 wurden bereits analysiert. Typ 2 (eine Kanten auf dem Gesamtgebietsrand) tritt nur am Rand des Berechnungsgebiets auf und wird hier nur kurz behandelt. Die Analyse erfolgt analog zu den bisherigen Knoten und führt zu folgender Abschätzung für die Anzahl der Matrixeinträge:

$$E_{HB}^{T2}(\text{Knoten}) = (37 + 5m) \cdot n_m. \tag{120}$$

In der Tabelle 9 werden die obersten Ebenen des Substrukturierungsbaumes nach den verschiedenen Typen von Knoten aufgeschlüsselt und die Anzahl der Matrixeinträge für jeden Knoten gezeigt. Da sich die Formeln (114), (115), (119) und (120) auf Teilgebiete mit $(2^m + 1)^2$ Gitterpunkten beziehen, reduzieren sich in der Tabelle die Werte von m und n_m in jeder Baumebene. Ab der Ebene 5 werden wie bei der Analyse des direkten Lösungsverfahrens nur noch Knoten im Gebietsinneren betrachtet. In der Tabelle stehen auch bereits die Werte für die Anzahl der arithmetischen Operationen, die in Kapitel 7.3.2 hergeleitet werden.

Mit den bisherigen Ergebnissen kann der Speicherbedarf von Teilbäumen der Höhe m für alle auftretenden Knotentypen abgeschätzt und damit der Speicherbedarf des gesamten Substrukturierungsbaumes ermittelt werden.

Zuvor muß aber noch berücksichtigt werden, daß bei den kleinen Knoten in den unteren Baumebenen die Matrizen fast voll besetzt sind. Eine kompakte Speicherung für dünnbesetzte Matrizen würde hier mehr Speicher benötigen als die volle Speicherung in konventioneller Speichertechnik. Zur Veranschaulichung wird in Tabelle 10 der Vergleich zwischen voller und dünner Speichertechnik am Beispiel von Teilgebieten im Gebietsinneren gemacht. Es ist jeweils nur die Anzahl der Unbekannten und der Speicherbedarf des obersten Knotens der Teilbäume angeführt. Für den Speicherbedarf der dünnen Matrizen wurden die Werte aus einer numerischen Simulation entnommen, da die Abschätzungen aus Tabelle 9 erst für größere m gelten.

Man erkennt, daß erst ab Teilgebieten mit 256 Elementen, d.h. Teilbäume mit mindestens fünf Baumstufen, am obersten Knoten des Teilbaums eine Ersparnis

Ebene	Anz. Knoten	Typ	E_{HB}(Knoten)	O_{HB}(Knoten)
0	1	W	$(18+2m)\cdot n_m$	$(34+4m)\cdot n_m$
1	4	1	$(30+4(m-1))\cdot n_{m-1}$	$(38+4(m-1))\cdot n_{m-1}$
2	4	1	$(30+4(m-2))\cdot n_{m-2}$	$(38+4(m-2))\cdot n_{m-2}$
	8	2	$(37+5(m-2))\cdot n_{m-2}$	$(40+4(m-2))\cdot n_{m-2}$
	4	3	$(46+6(m-2))\cdot n_{m-2}$	$(42+4(m-2))\cdot n_{m-2}$
3	4	1	$(30+4(m-3))\cdot n_{m-3}$	$(38+4(m-3))\cdot n_{m-3}$
	24	2	$(37+5(m-3))\cdot n_{m-3}$	$(40+4(m-3))\cdot n_{m-3}$
	36	3	$(46+6(m-3))\cdot n_{m-3}$	$(42+4(m-3))\cdot n_{m-3}$
4	4	1	$(30+4(m-4))\cdot n_{m-4}$	$(38+4(m-4))\cdot n_{m-4}$
	56	2	$(37+5(m-4))\cdot n_{m-4}$	$(40+4(m-4))\cdot n_{m-4}$
	196	3	$(46+6(m-4))\cdot n_{m-4}$	$(42+4(m-4))\cdot n_{m-4}$
5	1024	3	$(46+6(m-5))\cdot n_{m-5}$	$(42+4(m-5))\cdot n_{m-5}$

Tabelle 9: Anzahl der Matrixeinträge und arithmetische Operationen für die Baumknoten der obersten Baumebenen

Anzahl der Elemente	m	FG	E	SP_{HB}^{W}(Knoten)	SP_{RS}(Knoten)
4	1	9	43	129	41
16	2	21	159	477	221
64	3	45	431	1293	1013
256	4	93	1039	3177	4325

Tabelle 10: Vergleich zwischen dünner und voller Speichertechnik bei uniform verfeinerten Teilgebieten im Gebietsinneren

möglich ist. In den anderen Typen von Baumknoten gelten ähnliche Werte. Das hat zur Folge, daß man die untersten vier Ebenen des Substrukturierungsbaumes voll abspeichern muß. Der Speicherbedarf einer vollbesetzten Matrix der Baumebene s beträgt

$$(6 \cdot n_s)^2/2 = 18 \cdot n_s^2, \tag{121}$$

wobei n_s die Länge des Separators des Teilgebiets ist. Die Summe über die untersten vier Ebenen ergibt einen Speicherbedarf von

$$\sum_{s=m-3}^{m} 4^s \cdot 18 \cdot 2^{2m-2s} = 4 \cdot 18n = 72 \cdot n. \tag{122}$$

Um eine Abschätzung für den Speicherbedarf des gesamten Baumes zu erhalten, werden nur Knoten im Gebietsinneren betrachtet. Diese Verallgemeinerung fällt speziell bei größeren m kaum ins Gewicht, da die oberen Baumknoten einen geringeren Anteil am Gesamtspeicherbedarf haben als beim direkten Verfahren. Wenn man die Anzahl der Matrixeinträge über alle Baumebenen aufsummiert und berücksichtigt, daß die letzten vier Stufen voll gespeichert werden, ergibt sich als Gesamtanzahl der Matrixeinträge des Substrukturierungsbaumes

$$\begin{aligned} E_{HB}(\text{Baum}) &= \sum_{s=0}^{m-4} 4^s \cdot E_{HB}^{T3}(\text{Knoten}) \quad + \quad 4 \cdot 18n \\ &= \sum_{s=0}^{m-4} 4^s \cdot (46 + 6(m-s)) \cdot 2^{m-s} \quad + \quad 4 \cdot 18n. \end{aligned} \tag{123}$$

Mit der Formel für die geometrische Reihe

$$\sum_{s=0}^{k} 2^s = 2^{k+1} - 1 \tag{124}$$

und

$$\sum_{s=0}^{k} 2^s(s+1) = 2^{k+1} \cdot k \tag{125}$$

ergibt sich

$$\sum_{s=0}^{m-4} 2^s(m-s) = 5 \cdot 2^{m-3} - m - 1. \tag{126}$$

Die Formel (125) berechnet man durch Ableitung der Summenformel für $\sum_{s=0}^{k} x^{s+1}$ nach x. Mit (126) läßt sich die Anzahl der von Null verschiedenen Matrixeinträge im Baum mit

$$E_{HB}(\text{Baum}) \leq 9.5 \cdot n + 72 \cdot n = 81.5 \cdot n \tag{127}$$

abschätzen. Für den Speicherbedarf des gesamten Substrukturierungsbaumes gilt die Abschätzung

$$\boxed{SP_{HB}(\text{Baum}) \leq 28.5 \cdot n + 72 \cdot n = 100.5 \cdot n} \tag{128}$$

unter der Voraussetzung, daß ein Matrixeintrag bei kompakter Speicherung den Speicher von drei Gleitpunktzahlen benötigt. Das bedeutet, daß ca. 70 % des gesamten Speicherbedarfs in den untersten Ebenen des Substrukturierungsbaumes anfällt. Zum Vergleich wird nochmals der entsprechende Wert für den direkten Lösungsalgorithmus angeführt:

$$SP_{RS}(\text{Baum}) \leq (54.7 + 9 \cdot (ld(n) - 8)) \cdot n.$$

Damit ist der Speicherbedarf nur noch von der Ordnung $O(n)$ statt $O(n \cdot ld(n))$ beim direkten Verfahren.

Um die Werte der Abschätzungen mit den konkreten Werten einer numerischen Simulation zu vergleichen, wurden für ausgewählte m der tatsächliche Speicherbedarf für beide Lösungsalgorithmen an unserem Testfall mit quadratischen Grundriß ermittelt. In der Tabelle 11 sind zum Vergleich die Werte in

m	FG	SP_{RS}(Baum) in MByte	SP_{HB}(Baum) in MByte
5	1089	0.47 (0.63)	0.46 (0.88)
6	4225	2.3 (3.1)	2.1 (3.4)
7	16641	11.6 (14.5)	9.2 (13.4)

Tabelle 11: Vergleich zwischen dünner und voller Speichertechnik bei einer numerischen Simulation eines Testbeispiels mit quadratischem Grundgebiet

Klammern angegeben, die sich durch die Abschätzungen (102) und (128) ergeben. Die Angaben sind in MByte und veranschlagen einen Speicherbedarf von 8 Byte für eine Gleitpunktzahl. Man erkennt, daß die tatsächlichen Werte insbesondere bei kleinen Beispielen z.T. deutlich unter den Schranken liegen.

Während beim iterativen Verfahren eine wesentliche Speicherersparnis für den gesamten Baum erst bei sehr großen Systemen spürbar ist (s. Tab 11), ist der Speicherbedarf der obersten Knoten des Substrukturierungsbaumes auch bei kleineren Problemen deutlich niedriger. Der größte Knoten des Baums liegt beim iterativen Löser im Wurzelknoten und benötigt Speicherplatz für $(18 + 2m) \cdot 2^m \cdot 3$ Gleitpunktzahlen. Beim direkten Löser sind die größten Matrizen am Wurzelknoten sowie den Knoten in Ebene 1 mit jeweils $2 \cdot n$ Gleitpunktzahlen. Eine

Betrachtung dieser Baumknoten für vier verschiedene m findet sich in Tabelle 12. Hier zeigt sich ein großer Vorteil des iterativen Verfahrens. Bei einer uniformen Verfeinerung, d.h. einer Vervierfachung der Unbekannten, verdoppelt sich der Speicheraufwand der großen Knoten lediglich, statt wie bisher mit dem Faktor 4 zu wachsen.

m	n_m	$SP_{RS}^{W/E1}$	SP_{HB}^{W}
7	128	0.27	0.10
8	256	1.1	0.21
9	512	4.2	0.44
10	1024	16.8	0.93

Tabelle 12: Speicherbedarf der größten Knoten des Substrukturierungsbaumes bei direkten und iterativen Löser in MByte

Insgesamt kann man feststellen, daß der Speicherbedarf mit $O(n)$ eine Größenordnung günstiger ist als bei dem direkten Lösungsverfahren. Durch den zusätzlichen Aufwand für komplexere Datenstrukturen und die Speicherung von vollen Matrizen in den unteren Baumebenen ist die Speicherersparnis aber erst bei sehr großen Systemen spürbar. Dagegen nimmt die Größe der maximalen Knoten des Substrukturierungsbaumes auch in den betrachteten Beispielen im Vergleich zur vollen Speicherung der Matrizen sehr stark ab und führt zu einer gleichmäßigeren Verteilung des Speicherbedarfs und des Rechenaufwand auf alle Knoten des Substrukturierungsbaumes. Damit steckt der Hauptaufwand in den unteren Baumebenen und die Skalierbarkeit einer parallelen numerischen Simulation wird entscheidend verbessert.

Im Vergleich mit anderen iterativen Lösungsverfahren muß berücksichtigt werden, daß bei der rekursiven Substrukturierung viele Informationen in den verschiedenen Teilgebieten und Baumebenen redundant gehalten werden. Dafür können viele Berechnungsschritte parallel ausgeführt werden.

7.3.2 Anzahl der arithmetischen Operationen

Wir betrachten hier nur den Fall von vollständigen Baumdurchläufen in jedem Iterationsschritt. Die Einsparung an arithmetischen Operationen die durch die Methode der adaptiven Relaxation ermöglicht wird, ist sehr stark vom berechneten Problem abhängig. Sie kann deshalb in der theoretischen Analyse nicht erfaßt werden.

Die Anzahl der arithmetischen Operationen beim iterativen Algorithmus wird durch die Gauß-Seidel Glättungsschritte sowie die Modifikationen der rechten Seite (40) und (45) dominiert. Da diese Bearbeitungsschritte im Aufwand mit einer Matrix-Vektor Multiplikation vergleichbar sind, ist die Anzahl der Operationen abhängig von der Anzahl der Matrixeinträge der beteiligten Teilmatrizen K_{ii} und K_{ai}. Man erhält für die Anzahl der arithmetischen Operationen eines Baumknotens

$$OP_{HB}(\text{Knoten}) = (E_{voll}(K_{ii}) + 2 \cdot E_{HB}(K_{ia})) \cdot n_{it}. \tag{129}$$

Dabei entspricht n_{it} der Anzahl der Iterationen.

In der Abschätzung in Kapitel 7.3.1 für Einträge der Matrix K_{ii} ist die Symmetrie der Matrix und damit nur die halbe Matrix berücksichtigt. Bei einem Gauß-Seidel-Schritt mit dieser Matrix sind jedoch Operationen mit allen Matrixeinträgen, nicht nur mit denen der unteren Hälfte, durchzuführen. Deshalb benötigen wir auch die Matrixeinträge der gesamten Matrix

$$E_{voll}(K_{ii}) = 2 \cdot E_{HB}(K_{ii}) - n_i. \tag{130}$$

Damit lassen sich die Werte für die verschiedenen Knotentypen ermitteln:

$$E_{voll}^{T3}(K_{ii}) = (6 + 4m)n_m \tag{131}$$
$$E_{voll}^{T2}(K_{ii}) = (9 + 4m)n_m \tag{132}$$
$$E_{voll}^{T1}(K_{ii}) = (12 + 4m)n_m \tag{133}$$
$$E_{voll}^{W}(K_{ii}) = (34 + 4m)n_m \tag{134}$$

Mit (129) ergeben sich die Anzahl der arithmetischen Operationen für die verschiedenen Baumknoten

$$OP_{HB}^{T3}(\text{Knoten}) = (42 + 4m) \cdot n_m \cdot n_{it} \tag{135}$$
$$OP_{HB}^{T2}(\text{Knoten}) = (40 + 4m) \cdot n_m \cdot n_{it} \tag{136}$$
$$OP_{HB}^{T1}(\text{Knoten}) = (38 + 4m) \cdot n_m \cdot n_{it} \tag{137}$$
$$OP_{HB}^{W}(\text{Knoten}) = (34 + 4m) \cdot n_m \cdot n_{it} \tag{138}$$

In der Tabelle 9 werden Anzahl der arithmetischen Operationen für die obersten Ebenen des Substrukturierungsbaumes, aufgeschlüsselt nach den verschiedenen Typen von Baumknoten, gezeigt.

Zur Abschätzung der Operationen des gesamten Substrukturierungsbaumes nehmen wir wieder an, daß alle Baumknoten vom Typ 3 sind und bei einem Iterationsschritt der komplette Substrukturierungsbaum durchlaufen wird. Außerdem werden in den letzten vier Baumebenen volle Matrizen gespeichert. Die Anzahl der Operationen auf einer vollbesetzten Matrix eines Baumknotens der Baumebene s mit einer Separatorlänge n_s ist abhängig von der Anzahl der Matrixeinträge der Teilmatrizen K_{ii} und K_{ai} und beträgt

$$4 \cdot n_s^2 + 2 \cdot 8 \cdot n_s^2 = 20 \cdot n_s^2. \tag{139}$$

Die Summe aller Knoten der untersten vier Ebenen ergibt einen Aufwand an arithmetischen Operationen von

$$\sum_{s=m-3}^{m} 4^s \cdot 20 \cdot 2^{2m-2s} = 4 \cdot 20n = 80 \cdot n. \tag{140}$$

Die Formel zur Berechnung der Anzahl der Operationen des gesamten Baumes lautet

$$OP_{HB}(\text{Baum}) = \sum_{s=0}^{m-4} 4^s \cdot (42 + 4(m-s)) \cdot 2^{m-s} \quad + 80n. \tag{141}$$

Mit den Summenformeln (124), (125) und (126) erhält man eine obere Schranke für die Anzahl der arithmetischen Operationen des iterativen Lösungsalgorithmus bei Verwendung von hierarchischen Basen:

$$\boxed{OP_{HB}(\text{Baum}) \leq 87.75 \cdot n \cdot n_{it}} \tag{142}$$

Die Aussagen über das Konvergenzverhalten des iterativen Verfahrens aus Kapitel 5.4 ergeben ein Wachstum der Iterationszahlen n_{it} von $O(log(n)^2)$. Insgesamt erfordert der iterative Algorithmus unter Verwendung von Gauß-Seidel Glättungsschritten einen Aufwand an arithmetischen Operationen von der Ordnung $O(n \cdot log(n)^2)$ und ist wesentlich günstiger als das direkten Lösungsverfahren mit $O(n^{\frac{3}{2}})$.

8 Parallele numerische Simulation

"Parallelität entsteht nicht, sie ist bereits da" [35]. Bedingt durch unser weitgehend sequentiell ausgerichtetes Denken wird bei vielen Parallelisierungsversuchen aus einem sequentiellen Ansatz heraus schrittweise Parallelität entwickelt. Eine erfolgreiche Parallelisierung sollte aber bereits beim Algorithmus und nicht erst beim Programm beginnen [35]. Das Verfahren der rekursiven Substrukturierung befolgt diesen Ansatz, in dem es die betrachteten Probleme baumartig in nebenläufige Handlungsstränge zerlegt.

8.1 Parallelität der rekursiven Substrukturierung

Ein Finite-Element-Verfahren unter Verwendung des Prinzips der rekursiven Substrukturierung gehört zur Klasse der *Divide-and-conquer-Verfahren*, auch *Teile-und-herrsche-Verfahren* genannt. Dieses Verfahren läßt sich in die beiden folgenden Schritte gliedern ([19]):
Divide-Schritt: Das Problem wird in zwei oder mehrere Teilprobleme derselben Art wie das Originalproblem aufgespalten, die unabhängig voneinander gelöst werden können.
Conquer-Schritt: Die Lösungen der Teilprobleme werden zu einer Lösung des Gesamtproblems zusammengefügt.

Ein Divide-and-conquer-Verfahren zerlegt ein Problem also in mehrere, voneinander unabhängige Teilprobleme und bietet somit die Möglichkeit der Parallelisierung der Berechnung. Dadurch ist die rekursive Substrukturierung ein Lösungsverfahren, mit dem die Methode der Finiten Elemente sehr einfach verteilt auf mehreren Rechnereinheiten durchgeführt werden kann.

In unserem Fall stellt jeder Knoten des Substrukturierungsbaumes ein Teilproblem dar. Jeder Knoten wird solange in Teilprobleme zerlegt, d.h. in Sohnknoten, bis die Blätter des Substrukturierungsbaumes erreicht werden. Die Blätter rufen dann Standard-FE-Routinen zur Lösung des Problems auf.

Betrachten wir nun den Ablauf bei der Ausführung eines Berechnungsschrittes auf einem Baumknoten.

(1) Der Knoten empfängt die für den Berechnungsschritt erforderlichen Berechnungsdaten.

(2) Ausführung des Berechnungsschrittes.

(3) Der Knoten versendet die Ergebnisse des Berechnungsschrittes.

Die Vorgehensweise beim Senden und Empfangen unterscheidet sich bei top-down- (Divide-Schritt) und bei bottom-up-Baumdurchläufen (Conquer-Schritt) (s. Abb. 70). Bei einem Berechnungsschritt eines top-down-Durchlaufes (d.h.

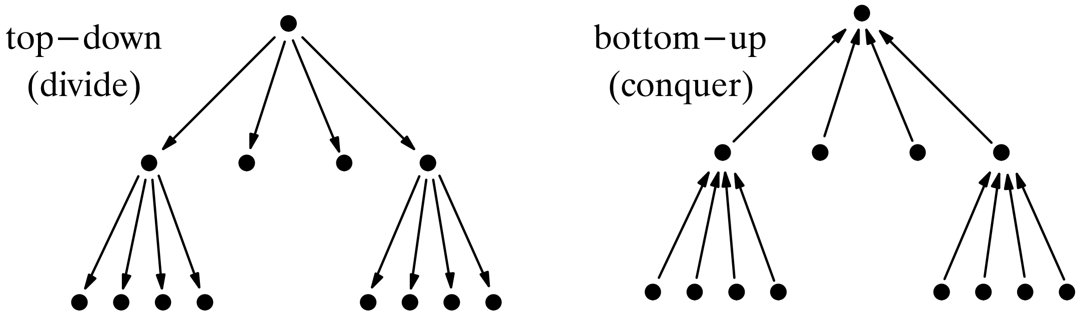


Abbildung 70: Datenfluß bei top-down- und bottom-up-Durchlauf

Schritt 1_{RS}, 3_{RS} oder 3_{RSI}(Phase 1)) empfängt der Knoten seine Daten von seinem Vaterknoten. Nach Beendigung des Berechnungsschrittes sendet er die Berechnungsdaten an all seine Söhne, die den Berechnungsschritt ebenfalls durchzuführen haben. Bei einem bottom-up-Berechnungsschritt (Schritt 2_{RS}, 2_{RSI}, 3_{RSI}(Phase 2) oder 4_{RS}) benötigt ein Baumknoten die Daten all seiner Söhne. Die Ergebnisse des Berechnungsschrittes werden an den Vaterknoten weitergegeben.

Bei einer numerischen Simulation ist es nicht erforderlich und aus Effizienzgründen auch nicht sinnvoll, alle Ergebnisse eines Berechnungsschrittes an einen anderen Baumknoten zu schicken. Es wird unterschieden zwischen *lokalen Daten* eines Baumknotens und *Datenströmen zwischen Baumknoten*. Die Datenströme, die von einem Baumknoten verschickt werden, enthalten je nach Berechnungsschritt unterschiedliche Daten (s. Tabelle 13). Der Wert $O(n_{FG}^2)$ für die Verfei-

Berechnungs-schritt	Datenstrom	Länge des Datenstroms
Schritt 1_{RS}	Zerlegungsinformation oder Verfeinerungsliste	$O(1)$ bzw. $O(n_{FG}^2)$
Schritt 2_{RS} Schritt 2_{RSI}	Matrix K_{aa} und Strukturinformation über Stützpunkte auf Teilgebietsrand	$O(n_a^2)$ bzw. $O(n_a log(n_a))$
Schritt 3_{RS} Schritt 3_{RSI}(P1)	Lösungsvektor für Teilgebietsrand von Sohnknoten	$O(n_i)$
Schritt 3_{RSI}(P2)	Vektor der rechten Seite f_a für Teilgebietsrand	$O(n_a)$
Schritt 4_{RS}	Ergebnisse der Berechnung	$O(1)$ bzw. $O(n_{FG})$

Tabelle 13: Datenströme einer parallelen Berechnung

nerungsliste betrifft den ungünstigsten Fall, daß alle Elemente im Teilgebiet des Knoten verfeinert werden müssen. Bei einer adaptiven Verfeinerung ist die Liste aber wesentlich kürzer und i.a. nicht in alle Teilgebiete zu verschicken. Im

Falle einer uniformen Verfeinerung ist überhaupt keine Verfeinerungsliste, sondern lediglich ein Startsignal für den nächsten Adaptionszyklus zu versenden. Im Schritt 2 ist die Länge des Datenpakets abhängig von der Besetzungsstruktur und der Speichertechnik der Matrix. Der Speicherbedarf der Matrix ist beim direkten Algorithmus $O(n_a^2)$ und beim iterativen Algorithmus $O(n_a log(n_a))$. Der Kommunikationsaufwand von Schritt 4_{RS} ist abhängig von der Implementierung des Verfahrens. Da die Ergebnisse einer FE-Berechnung (Verschiebungen, abgeleitete Größen) nur an den Blättern des Substrukturierungsbaumes entstehen, gibt es folgende drei Varianten:

V1: Die Blätter senden die Daten direkt an den Rechner, der die Fehlerschätzung oder das Postprocessing durchführt. In diesem Fall entsteht ein Datenpaket von $O(n_{FG})$ an einem Blattknoten und von $O(1)$ an allen anderen Knoten.

V2: Die Ergebnisse werden an den Vaterknoten versendet. Jeder Knoten konkateniert die Berechnungsergebnisse seiner Söhne und verschickt sie an seinen Vater. Hier ist die Länge der zu verschickenden Daten abhängig von der Anzahl der Finiten Elemente des zu dem Knoten gehörigen Teilgebiets. Die Anzahl der Elemente entspricht dem Quadrat der Länge eines Separators und ist von der Ordnung $O(n_i^2)$.

V3: Als Kombination der beiden bisherigen Varianten können die Ergebnisse bis zu einer bestimmten Baumebene gesammelt werden. Ab diesen Knoten werden sie entweder dezentral verarbeitet oder an einen Rechner zur Auswertung verschickt.

Neben den in Tabelle 13 beschriebenen Datenströmen zwischen den Baumknoten gibt es lokale Daten, die in den Baumknoten gehalten werden. Dadurch wird die Kommunikation zwischen den Baumknoten reduziert.

In Abb. 71 sieht man den *Datenflußgraphen* eines parallelen V-Zyklus. Ein V-Zyklus ist ein Divide-Schritt gefolgt von einem Conquer-Schritt. Datenströme zwischen den Baumknoten sind in der Abbildung mit durchgezogenen Linien gekennzeichnet, während die lokalen Daten durch gestrichelte Linien angedeutet sind. Man erkennt gut, wie sich die Datenströme im Divide-Schritt in jedem Baumknoten zerteilen, um im Conquer-Schritt wieder zusammengefügt zu werden. Das Ergebnis des V-Zyklus, ein einziger Datenstrom, dient als Eingabe für den nächsten V-Zyklus.

Der in Kapitel 3.2.5 beschriebene W-Zyklus entspricht zwei aufeinander folgenden V-Zyklen und ist in Abb. 72 zu sehen. Auch hier sind die lokalen Daten gestrichelt angedeutet und müssen nicht durch den Substrukturierungsbaum transportiert werden. Der Ergebnisdatenstrom des W-Zyklus enthält die Berechnungsergebnisse für ein FE-System und dient als Eingabestrom für den nächsten adaptiven Berechnungszyklus, der wieder ein W-Zyklus ist und eine Berechnung auf einem feineren Netz durchführt.

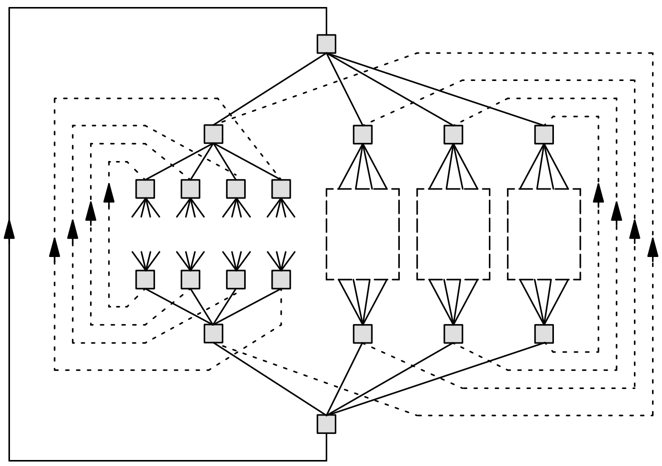


Abbildung 71: Datenfluß eines parallelen V-Zyklus

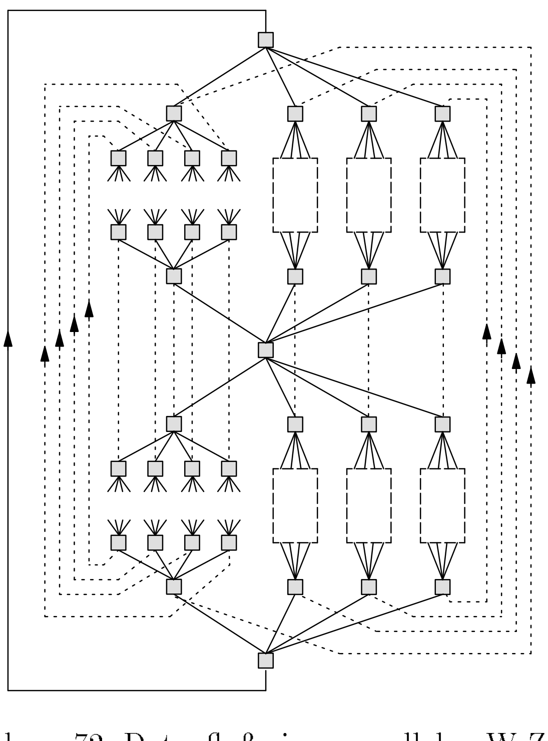


Abbildung 72: Datenfluß eines parallelen W-Zyklus

Der iterative Lösungsalgorithmus wird von einer Folge mehrerer V-Zyklen gebildet. Diese Folge wird terminiert, wenn die Lösung der Gleichungssysteme genau genug ist. Anschließend wird der nächste Berechnungszyklus mit einem feineren Netz gestartet.

Die Darstellung mittels Datenflußgraphen entspricht der Formulierung in einer funktionalen Programmiersprache und zeigt sehr deutlich die potentielle Parallelität eines Algorithmus. Das Verfahren der rekursiven Substrukturierung wird in [20] am Beispiel von ARESO in einer funktionalen Programmiersprache beschrieben.

Durch die baumartige Zerlegung des Gesamtproblems in Teilprobleme mit klar strukturierter Interaktion bietet sich eine Implementierung auf einem Parallelrechner mit *MIMD Architektur* und *message-passing* an. Bei MIMD (Multiple Instruction streams Multiple Data streams) Architekturen besitzt jeder Prozessor sein eigenes Programm, in einem Multirechnersystem (message passing system, loosely coupled system) verfügt jeder Prozessor über einen eigenen Speicher [35]. Im folgenden Abschnitt wird die Realisierung einer verteilten numerischen Simulation auf dieser Architektur beschrieben.

8.2 Verteilte Ausführung einer FE-Simulation mit ARESO

In diesem Abschnitt wird beschrieben, wie eine verteilte numerische Simulation basierend auf dem Prinzip der rekursiven Substrukturierung mit dem Programm ARESO (s. Kap. 3.2.6) ausgeführt wird.

In ARESO wird zwischen den folgenden Prozessen zur Bearbeitung der Daten eines Baumknotens K unterschieden.

$I(K)$: Ingenieur, der das Problem stellt und die Ergebnisse auswertet.

$A(K)$: ARESO-Prozeß, der die Berechnungsschritte auf den Knoten des Substrukturierungsbaumes ausführt.

$FEM(K)$: Finite-Element-Programm, das die Elementgleichungssysteme und die Elementinformationen erstellt.

$FS(K)$: Fehlerschätzer.

Von diesen Prozessen können mehrere gleichzeitig laufen. Insbesondere werden ein ARESO-Prozeß und das Finite-Element-Programm auf allen eingesetzten Rechnern gestartet. Der Prozeß $FS(K)$ ist i.a. nur am Rechner des Wurzelknotens notwendig, ebenso der oder die Ingenieure zur Bearbeitung der Ein- und Ausgabe. Die Prozesse $FEM(K)$ und $FS(K)$ werden auch Hilfsprozesse oder Hilfsdienste genannt.

Zwischen den vier verschiedenen Typen von Prozessen sind *Schnittstellen* notwendig, über die Daten kommuniziert werden (s. Abb. 73). Diese Daten werden

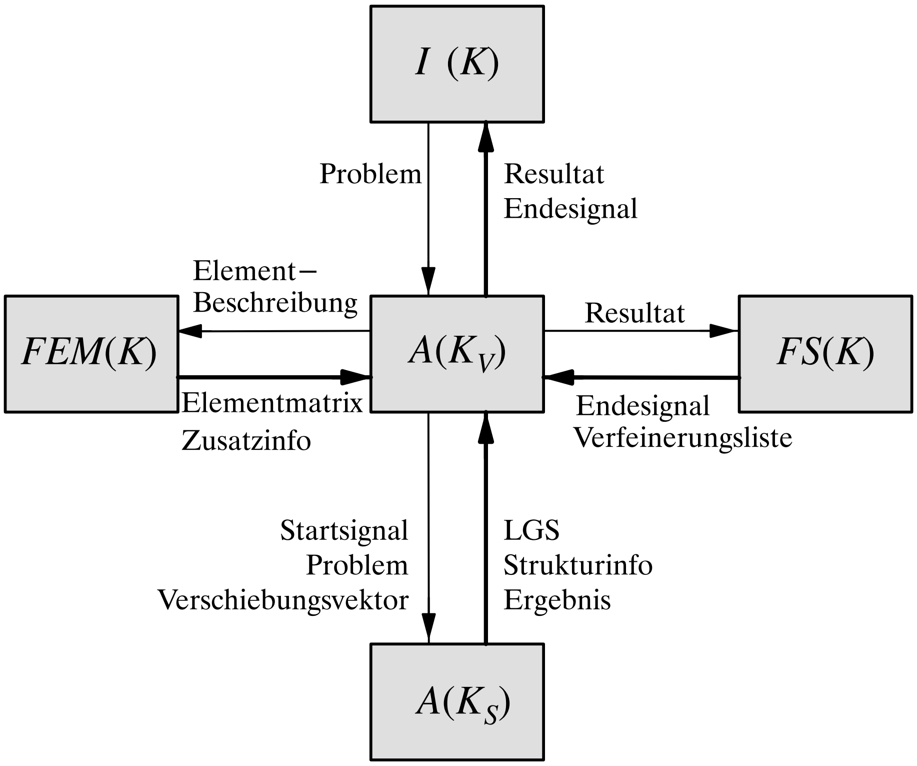


Abbildung 73: Schnittstellen zwischen den verschiedenen Prozessen

unterschieden in *Aufträge* (dünne Pfeile) und *Ergebnisse* (fette Pfeile). Die Berechnung wird durch den Ingenieur gestartet, der das Randwertproblem für den Wurzelknoten formuliert und an einen ARESO-Prozeß übergibt. Die numerische Simulation wird beendet, wenn der Fehlerschätzer erkennt, daß die geforderte Genauigkeit der Berechnung erreicht ist und ein Endesignal als Ergebnis liefert. In Abbildung 73 sind zwei ARESO-Prozesse eingezeichnet, um die Schnittstelle zwischen einem Vaterknoten (K_V) und einem Sohnknoten (K_S) zu zeigen.

Abbildung der Prozesse auf ein Rechnernetz

Ein entscheidender Faktor für die Abbildung der Prozesse auf ein Rechnernetz ist die Topologie des Verbindungsnetzwerks (s. [2], [35]). Dabei sollte die Verbindungsstruktur des Rechners die Kommunikationsstruktur des Programms widerspiegeln. In unserem Divide-and-conquer-Algorithmus eignen sich besonders hierarchisch strukturierte baumartige Verbindungsnetzwerke ([35], [53]). Wie sich in den numerischen Beispielen zeigen wird, wirkt sich der Einsatz solcher Netzwerke positiv auf die parallelen numerischen Simulationen aus.

Nun stellt sich die Frage, wie man die Knoten des Substrukturierungsbaumes auf die verschiedenen Prozesse abbildet. Prinzipiell kann man in ARESO jeden Baumknoten als eigenständigen Prozeß betrachten. Das ermöglicht es, bei ausreichender Rechneranzahl jeden Baumknoten auf einen eigenen Rechner abzubilden, auf dem die Prozesse $A(K)$, $FEM(K)$ und $FS(K)$ laufen. In der Praxis übersteigt die Anzahl der Baumknoten die Anzahl der Rechnereinheiten deutlich. Deshalb müssen mehrere Baumknoten jeweils einem ARESO-Prozeß zugewiesen werden. Dieser ARESO-Prozeß verwaltet die Daten der Baumknoten und führt die einzelnen Berechnungsschritte auf den Baumknoten aus. Zudem startet der ARESO-Prozeß bei Bedarf die Hilfsprozesse.

Die Zuordnung der Baumknoten zu den einzelnen Prozessoren erfolgt *statisch* bei der Problembeschreibung durch den Ingenieur mittels einer Konfigurationstabelle. Eine *dynamische* Verteilung der Baumknoten während eines Berechnungslaufes ist durch Migration der Daten der entsprechenden Knoten möglich. Solch ein dynamischer Lastausgleich ist zwar technisch realisiert, wird aber in den praktischen Berechnungen bisher nicht eingesetzt. Der Grund liegt darin, daß noch geeignete Strategien zur Erfassung der Last und zur Umverteilung der Daten gefunden werden müssen. Diese Vorgehensweise ist insbesondere bei adaptiven Berechnungen sinnvoll, war aber nicht Aufgabe der vorliegenden Arbeit. In [15] und [5] wurde bereits eine dynamische Lastverteilung an einer Simulation des ARESO-Programms realisiert. Dabei wird anhand eines zentralen Servers ([15]) bzw. mittels eines Maklersystems ([5]) versucht, Lastungleichheiten zu entdecken und durch Datenumverteilung zu beseitigen. Bei beiden Arbeiten zeigt sich, daß eine dynamische Zuordnung der Baumknoten eines adaptiven Substrukturierungsbaumes auf ein Rechnernetz eine gleichmäßigere Auslastung der einzelnen Rechner bewirkt und die Gesamtrechenzeit verkürzt wird. Strategien für das Lastverteilungsproblem bei parallelen Multilevelverfahren werden z.B. auch in [8] behandelt.

Realisierung der Kommunikation

Ein ARESO-Prozeß läuft also auf einer ihm zugewiesenen Rechnereinheit ab und verwaltet einen oder mehrere Baumknoten. Für alle ihm zugeteilten Knoten erzeugt er im Laufe der Berechnung die Strukturinformationen und die Knotengleichungssysteme. Zudem verwaltet er die lokalen Daten der Knoten sowie deren Aufträge und Ergebnisse in einer sogenannten *internen Datentabelle* (IDT). Der ARESO-Prozeß ist über eine Menge von Kommunikationskanälen ansprechbar, über die er Datenpakete sendet und empfängt.

Die Kommunikation in ARESO erfolgt über Datenpakete. Dabei wird zwischen *externer Datenkommunikation* zwischen verschiedenen Rechnereinheiten und *interner Datenkommunikation* zwischen Baumknoten auf derselben Rechnereinheit unterschieden. Interne Datenkommunikation wird nicht explizit durchgeführt. Die entsprechenden Daten bleiben lokal bei den Baumknoten und sind von allen Baumknoten der Rechnereinheit zugreifbar. Alle Aufträge und Ergebnisse die Baumknoten der selben Rechnereinheit betreffen werden in der internen Datentabelle verwaltet.

Externe Datenkommunikation erfolgt über das Rechnernetz und wird in einem Datenpuffer, der *externen Datentabelle* (EDT), abgelegt. Die Aufträge und Ergebnisse der externen Datentabelle werden vom ARESO-Prozeß in die interne Datentabelle eingetragen. Ein ARESO-Prozeß kann sowohl aus der internen als auch externen Datentabelle auswählen, welcher Auftrag als nächstes bearbeitet wird. Die Kommunikation in ARESO ist *asynchron*, da die Kommunikationskanäle Nachrichten speichern können und ein Prozeß eine Nachricht senden kann, ohne zu warten oder zu blockieren, bis die Nachricht empfangen ist [2]. Als Konsequenz können bereits Berechnungsschritte auf bestimmten Baumknoten durchgeführt werden, während andere noch auf die Ergebnisse ihrer Aufträge warten.

In der internen Datentabelle erhalten die Baumknoten einen Status (*rechenbereit/wartend*). Nach dem Eintreffen aller Ergebnisse eines Auftrags bekommt ein Baumknoten den Status rechenbereit und kann einen Berechnungsschritt durchführen. Anschließend löscht er alle Datenpakete des Auftrags und den Auftrag selbst. Je nach Berechnungsschritt werden in einem top-down-Schritt Aufträge an die Söhne oder in einem bottom-up-Schritt Ergebnisse an den Vaterknoten verschickt.

Die Datenpakete, die über das Verbindungsnetzwerk gesendet werden, enthalten als Information den Empfänger, den Sender, den Auftragstyp und die Daten. Die Verwaltung der Adressen der Sender und Empfänger sowie der Kommunikationskanäle erfolgt in jedem ARESO-Prozeß für die ihm zugeteilten Baumknoten. Sie wird in der Konfigurationsbeschreibung des zu lösenden Randwertproblems spezifiziert und bei einer dynamischen Lastverteilung aktualisiert.

In der aktuellen Implementierung erfolgt die Kommunikation auf Basis des UNIX-Betriebssystems und ist dadurch sowohl auf sehr vielen Rechnerplattformen einsetzbar als auch wenig störanfällig. Datenpakete werden über den entsprechenden UNIX-Befehl "remote copy" (rcp) über das Netz verschickt und in einem speziellen Dateiverzeichnis gepuffert. Als eine etwas schnellere Alternative bietet sich eine rechnerübergreifende Kommunikation direkt von Hauptspeicher zu Hauptspeicher an. Erste Versuche wurden mit [47] bereits unternommen. Eine Umstellung auf vorhandene Softwarepakete, wie z.B. PVM oder P4, ist jederzeit möglich.

Abschließend zeigt der folgende Algorithmus die Ausführungsschleife, die ein ARESO-Prozeß nach seiner Erzeugung bis zu seiner Terminierung wiederholt:

```
  ARESO-Prozeß:

begin
lies Bauteilbeschreibungen ein
lies Konfigurationsdatei ein
wiederhole
     begin
     Übernahme von Datenpaketen von der EDT in die IDT
     falls es einen rechenbereiten Auftrag in IDT gibt
          begin
               wähle einen rechenbereiten Auftrag
               führe den Auftrag aus
               falls top-down-Durchlauf
                    sende Auftrag an alle Söhne
               sonst
                    sende Ergebnis an Vater
          end
     sonst
          werde einstellbaren Zeitraum inaktiv
     end
bis (Programmabbruch-Signal liegt vor)
end
```

8.3 Analyse der verteilten Simulation

Die Analyse einer verteilten Simulation ist abhängig von dem zu berechnenden Problem, der Rechnernetztopologie, der Teilungsstrategie im Substrukturierungsbaum und der Verteilungsstrategie für den Substrukturierungsbaum auf das Rechnernetz.

Wir betrachten wieder ein quadratisches Grundgebiet mit $(2^m+1)^2$ Freiheitsgraden und die Teilungsstrategie der Viertelung. Zudem beschränken wir uns auf die Analyse der Berechnungsschritte zur Erstellung und Lösung der Knotengleichungssysteme, die asymptotisch die meisten Kosten verursachen.

Als Testumgebung für die Analyse verwenden wir ein Workstationnetz das in vielen Universitäten und anderen Forschungseinrichtungen vorhanden ist. Für dieses Rechnernetz gelten die folgende Annahmen und technischen Daten. [4]. Die Workstation sind mit einem oder mehreren Ethernets verbunden. Jede Workstation kann 1 Mio. Additionen und 1 Mio. Multiplikationen gleichzeitig pro Sekunde ausführen und hat einen Hauptspeicher von 32 MByte. Eine Gleitpunktzahl ist 8

[4] Die technischen Daten sind durch Messungen aus den den verwendeten Workstationnetzen an der TU München entnommen worden

Byte lang. Die Übertragungsrate des Netzwerkes ist 1 MByte pro Sekunde. Auf allen Rechnereinheiten läuft je ein ARESO-Prozeß. Diese technischen Daten entsprechen den in den praktischen Tests verwendeten Rechnern und den tatsächlich erzielten Rechnerleistungen.

Das Verbindungsnetz zwischen den Rechnern ist ein hierarchisch strukturiertes Netzwerk, das die Rechner in verschiedene Cluster und Subcluster einteilt. Dadurch kann die Kommunikation in den verschiedenen Subclustern voneinander unabhängig und somit parallel stattfinden. Wir gehen in der folgenden Analyse davon aus, daß alle Datenpakete von einer Baumebene zur nächsthöheren gleichzeitig verschickt werden. Außerdem nehmen wir an, daß das Versenden von Datenpaketen in verschiedenen Teilbäumen voneinander unabhängig ist und keine Kollisionen verursacht.

Die Untersuchung geeigneter Topologien, die eine solche parallele Kommunikation ermöglichen, ist nicht Gegenstand der vorliegenden Arbeit, wird aber in [53] behandelt. Darin wird unter anderem das ARESO-Programm auf verschiedenen Netztopologien (Binärbaum und Dualhypercube) getestet.

Für die Abbildung der Knoten des Substrukturierungsbaumes auf die Rechner werden folgende drei Strategien untersucht:

Verteilungsstrategie 1 (VS1): Alle Knoten des Substrukturierungsbaumes werden einer einzigen Workstation zugeordnet, d.h. die Simulation wird sequentiell ausgeführt.

Verteilungsstrategie 2 (VS2): Jeder Knoten bis zu einer Ebene s wird einem anderen ARESO-Prozeß, d.h. einer anderen Workstation, zugeordnet. Jeder Knoten ab der Ebene $s+1$ wird dem ARESO-Prozeß zugeordnet, der seinen Vaterknoten verwaltet.

Verteilungsstrategie 3 (VS3): Die Knoten einer bestimmten Ebene s werden den verschiedenen ARESO-Prozessen zugeordnet. Jeder Knoten ab der Ebene $s+1$ wird dem ARESO-Prozeß zugeordnet, der seinen Vaterknoten verwaltet. Ein Knoten aus einer Ebene 0 bis $s-1$ wird von einem der ARESO-Prozesse verwaltet, die seine Sohnknoten verwalten und der bzgl. des Speicherplatzes am wenigsten ausgelastet ist.

In den Abbildungen 74 und 75 sind die Verteilungsstrategien VS2 und VS3 für 21 bzw. 16 Rechner zu sehen. Bei der Verteilungsstrategie VS3 werden in den oberen Baumebenen nur drei Sohnknoten auf eine andere Rechnereinheit verschickt. Ein Sohnknoten bleibt bei dem ARESO-Prozeß des Vaterknotens. Da der Vater auf die Ergebnisse aller Söhne warten muß, entstehen keine zusätzlichen sequentiellen Berechnungsschritte oder Wartezeiten im Vergleich zu Strategie VS2. Für jeden Baumknoten wird aber der Datentransfer zu einem Sohnknoten eingespart. Der Speicher akkumuliert sich bei dem Rechner, der den Wurzelknoten zugewiesen bekommt. Dieser Rechner wird dadurch zum Engpaß der Berechnung.

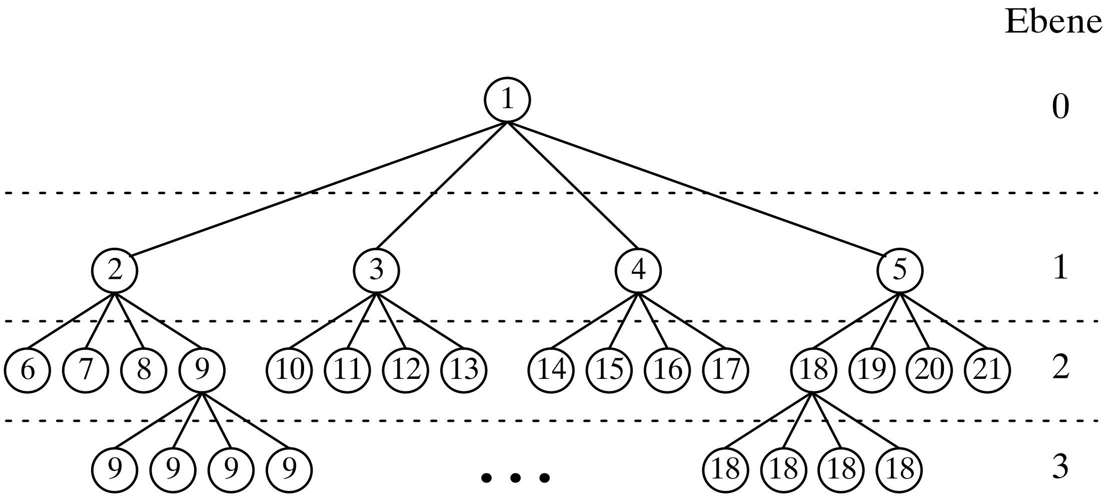


Abbildung 74: Verteilungsstrategie VS2 für 21 Rechner

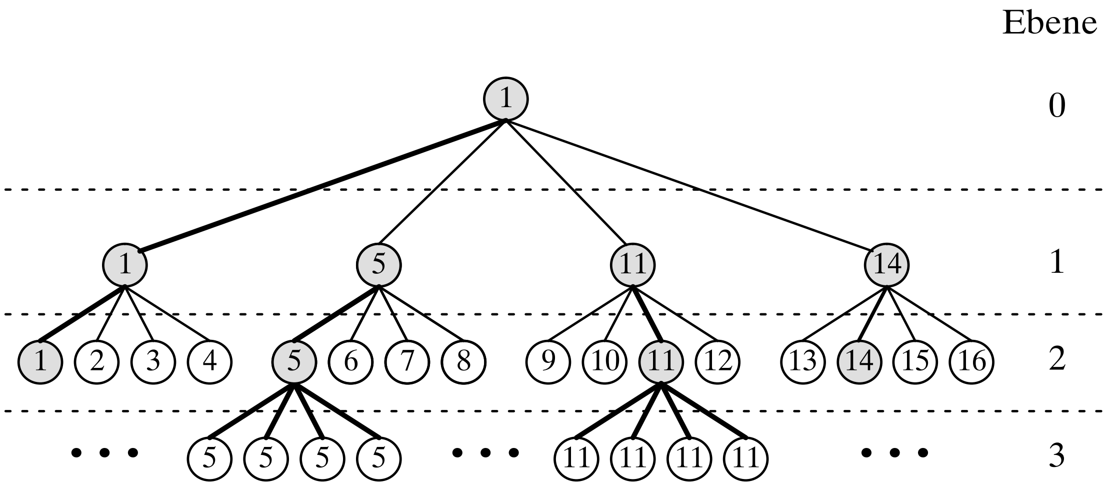


Abbildung 75: Verteilungsstrategie VS3 für 16 Rechner

Zur Beurteilung der Verteilungsstrategien werden folgende Bezeichnungen eingeführt:

$m_{MAX}(VS)$: Parameter für das größte berechenbare Problem mit der Verteilungsstrategie VS, d.h. der zugehörige Substrukturierungsbaum besteht aus $m_{MAX}(VS)$ Ebenen.

$SP_{MAX}(VS, m)$: Maximale Speicherplatzbelastung, die bei einem Substrukturierungsbaum der Tiefe m unter der Verteilungsstrategie VS auf einer Rechnereinheit auftritt.

$R(VS, m)$: Rechenzeit, die zur Erstellung und Lösung der Knotengleichungssysteme erforderlich ist, wenn die Kommunikation keine Zeit beanspruchen würde. Die Rechenzeit wird in Sekunden angegeben.

$K(VS, m)$: Zeit für die Kommunikation der Datenpakete über die Netzverbindungen.

$T(VS, m)$: Gesamtrechenzeit ($T(VS, m) = R(VS, m) + K(VS, m)$).

$r(x)$: Zeit, die zur Ausführung von x Gleitpunktoperationen bei der Leistung von 1 Mio Additionen und 1 Mio Multiplikationen pro Sekunde benötigt wird.

$k(x)$: Zeit, in der eine Datenmenge von x Gleitpunktzahlen bei einer Übertragungsrate von 1 MByte pro Sekunde kommuniziert wird.

$DP(K)$: Länge des Datenpakets, das ein Knoten K versendet.

$SB(K)$: Substrukturierungsbaum, dessen Wurzel Knoten K ist.

Der asymptotisch größte Aufwand an Kommunikation entsteht bei der Assemblierung im zweiten Berechnungsschritt (s. Tabelle 13 auf Seite 117). Die Länge eines zu kommunizierenden Datenpakets ist im wesentlichen abhängig von der Anzahl der Einträge in der Matrix K_{aa}, die bei der Assemblierung von den Sohnknoten zum Vaterknoten geschickt wird.

Beim direkten Lösungsverfahren ist die Anzahl der Einträge durch $n_a^2/2$ definiert und kann für die verschiedenen Baumebenen aus Tabelle 3 auf Seite 99 abgeleitet werden. Es sind volle Matrizen mit der Größe $n_a^2/2$ zu verschicken. Beim iterativen Verfahren ist die Anzahl der Matrixeinträge von K_{aa} der Formel (109) zu entnehmen. Auf Grund der Speichertechnik ist

$$DP_{HB}(K_{aa}) = 3 \cdot n_a(6 + m) \tag{143}$$

und somit von der Ordnung $O(n_a log(n_a))$.

In Tabelle 14 sind die Längen der Datenpakete für die größten Knoten der oberen Ebenen des Substrukturierungsbaumes sowohl für den direkten als auch für den iterativen Lösungsalgorithmus aufgeführt. Bei der letzten Spalte wird (143) und die Tatsache berücksichtigt, daß die Tiefe der Teilbäume sich von Ebene zu Ebene reduziert.

Eb.	Anz. Knoten	$\frac{n_a}{\sqrt{n}}$	$\frac{DP_{RS}(Knoten)}{n}$	$\frac{DP_{HB}(Knoten)}{\sqrt{n}}$
0	1	-	-	-
1	4	1	0.5	$3 \cdot (m+5)$
2	4	4/4	0.5	$3 \cdot (m+4)$
3	36	4/8	0.125	$3/2 \cdot (m+3)$
4	196	4/16	0.031	$3/4 \cdot (m+2)$

Tabelle 14: Länge der größten Datenpakete in den oberen Baumebenen bei direktem und iterativem Lösungsverfahren

8.3.1 Verteilte Simulation mit dem direkten Lösungsverfahren

Zunächst untersuchen wir die verteilte Simulation mit dem direkten Lösungsalgorithmus. Hier lauten die Abschätzungen für die Berechnungskosten (s. Kap. 7.1)

$$SP_{RS}(Baum) \leq (54.7 + 9 \cdot (ld(n) - 8)) \cdot n \tag{144}$$

$$OP_{RS}(Baum) \leq 21.2 \cdot n^{\frac{3}{2}}. \tag{145}$$

Bei der Verteilungsstrategie VS1 ergibt sich für $m = 7$ ein Speicherbedarf von 14.7 MByte. Das nächstgrößere Problem ist nicht mehr auf einer Workstation mit unseren Annahmen berechenbar, da $SP_{MAX}(VS1, 8) = 67$ MByte.

Die sequentielle Rechenzeit ergibt sich aus (145) und ist für einige Problemgrößen in Tabelle 15 angegeben.

m	5	6	7
T(VS1,m)	0.762	5.82	45.5

Tabelle 15: Sequentielle Berechnungszeit bei VS1 und direkten Löser

Für Verteilungsstrategie VS2 betrachten wir unser Testbeispiel auf einem Rechnernetz, das aus 85 Workstations besteht. In diesem Fall wird jeder Knoten der Ebene $0 \leq s \leq 3$ einem eigenen ARESO-Prozeß zugeordnet. Alle Knoten der Ebene $s \geq 4$ werden demselben ARESO-Prozeß wie der Vaterknoten zugeteilt. Der maximale Speicherbedarf entsteht entweder am Wurzelknoten oder an einem Teilbaum der Ebene 3. Mit den Werten der Ebene 3 und 4 aus Tabelle 3 sowie Formel (104) ergibt sich

$$SP_{MAX}(VS2, m) \quad = \quad \max(2 \cdot n, (0.5622 + 9(ld(n) - 8) \cdot 4^{-3}) \cdot n).$$

Damit ist $m_{MAX}(VS2) = 10$ und $SP_{MAX}(VS2, 10) = 18.9$ MByte.

Die Anzahl der Gleitpunktoperationen ergibt sich durch die Summe der maximalen Knoten der obersten drei Ebenen und den Operationen des maximalen Teilbaumes der Ebene 3 (s. Tab. 3). Da sich der Rechenaufwand in den unteren Ebenen pro Baumebene halbiert, kann als obere Schranke für die Operationen eines Teilbaumes der Ebene 3 die Operationszahl des maximalen Knotens der Ebene 3 mit dem Faktor 2 multipliziert werden.

$$R(VS2, m) \quad = \quad r(3.65 \cdot n^{\frac{3}{2}}) \tag{146}$$

Bei der Kommunikation sind nur die Datentransfers zwischen den obersten vier Ebenen zu berücksichtigen. Unter der Annahme, daß die Kommunikation in verschiedenen Teilbäumen voneinander unabhängig läuft, sind nur die maximalen Kommunikationszeiten der Ebenen zu addieren. Jeder Knoten der Ebene 0 bis 2 empfängt vier Datenpakete von seinen Sohnknoten. Da wir in unserer Analyse eine uniforme Verfeinerung unterstellen, ergibt sich die Kommunikationszeit einer Ebene durch das maximale Datenpaket der Ebene multipliziert mit dem Faktor 4 und die gesamte Kommunikationszeit zu

$$K(VS2, m) \quad = \quad k(4.5 \cdot n). \tag{147}$$

Die Beschleunigung ist selbst ohne Berücksichtigung der Kommunikation wegen (145) und (146) mit $21.2/3.65 \approx 6$ beschränkt.

m	7	8	9	10
$R(VS2, m)$	7.8	62	493	3931
$K(VS2, m)$	0.6	2.4	9.5	38
$T(VS2, m)$	8.4	64	503	3969

Tabelle 16: Berechnungszeit bei VS2, direktem Löser und 85 Workstations

Will man den Baum eine Stufe weiter parallelisieren, benötigt man für VS2 341 Rechnereinheiten. Dann gilt:

$$\begin{aligned} SP_{MAX}(VS2, m) &= \max(2 \cdot n, (0.0703 + 9(ld(n) - 8) \cdot 4^{-4}) \cdot n) \\ R(VS2, m) &= r(3.60 \cdot n^{\frac{3}{2}}) \\ K(VS2, m) &= k(4.625 \cdot n) \end{aligned}$$

Hier ist keine wesentliche Beschleunigung im Vergleich zu 85 Workstations möglich (s. Tab. 16 und 17) und auch die maximal berechenbare Problemgröße bleibt unverändert, da der Wurzelknoten bei $m = 11$ bereits 67 MByte Speicherplatz benötigt.

Bei der Verteilungsstrategie VS3 angewandt auf einem Rechnernetz von 64 Workstations sammelt sich der meiste Speicherbedarf an dem Rechner, der den

m	7	8	9	10
$R(VS2, m)$	7.7	61	486	3877
$K(VS2, m)$	0.6	2.4	9.7	38
$T(VS2, m)$	8.3	63	496	3915

Tabelle 17: Berechnungszeit bei VS2, direktem Löser und 341 Workstations

Wurzelknoten verwaltet. Er speichert bis zur Baumebene 2 je einen Knoten pro Ebene und ab der Baumebene 3 einen gesamten Teilbaum ab. Mit den Werten aus Tabelle 3 ergibt sich

$$SP_{MAX}(VS3, m) \quad = \quad (5.69 + 9(ld(n) - 8) \cdot 4^{-3}) \cdot n.$$

Damit ist bereits bei $m_{MAX}(VS3) = 9$ mit $SP_{MAX}(VS3, 9) = 14.9$ MByte die maximale Problemgröße erreicht. Auch der Einsatz der vierfachen Anzahl an Rechnern bringt keine Verbesserung hinsichtlich der berechenbaren Problemgröße, da der Engpaß sich bereits in den oberen drei Ebenen angesammelt hat.

In Tabelle 18 ist ein Überblick über die maximal berechenbaren Probleme bei den verschiedenen Verteilungsstrategien und dem direkten Lösungsverfahren zu sehen.

Strategie	Anz. Rechner	m_{MAX}	SP_{MAX} in MByte
VS1	1	7	15
VS2	85	10	19
	341	10	17
VS3	64	9	15
	256	9	15

Tabelle 18: Überblick über die maximal berechenbaren Probleme mit dem direkten Löser

In [63] werden weitere Verteilungsstrategien diskutiert, wie z.B. der Einsatz von großen Vektorrechnern für die obersten drei Ebenen, eine feingranulare Parallelisierung von großen Baumknoten oder die Verwendung von Blättern des Substrukturierungsbaumes die aus mehreren Finiten Elementen bestehen. Außerdem werden die hier analysierten Verteilungsstrategien am Beispiel von einem Workstationcluster mit 128 Rechnern unter Verwendung der Teilungsstrategie der Bisektion betrachtet. Dabei zeigt sich ebenfalls, daß die maximal berechenbare Problemgröße deutlich gegenüber einem sequentiellen Berechnungslauf ansteigt. Die Beschleunigung ist aber auch hier mit dem Faktor 7 begrenzt.

Die eher pessimistischen Vorhersagen dieses Kapitels für eine verteilte Simulation des direkten Lösungsverfahrens beziehen sich allerdings nur auf den Berechnungsschritt 2_{RS}. Da eine FE-Berechnung basierend auf der rekursiven

Substrukturierung, aber in den anderen Berechnungsschritten, vor allem in den Operationen auf Elementebene, die Parallelität voll ausnutzen kann, ergeben die praktisch durchgeführten numerischen Simulationen bessere Beschleunigungen als in der theoretischen Analyse. Für den asymptotischen Fall gelten jedoch die obigen Abschätzungen, d.h. die mögliche parallele Beschleunigung ist begrenzt und verhindert damit die Skalierbarkeit des direkten Algorithmus auf eine größere Anzahl von Rechnereinheiten.

8.3.2 Verteilte Simulation mit dem iterativen Lösungsverfahren

In diesem Abschnitt wird gezeigt, daß sich der Einsatz des iterativen Lösungsverfahrens positiv auf eine parallele FE-Berechnung auswirkt. Im Vergleich zum direkten Lösungsverfahren sind größere Probleme berechenbar. Zudem ist die Parallelisierung bezüglich der Anzahl der eingesetzten Rechner besser skalierbar.

Zuerst betrachten wir die sequentielle Version des Verfahren unter VS1. Hier lauten die Abschätzungen für die Berechnungskosten (vgl. (128), (142))

$$SP_{HB}(Baum) \leq 100.5 \cdot n \tag{148}$$

$$OP_{HB}(Baum) \leq 87.75 \cdot n \cdot n_{it}. \tag{149}$$

Die Anzahl der Iterationen n_{it} ist abhängig von dem zu berechnenden Problem. Wir verwenden hier Werte die aus praktisch durchgeführten Simulationen stammen und für die meisten berechneten Probleme repräsentativ sind. Es wird unterschieden, ob es sich um ein Wärmeleitungsproblem (WP) oder ein Elastizitätsproblem (EP) handelt. Bei Wärmeleitungsproblemen sind in unseren Beispielen (s. Kap. 9) bei einer uniformen Verfeinerung ca. vier zusätzliche Iterationen notwendig, bei Elastizitätsproblemen sind mehr Iterationen notwendig. Wir nehmen deshalb in unserer Analyse folgende Iterationszahlen an:

$$\begin{aligned} n_{it} &= 4 \cdot m, && \text{bei} \quad \text{Wärmeleitungsproblemen} \\ n_{it} &= 15 \cdot m + 10, && \text{bei} \quad \text{Elastizitätsproblemen.} \end{aligned}$$

Es ergeben sich $m_{MAX}(VS1) = 7$ und $SP_{MAX}(VS1, 7) = 13.4$ MByte. Dies ist der gleiche Wert wie beim direkten Verfahren (s. Kap. 8.3.1). Das nächstgrößere Problem benötigt allerdings nur $SP_{MAX}(VS1, 8) = 53$ MByte.

Die Rechenzeit ergibt sich allein aus der Anzahl der Gleitpunktoperationen, da keine Kommunikation erforderlich ist. In Tabelle 19 ist die sequentielle Rechenzeit für beide Probleme (WP,EP) bei drei unterschiedlichen Problemgrößen zu sehen.

Betrachten wir nun die Verteilungsstrategie VS2 mit einem Rechnernetz von 85 Workstations. In diesem Fall wird jeder Knoten der Ebene $0 \leq s \leq 3$ einem eigenen ARESO-Prozeß zugeordnet. Alle Knoten der Ebene $s \geq 4$ werden demselben ARESO-Prozeß wie der Vaterknoten zugeteilt. Der maximale Speicherbedarf

m	5	6	7
$T_{WP}(VS1,m)$	1.9	8.9	41
$T_{EP}(VS1,m)$	8.1	37	168

Tabelle 19: Sequentielle Berechnungszeit bei VS1 und iterativen Verfahren

entsteht entweder am Wurzelknoten oder an einem Teilbaum der Ebene 3. Mit (148) und Tabelle 9 erhält man

$$\begin{aligned} SP_{MAX}(VS2,m) &= \max(SP_{HB}(Wurzel), \max_{K\in EB_3} SP_{HB}(SB(K))) \\ &= \max(3\cdot(18+2\cdot m)\cdot 2^m, 100.5\cdot(2^{m-3}+1)^2). \end{aligned}$$

Damit ist $m_{MAX}(VS2) = 10$ und $SP_{MAX}(VS2,10) = 13.4$ MByte. Durch den Einsatz mehrerer Workstations sind also deutlich größere Probleme berechenbar.

Bei einer Steigerung der Rechnerzahl auf 341 ergibt sich als Speicherbedarf eines Teilbaumes der vierten Ebene

$$\max_{K\in EB_4} SP_{RS}(SB(K) = 100.5\cdot(2^{m-4}+1)^2.$$

Dadurch steigt $m_{MAX}(VS2)$ auf 11 bei gleichbleibender maximaler Speicherbelastung. Hier macht sich das deutlich verlangsamte Wachstum am Wurzelknoten im Vergleich zum direkten Algorithmus bemerkbar (s. Kap. 8.3.1). Beim iterativen Lösungsverfahren erreicht der Speicherbedarf des Wurzelknoten bei $m = 15$ erst 38 MByte. Erst ab einer Problemgröße von $m = 16$ kann die Berechnung wegen Speichermangel im Wurzelknoten nicht mehr durchgeführt werden. Bis zu dieser Problemgröße ist bei einer Vervierfachung der Rechneranzahl auch die vierfache Problemgröße berechenbar.

Die Anzahl der Gleitpunktoperationen bei 85 Rechnern ergibt sich durch die Summe der maximalen Knoten der obersten drei Ebenen (s. Tab. 9) und den Operationen des maximalen Teilbaumes der Ebene 3.

$$\begin{aligned} R(VS2,m) &= r(\ \sum_{s=0}^{2} \max_{K\in EB_s} OP_{HB}(K) + \max_{K\in EB_3} OP_{HB}(SB(K))\ ) \\ &= r(\ n_{it}\cdot((59.5+7m)\cdot 2^m + 87.75\cdot(2^{m-3}+1)^2)\ ) \end{aligned}$$

Bei der Kommunikation sind nur die Datentransfers zwischen den obersten vier Ebenen zu berücksichtigen. Unter der Annahme, daß die Kommunikation in verschiedenen Teilbäumen voneinander unabhängig läuft, ist die Kommunikationszeit nur abhängig von der Anzahl der kommunizierenden Ebenen. Es gibt zwei Arten von Datentransfers. In Schritt 2_{RSI} werden einmal die Matrizen K_{aa} von den Sohnknoten an die Vaterknoten verschickt. Während der Iteration zur Lösung der Gleichungssysteme verschickt pro Iterationsschritt jeder Vaterknoten den Vektor u an seine vier Söhne und empfängt die Vektoren f_a von seinen Söhnen. Mit den

Werten aus der Tabelle 4 für die Länge der Datenpakete von K_{aa} und Tabelle 3 für die Länge der Vektoren u und f_a erhält man als Kommunikationszeit

$$\begin{aligned} K(VS2,m) &= k(\quad 4 \cdot \sum_{s=1}^{3} \max_{K \in EB_s} DP_{HB}(K) \quad + \\ &\quad + \quad n_{it} \cdot 4 \cdot (\sum_{s=1}^{3} \max_{K \in EB_s} DP_{HB}(f_a) + \sum_{s=0}^{2} \max_{K \in EB_s} DP_{HB}(u)) \quad) \\ &= k(\quad 4 \cdot n_m((7.5m + 31.5) + 8 \cdot n_{it}) \quad). \end{aligned}$$

Will man den Baum eine Stufe weiter parallelisieren, benötigt man für die Verteilungsstrategie VS2 341 Rechnereinheiten. Dann gilt:

$$\begin{aligned} R(VS2,m) &= r(\quad n_{it} \cdot ((63.25 + 7.5m) \cdot 2^m + 87.75 \cdot (2^{m-4} + 1)^2) \quad) \\ K(VS2,m) &= k(4 \cdot n_m((8.25m + 33) + 9 \cdot n_{it})) \end{aligned}$$

Wenden wir uns nun der Verteilungsstrategie VS3 zu. Bei dieser Verteilungsstrategie sammelt sich der meiste Speicherbedarf an dem Knoten, der den Wurzelknoten verwaltet. Bei einem Rechnernetz von 64 Workstation speichert er bis zur Ebene 2 je einen Knoten pro Ebene und ab der Baumebene 3 einen gesamten Teilbaum.

$$\begin{aligned} SP_{MAX}(VS3,m) &= \sum_{i=0}^{2} \min SP_{HB}(Knoten) + \min_{K \in EB_3} SP_{HB}(SB(K))) \\ &= 3 \cdot 2^m \cdot (36.5 + 5m) + 100.5 \cdot (2^{m-3} + 1)^2 \end{aligned}$$

Damit ist bei $m_{MAX}(VS3) = 10$ mit $SP_{MAX}(VS3,10) = 15.5$ MByte die maximale Problemgröße erreicht. Der Einsatz der vierfachen Anzahl an Rechnern bringt eine weitere Verbesserung der berechenbaren Problemgröße.

$$SP_{MAX}(VS3,m) = 3 \cdot 2^m \cdot (39.75 + 5.5m) + 100.5 \cdot (2^{m-4} + 1)^2$$

Hier erreicht man bei $m_{MAX}(VS3) = 11$ die maximale Speicherbelastung von $SP_{MAX}(VS3,11) = 18.3$ MByte. Dabei macht sich die deutliche Einsparung an Speicherbedarf in den Knoten der oberen Baumebenen im Vergleich zu direkten Verfahren bemerkbar. Durch eine Erhöhung der Anzahl der eingesetzten Rechner können größere Probleme berechnet werden. Beim direkten Verfahren gilt dies nur bei kleinen Problemen (s. Kap. 8.3.1).

Die reine Berechnungszeit ändert sich im Vergleich zu VS2 nicht. Die Gesamtkosten für den Datentransfer nehmen zwar ab, da die Kommunikation zu einem Sohn wegfällt, die Kommunikationszeit ist aber unter unseren Annahmen wie bei VS2 nur abhängig von der Anzahl der kommunizierenden Ebenen. Demzufolge gilt

$$R(VS3,m) = R(VS2,m) \tag{150}$$

$$K(VS3, m) = K(VS2, m). \tag{151}$$

Die theoretischen Berechnungszeiten für den iterativen Lösungsalgorithmus unter der Verteilungsstrategie VS3 sind in den Tabellen 20, 21 und 22 zu sehen.

m	7	8	9	10
$R_{WP}(VS3, m)$	1.1	4.0	16	64
$K_{WP}(VS3, m)$	1.3	2.8	6.3	14
$T_{WP}(VS3, m)$	2.4	6.8	24	78

Tabelle 20: Berechnungszeit für WP bei VS3, iterativem Löser und 64 Workstations

m	7	8	9	10	11
$R_{WP}(VS3, m)$	0.6	1.8	5.9	20	77
$K_{WP}(VS3, m)$	1.4	3.2	7.0	16	34
$T_{WP}(VS3, m)$	2.0	5.0	13	36	111

Tabelle 21: Berechnungszeit für WP bei VS3, iterativem Löser und 256 Workstations

m	7	8	9	10	11
$R_{EP}(VS3, m)$	2.5	7.4	24	82	308
$K_{EP}(VS3, m)$	4.6	10	23	51	111
$T_{EP}(VS3, m)$	7.1	17	47	133	419

Tabelle 22: Berechnungszeit für das Elastizitätsproblem bei VS3, iterativem Löser und 256 Workstations

Insgesamt liegen für VS2 und VS3 die reinen Berechnungszeiten sowie die Kommunikationszeiten deutlich unter den Werten, die sich bei der Analyse des direkten Lösers ergeben. Zusätzlich bietet sich aufgrund des reduzierten Speicherbedarfs und des verringerten Rechenaufwands die Möglichkeit, bei einer gleichzeitigen Erhöhung der Problemgröße und der Anzahl der eingesetzten Rechner, das vergrößerte Problem in annähernd der selben Zeit zu berechnen. Diese Skalierbarkeit des Algorithmus ist bei dem direkten Lösungsverfahren durch das starke Anwachsen der oberen Baumknoten nicht gegeben.

Zum Abschluß dieses Abschnitts ist in Tabelle 23 ein Überblick über die maximal berechenbaren Probleme bei den verschiedenen Verteilungsstrategien und dem iterativen Lösungsverfahren zu sehen. Im Gegensatz zum direkten Verfahren (s. Tab 18) bedeutet hier eine Steigerung der Anzahl der eingesetzten Rechner auch die Möglichkeit ein entsprechend größeres Problem zu berechnen.

Strategie	Anz. Rechner	m_{MAX}	SP_{MAX} in MByte
VS1	1	7	13
VS2	85	10	13
	341	11	13
VS3	64	10	16
	256	11	18

Tabelle 23: Überblick über die maximal berechenbaren Probleme mit dem iterativen Löser

9 Numerische Ergebnisse

In diesem Kapitel werden einige praktisch durchgeführte Finite-Element-Berechnungen von ausgewählten Modellproblemen vorgestellt.

9.1 Die Testbeispiele

Der erste Typ von elliptischen Randwertproblemen sind stationäre Wärmeleitungsprobleme. Dabei handelt es sich um die Laplace-Gleichung mit Dirichlet-Randbedingungen.

$$\begin{aligned} -\triangle u &= f(x,y) \quad in \quad \Omega \\ u &= g(x,y) \quad auf \quad \Gamma = \partial\Omega \end{aligned} \tag{152}$$

Wir betrachten dabei den Spezialfall $f(x,y) = 0$.

Beispiel 1 (HEAT1): Die Laplace-Gleichung im Einheitsquadrat $(0,1) \times (0,1)$ und der Randbedingung
$g(x,y) = \frac{1}{2} \cdot cos(4\pi(x-y))\frac{sinh(4\pi(x+y))}{sinh(8\pi)}$.

Beispiel 2 (HEAT2): Die Laplace-Gleichung im Einheitsquadrat $(0,1) \times (0,1)$ und der Randbedingung
$g(x,y) = sin(\pi x) \cdot \frac{sinh(\pi(1-y))}{sinh(\pi)}$.

Beispiel 3 (HEAT3): Die Laplace-Gleichung in einem L-förmigen Grundgebiet $(-1,1)^2 \setminus (-1,0)^2$ und der Randbedingung
$g(x,y) = Re((x+iy)^{\frac{2}{3}})$.

Da die Randfunktionen der Beispiele harmonisch im Gebietsinneren sind, ergeben sie auch die analytische Lösung des Dirichlet-Problems. Die Abbildungen 76, 77 und 78 zeigen den Lösungsverlauf der Beispiele. Bei den ersten beiden Problemen verläuft die Lösung glatt. Sie hat aber in Beispiel 1 einen starken Anstieg in der nordöstlichen Ecke des Grundgebiets. Das dritte Beispiel enthält eine Singularität in der Lösung, die durch die einspringende Ecke bedingt ist.

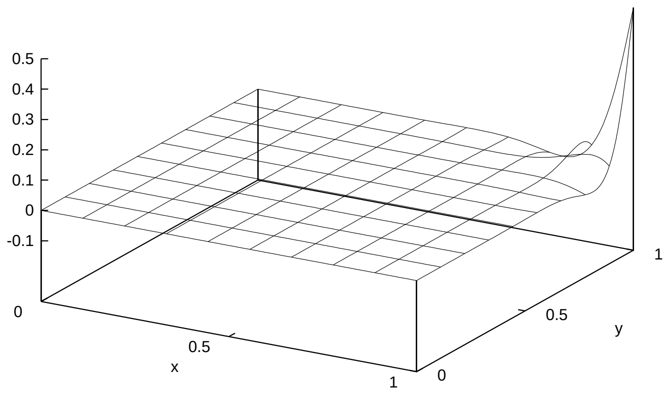


Abbildung 76: Beispiel 1 für Wärmeleitung (HEAT1)

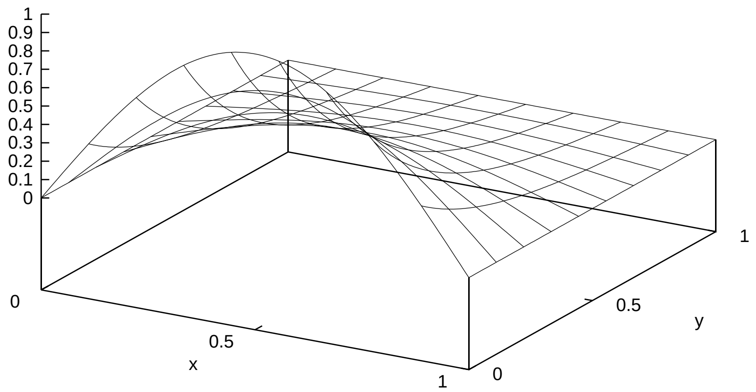


Abbildung 77: Beispiel 2 für Wärmeleitung (HEAT2)

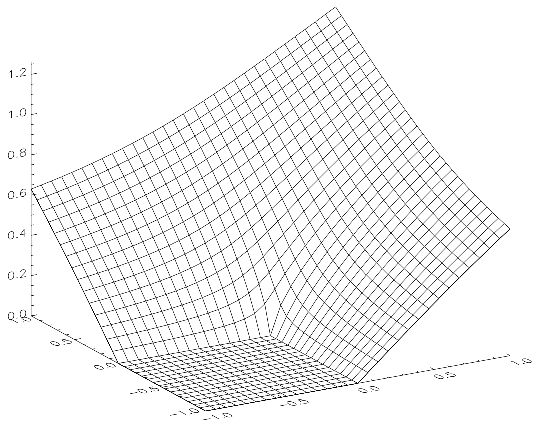


Abbildung 78: Beispiel 3 für Wärmeleitung (HEAT3)

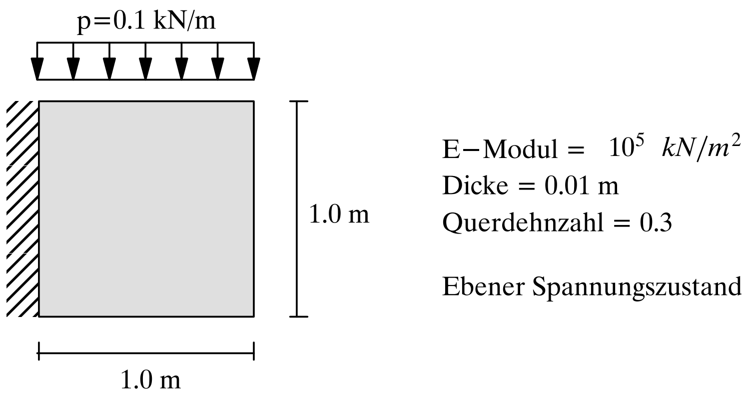


Abbildung 79: Statisches Modell einer Kragscheibe

Beim zweiten betrachteten Typ von Randwertproblemen handelt es sich um Scheibenprobleme aus dem Gebiet der ebenen Elastizität (s. Kap 2.2).

Beispiel 4 (PANEL1): Eine Kragscheibe im ebenen Spannungszustand die am linken Rand eingespannt ist und durch eine konstante Last von oben beansprucht wird. Die genauen Daten, wie der Elastizitätsmodul E, die Poisson-Zahl ν und die Scheibendicke, sind aus Abb. 79 zu entnehmen.

Beispiel 5 (PANEL2): Eine Winkelscheibe mit L-förmigen Grundgebiet unter Zugbelastung. Auf die Scheibe wirkt an der linken Seite eine konstante Last als Neumann-Randbedingung. Die obere Kante ist horizontal beweglich gelagert und vertikal festgehalten. Die Ausmaße und die physikalischen Eigenschaften sind in Abb. 5 zu sehen.

Bei allen Testläufe werden folgende Abkürzungen verwendet:

θ = Abbruchschranke
ν = Anzahl der Glättungsschritte
unsym = iterativer Grund-Algorithmus (unsymmetrisch, $\nu = 1$)
sym = Symmetrischer iterativer Algorithmus
Konvergenzrate: $\boxed{\varrho_m = \frac{||\Delta x_m||}{||\Delta x_{m-1}||}}$
NEL = Anzahl der Finiten Elemente
FG = Anzahl der Freiheitsgrade
n_{it} = Anzahl der Iterationen zur Lösung der Gleichungssysteme

9.2 Numerische Simulation der Testbeispiele

9.2.1 Uniforme Verfeinerung

Zuerst betrachten wir die Zahl der Iterationen und die Konvergenzraten bei einer uniformen Verfeinerung, d.h. bei einem schrittweisen Anwachsen der Anzahl der Unbekannten mit dem Faktor 4. In den folgenden Tabellen stehen die Zahl der Iterationen die benötigt werden, um die euklidische Norm des Residuums auf ein vorgegebenes θ zu reduzieren. Soweit nichts anderes vermerkt ist, handelt es sich um Testläufe mit dem unsymmetrischen iterativen Algorithmus und einem Glättungsschritt pro Baumdurchlauf.

In Tabelle 24 sieht man die Iterationszahlen und Konvergenzraten für das Beispiel HEAT2. Man erkennt, daß sich bei einer Verfeinerung des Systems die Zahl der erforderlichen Iterationsschritte zum Erzielen der geforderten Genauigkeit etwa um eine konstante Zahl erhöht. Dieses konstante Anwachsen der Iterationszahlen bei einer Vervierfachung der Problemgröße entspricht dem in Kapitel 5.4 unterstellten Konvergenzverhalten von logarithmischer Ordnung. Die Iterationszahlen dieser und der folgenden praktischen numerischen Simulationen liegen dabei unter dem theoretisch erforderlichen Wachstum von $O(log(n)^2)$.

NEL	FG	ϱ_m	n_{it} $\theta = 10^{-4}$	n_{it} $\theta = 10^{-5}$	n_{it} $\theta = 10^{-6}$
16	25	0.54	14	18	21
64	81	0.52	12	16	20
256	289	0.60	13	17	21
1024	1089	0.64	15	20	25
4096	4225	0.72	19	25	32
16.384	16.641	0.78	23	31	40

Tabelle 24: Iterationszahlen und Konvergenzraten für Beispiel HEAT2

Auch bei dem zweiten Beispiel HEAT1 ergeben sich ähnliche Iterationszahlen und ebenfalls ein Konvergenzverhalten von logarithmischer Ordnung (s. Tab. 25). Die Konvergenzrate ist etwas schlechter als beim Beispiel HEAT2, da wegen des starken lokalen Charakters der Lösung etwas länger iteriert wird, obwohl in den meisten Teilgebieten keine Verbesserung der Lösung möglich ist. Im asymptotischen Fall sind die Konvergenzraten gleich, da sie nur von der Steifigkeitsmatrix, die in beiden Fällen identisch ist, und nicht von den Randbedingungen abhängen.

Bei den Beispielen aus der ebenen Elastizität sind die Iterationszahlen deutlich höher (s. Tab. 26 und Tab. 27). Durch die Anwendung des symmetrischen Iterationsverfahrens mit zwei Vor- und zwei Nach-Glättungsschritten erzielt man aber auch hier günstigere Iterationszahlen. Die Iterationszahlen wachsen bei der Kragscheibe etwa logarithmisch an. Bei der Winkelscheibe liegen sie am Anfang höher, wachsen dann aber kaum noch an.

NEL	FG	n_{it}	ϱ_m
16	25	14	0.53
64	81	13	0.57
256	289	14	0.60
1024	1089	17	0.69
4096	4225	21	0.76
16.384	16.641	24	0.81

Tabelle 25: Iterationszahlen und Konvergenzraten für Beispiel HEAT1 bei $\theta = 10^{-4}$

NEL	FG	unsym.		sym.,$\nu = 2$	
		n_{it}	ϱ_m	n_{it}	ϱ_m
4	18	34	0.79	15	0.55
16	50	45	0.84	21	0.67
64	162	56	0.88	27	0.76
256	578	70	0.91	35	0.83
1024	2178	85	0.93	43	0.87
4096	8450	101	0.95	52	0.90

Tabelle 26: Iterationszahlen und Konvergenzraten für die Kragscheibe bei $\theta = 10^{-4}$

NEL	FG	unsym.		sym.,$\nu = 2$	
		n_{it}	ϱ_m	n_{it}	ϱ_m
3	16	70	0.91	29	0.76
12	42	90	0.93	39	0.82
48	130	99	0.94	44	0.84
192	450	102	0.94	47	0.86
768	1666	103	0.94	50	0.87
3072	6402	103	0.94	52	0.88

Tabelle 27: Iterationszahlen und Konvergenzraten für die Winkelscheibe bei $\theta = 10^{-4}$

Um die Anwendbarkeit des Verfahrens auf Finite Elemente mit biquadratischen Ansatzfunktionen zu belegen, wird das Beispiel der Kragscheibe mit biquadratischen 9-Knoten Scheibenelementen gerechnet. Da jedes Finite Element neun Stützpunkte mit jeweils zwei Unbekannten besitzt, ist die Anzahl der Unbekannten eines Netzes ca. viermal so hoch wie bei einem Netz aus bilinearen Vier-Knoten-Elementen mit gleicher Elementanzahl. In Tabelle 28 sieht man das Konvergenzverhalten bei symmetrischer und unsymmetrischer Iteration und $\theta = 10^{-4}$. Im Vergleich zu den bilinearen Finiten Elementen liegen die Iterationszahlen bei gleicher Anzahl von Freiheitsgraden etwas höher. Aber auch hier ist ihr Wachstum von logarithmischer Ordnung.

NEL	FG	unsym.		sym.,$\nu = 2$	
		n_{it}	ϱ_m	n_{it}	ϱ_m
4	50	55	0.92	26	0.92
16	162	67	0.94	33	0.94
64	578	81	0.95	41	0.95
256	2178	97	0.95	49	0.95
1024	8450	113	0.95	58	0.95
4096	33.282	130	0.95	67	0.95

Tabelle 28: Iterationszahlen und Konvergenzraten für die Kragscheibe bei biquadratischen Ansatzfunktionen

9.2.2 Adaptive Relaxation auf uniformen Netzen

In diesem Abschnitt betrachten wir Berechnungsläufe, die auf einem uniform verfeinerten Netz eine adaptive Relaxation durchführen. An diesen Läufen erkennt man den Vorteil der adaptiven Relaxation im Vergleich zu dem Verfahren, das in jedem Iterationsschritt alle Baumknoten durchläuft.

Bei der adaptiven Relaxation werden Baumknoten nur besucht, wenn in ihrem Vaterknoten eine Korrektur der Lösung durchgeführt wurde. Korrekturen, die unter der vorgeschriebenen Genauigkeitsschranke liegen, werden nicht durchgeführt. Das Verfahren der adaptiven Relaxation führt zu mehr Glättungsschritten auf den oberen Knoten des Substrukturierungsbaumes. Die Blätter werden dagegen nur sehr selten besucht. Insgesamt spart man an Rechenoperationen bei etwa gleicher Genauigkeit der Berechnung. In den Tabellen 29, 30 und 31 sind die Anzahl der arithmetischen Operationen bei der Gleichungslösung und die Anzahl der Blattdurchläufe n_{Blatt} für verschiedene Beispiele angeführt.

Bei sehr lokalen Phänomenen, wie bei Beispiel HEAT1, kann der Berechnungsaufwand durch den Einsatz der adaptiven Relaxation um den Faktor 10 gesenkt werden (s. Tab. 29). Bei den anderen Beispielen ist die Einsparung zwar geringer, aber auch hier ist die Anzahl der Blattdurchläufe deutlich niedriger als bei kompletten Baumdurchläufen (s. Tab. 30, 31).

NEL	FG	adaptiv		feste Reihenfolge	
		OP	n_{Blatt}	OP	n_{Blatt}
1024	1089	$1.6 \cdot 10^5$	96	$6.3 \cdot 10^5$	4864
1096	4225	$4.4 \cdot 10^5$	192	$3.2 \cdot 10^6$	$2.4 \cdot 10^4$
16.384	16.641	$1.4 \cdot 10^6$	336	$1.5 \cdot 10^7$	$1.1 \cdot 10^5$

Tabelle 29: Beispiel HEAT1, Vergleich von adaptiver Relaxation und kompletten Baumdurchläufen

NEL	FG	adaptiv		feste Reihenfolge	
		OP	n_{Blatt}	OP	n_{Blatt}
768	833	$2.6 \cdot 10^5$	348	$2.9 \cdot 10^5$	2112
3072	3201	$6.4 \cdot 10^5$	588	$1.3 \cdot 10^6$	$1.0 \cdot 10^4$
12.228	12.545	$2.1 \cdot 10^6$	240	$6.8 \cdot 10^6$	$4.9 \cdot 10^4$

Tabelle 30: Beispiel HEAT3, Vergleich von adaptiver Relaxation und kompletten Baumdurchläufen

NEL	FG	adaptiv		feste Reihenfolge	
		OP	n_{Blatt}	OP	n_{Blatt}
256	578	$1.2 \cdot 10^6$	1816	$1.6 \cdot 10^6$	$1.8 \cdot 10^4$
1024	2178	$4.2 \cdot 10^6$	4728	$8.2 \cdot 10^6$	$8.7 \cdot 10^4$
4096	8450	$1.5 \cdot 10^7$	9036	$4.0 \cdot 10^7$	$4.1 \cdot 10^5$
16.384	16.641	$4.6 \cdot 10^7$	$1.0 \cdot 10^4$	$1.9 \cdot 10^8$	$1.9 \cdot 10^6$

Tabelle 31: Beispiel PANEL1, bilineare Ansatzfunktionen, Vergleich von adaptiver Relaxation und kompletten Baumdurchläufen

9.2.3 Adaptive Relaxation auf adaptiven Netzen

In diesem Abschnitt betrachten wir voll adaptive Finite-Element-Berechnungen. Bei den Beispielen werden adaptive Relaxationsschritte auf adaptiv verfeinerten Netzen durchgeführt. Die Verfeinerung der Netze wird durch die adaptiven Relaxationsschritte gesteuert.

Gestartet wird auf einem groben Netz mit einer vorgegebenen Abbruchschranke. Sobald ein Blatt des Substrukturierungsbaumes geglättet wird, d.h. eine Korrektur der Lösung erforderlich ist die über der zulässigen Schranke liegt, wird die Iteration zur Lösung der Gleichungssysteme unterbrochen und das Blatt verfeinert. Anschließend wird mit dem neuen Netz und der aktuellen Näherungslösung die Iteration fortgesetzt.

Durch eine schrittweise Reduktion der Abbruchschranke auf die geforderte Genauigkeit der Berechnung werden nacheinander alle Blätter die einen Mindestbeitrag zur Lösung des Gesamtproblems beisteuern verfeinert. Das Ende des adaptiven Berechnungszyklus ist erreicht, wenn in einem adaptiven Relaxationsschritt kein Blattknoten mehr korrigiert wird.

In den Tabellen 32 bis 35 sind die Abbruchschranken θ, die erzielte Genauigkeit der Lösung, die Anzahl der dafür notwendigen Freiheitsgrade und die Anzahl der arithmetischen Operationen für einige Beispiele aufgeführt.

Dabei stehen in den Tabellen die Ergebnisse von vollständig adaptiven Berechnungsläufen für ein vorgegebenes θ. Diese Berechnungsläufe starten auf einem groben Netz mit 4 Elementen und enden vollautomatisch auf adaptiven Netzen mit der in den Tabellen angegebenen Anzahl von Freiheitsgraden. In den einzelnen Tabellen sieht man zu jedem Beispiel adaptive Läufe mit unterschiedlichen Abbruchschranken. Die Abbruchschranken werden jeweils auf 1/4 reduziert und man erkannt, daß die entstehenden Netze etwa viermal soviel Freiheitsgrade benötigen um die Genauigkeitsanforderung zu erzielen. Dies entspricht dem asymptotischen Konvergenzverhalten der betrachteten Problemklassen.

θ	$\|e\|_\infty$	FG	Anz. Op.
$2.4 \cdot 10^{-4}$	$1.2 \cdot 10^{-3}$	353	$3.3 \cdot 10^{5}$
$6.1 \cdot 10^{-5}$	$3.1 \cdot 10^{-4}$	1361	$1.4 \cdot 10^{6}$
$1.5 \cdot 10^{-5}$	$9.6 \cdot 10^{-5}$	4211	$1.1 \cdot 10^{7}$
$3.8 \cdot 10^{-6}$	$2.2 \cdot 10^{-5}$	18137	$5.2 \cdot 10^{7}$

Tabelle 32: Beispiel HEAT1, vollständig adaptive Läufe für verschiedene Genauigkeitsanforderungen

Um zu verdeutlichen in wievielen Zwischenschritten ein endgültiges FE-Netz zustande kommt, wird in Tabelle 33 der Berechnungslauf von Beispiel HEAT1 mit $\theta = 6.1 \cdot 10^{-5}$ (s. Tab. 32, Zeile 2) genauer aufgeschlüsselt. In jedem Zwischenschritt wird die Abbruchschranke solange verschärft bis ein Blatt relaxiert wird. Anschließend erfolgt ein kompletter Durchlauf durch den adaptiven Baum. Alle

$\theta(Wurzel)$	$\theta(Blatt)$	$\|e\|_\infty$	FG	Anz. Op.
$3.9 \cdot 10^{-3}$	$1.6 \cdot 10^{-2}$	$6.0 \cdot 10^{-2}$	25	$8.1 \cdot 10^{2}$
$2.4 \cdot 10^{-4}$	$3.9 \cdot 10^{-3}$	$3.9 \cdot 10^{-2}$	41	$4.3 \cdot 10^{3}$
$2.4 \cdot 10^{-4}$	$1.6 \cdot 10^{-2}$	$1.2 \cdot 10^{-2}$	125	$2.3 \cdot 10^{4}$
$9.5 \cdot 10^{-7}$	$2.4 \cdot 10^{-4}$	$8.1 \cdot 10^{-3}$	141	$6.7 \cdot 10^{4}$
$9.5 \cdot 10^{-7}$	$9.8 \cdot 10^{-4}$	$2.7 \cdot 10^{-3}$	317	$1.1 \cdot 10^{5}$
$6.0 \cdot 10^{-8}$	$2.4 \cdot 10^{-4}$	$1.1 \cdot 10^{-3}$	637	$2.4 \cdot 10^{5}$
$6.0 \cdot 10^{-8}$	$9.8 \cdot 10^{-4}$	$5.1 \cdot 10^{-4}$	1185	$3.8 \cdot 10^{5}$
$3.7 \cdot 10^{-9}$	$6.1 \cdot 10^{-5}$	$3.1 \cdot 10^{-4}$	1361	$6.3 \cdot 10^{5}$

Tabelle 33: Beispiel HEAT1, Detaillierter Lauf für $\theta = 6.1 \cdot 10^{-5}$

weiteren Blätter die in diesem Durchlauf die Genauigkeitsanforderung verletzen werden verfeinert und ein neuer Zwischenschritt mit einer groben Abbruchschranke gestartet. In der Tabelle 33 sieht man die Abbruchschranken im Wurzelknoten und in dem untersten Blattknoten am Ende jedes Zwischenschrittes. Zusätzlich wird die Genauigkeit der Lösung und die Anzahl der Operationen der Zwischenschritte aufgeführt.

θ	$\|e\|_\infty$	FG	Anz. Op.
$9.8 \cdot 10^{-4}$	$4.7 \cdot 10^{-3}$	257	$3.1 \cdot 10^{5}$
$2.4 \cdot 10^{-4}$	$1.1 \cdot 10^{-3}$	613	$1.5 \cdot 10^{6}$
$6.1 \cdot 10^{-5}$	$3.9 \cdot 10^{-4}$	1649	$7.2 \cdot 10^{6}$
$1.5 \cdot 10^{-5}$	$8.6 \cdot 10^{-5}$	5899	$4.8 \cdot 10^{7}$
$3.8 \cdot 10^{-6}$	$2.8 \cdot 10^{-5}$	23677	$2.7 \cdot 10^{8}$

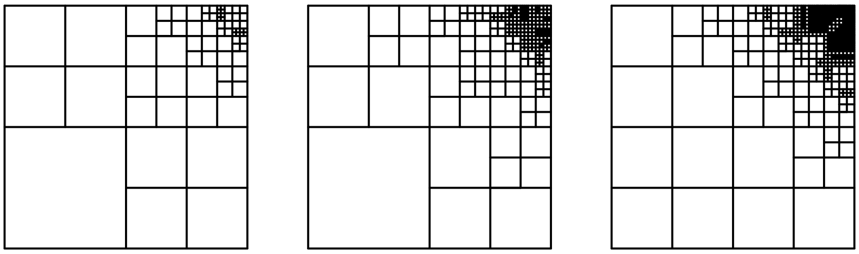
Tabelle 34: Beispiel HEAT3, vollständig adaptive Läufe für verschiedene Genauigkeitsanforderungen

θ	$\|e\|_\infty$	FG	Anz. Op.
$2.4 \cdot 10^{-4}$	$7.6 \cdot 10^{-4}$	206	$4.0 \cdot 10^{5}$
$6.1 \cdot 10^{-5}$	$1.6 \cdot 10^{-4}$	982	$4.0 \cdot 10^{6}$
$1.5 \cdot 10^{-5}$	$4.9 \cdot 10^{-5}$	3486	$1.9 \cdot 10^{7}$
$3.8 \cdot 10^{-6}$	$1.5 \cdot 10^{-5}$	13150	$1.2 \cdot 10^{8}$

Tabelle 35: Beispiel PANEL1, vollständig adaptive Läufe für verschiedene Genauigkeitsanforderungen

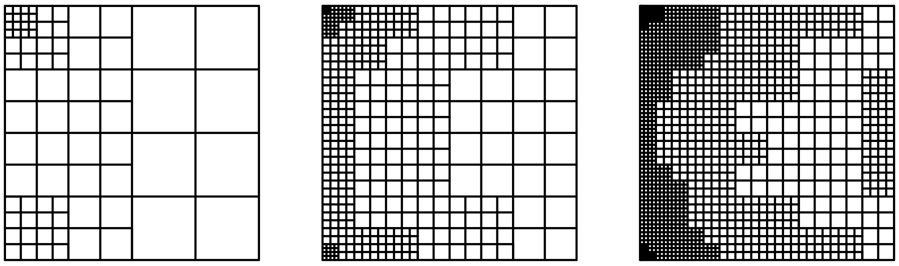
Die Genauigkeit der erzielten Lösung ist in allen Beispielen etwa um den Faktor 5 schlechter als die Abbruchschranke. [5] Der Faktor wächst leicht an bei

[5] Die exakte Lösung der Wärmeleitungsprobleme ist bekannt. Bei dem Scheibenproblem wurden ausgewählte Punkte einer uniformen Verfeinerung mit ca. 32.000 FG und direkter Gleichungslösung als Referenzlösung herangezogen



$\theta = 2.4 \cdot 10^{-4}$ $\theta = 6.1 \cdot 10^{-5}$ $\theta = 1.5 \cdot 10^{-5}$

Abbildung 80: Adaptive Netze für Beispiel HEAT1

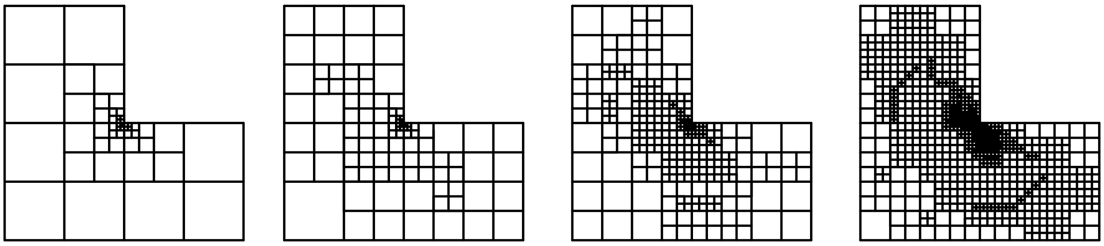


$\theta = 2.4 \cdot 10^{-4}$ $\theta = 6.1 \cdot 10^{-5}$ $\theta = 1.5 \cdot 10^{-5}$

Abbildung 81: Adaptive Netze für die Kragscheibe

zunehmender Verfeinerung. Der Grund liegt in dem logarithmischen Wachstum der Kondition der Systemmatrizen, die den Zusammenhang zwischen der Größe des Residuums und der Genauigkeit des Lösungsvektors beeinflußt.

In den Abbildungen 80, 81 und 82 sind zu den Tabellen gehörigen adaptiven FE-Netze zu sehen. In der graphischen Darstellung der Wärmeleitungsprobleme sind jeweils vier Elemente zu einem Viereck zusammengefaßt worden. Bei allen Beispielen führt die Kombination von rekursiver Substrukturierung und adaptiver Relaxation zu problemangepaßten FE-Diskretisierungen. Im Bereich der Elastizitätsprobleme entsprechen sie qualitativ den FE-Netzen die sich bei Einsatz der Fehlerindikatoren aus Kapitel 2.3.1 ergeben.



$\theta = 9.8 \cdot 10^{-4}$ $\theta = 2.4 \cdot 10^{-4}$ $\theta = 6.1 \cdot 10^{-5}$ $\theta = 1.5 \cdot 10^{-5}$

Abbildung 82: Adaptive Netze für Beispiel HEAT3

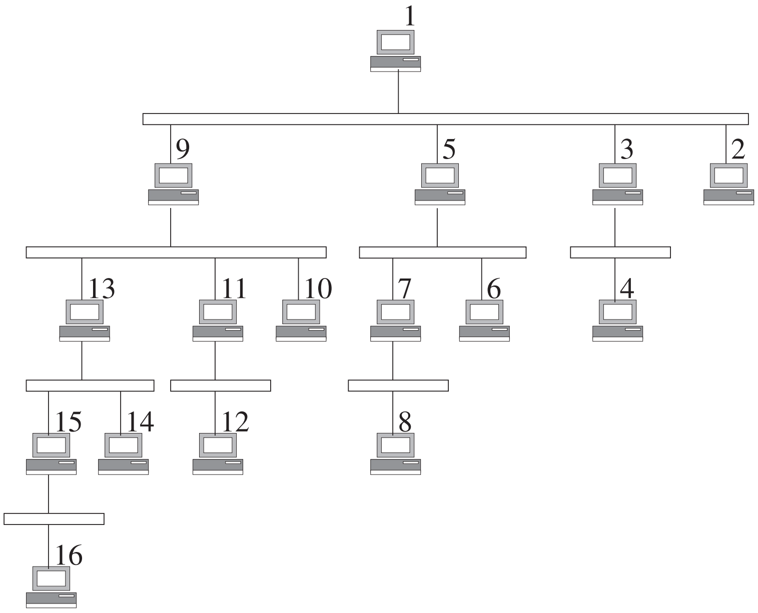


Abbildung 83: Baumartige Ethernet-Topologie mit 8 Bussen und 16 Rechnern

9.3 Parallele Berechnungsläufe

Die meisten parallelen Berechnungsläufe fanden auf 16 lose gekoppelten Workstations vom Typ HP720 statt. Jede dieser Workstation hatte zur Zeit der Ausführung der Tests einen für die Simulation freien Hauptspeicher von 32 MByte. Auf zwei Workstations stand ein erweiterter Speicher vom 100 MByte Hauptspeicher und 100 MByte Swap-Space zur Verfügung. Alle Testläufe fanden zu Zeiten statt, als das Rechnernetz nur wenig ausgelastet war. Die Zeitangaben beziehen sich auf die komplette Lösung eines FE-Problems mit den angegebenen Freiheitsgraden. Darin sind sämtliche Operationen enthalten die zur Verfeinerung des Netzes, Erstellung der Elementgleichungssysteme, Lösung der Systemgleichungen, Auswertung der Ergebnisse etc. anfallen.

Die Rechner sind durch mehrere Ethernet-Busse miteinander verbunden. Durch einen zusätzlichen Ethernet-Anschluß an einigen Workstations stehen hierarchische Topologien für baumartige Algorithmen zur Verfügung (s. [53] und Abb. 83).

Die Verwendung von Workstationnetzen bietet einige Vorteile, die in [38] genauer aufgeführt werden. Workstationnetze sind in der Industrie und an den Universitäten weit verbreitet und bieten deshalb eine vertraute Programmierumgebung zur Entwicklung von parallelen Algorithmen. Ihre Kapazitäten sind meist schwach ausgelastet. Da die Workstation an vorderster Front bei der Soft- und Hardwareentwicklung stehen, sind mit ihnen modernste Prozessoren, billiger Speicher und optimierte Compiler verfügbar.

Es gibt aber auch einige Nachteile, die eine Parallelisierung erschweren ([38]).

Dabei handelt es sich hauptsächlich um die langsame Geschwindigkeit der Kommunikation. Zusätzliche Nachteile ergeben sich durch die Problematik von Ausfällen von Workstations in einem Netzwerk, der meist schlechten Software-Unterstützung einer Parallelisierung und dem Problem, daß meist kein exklusiver Zugriff auf das Netzwerk vorliegt. Diese Nachteile, insbesondere die langsame Kommunikation, machen sich besonders beim iterativen Verfahren stark bemerkbar. Sie führen dazu, daß die theoretischen Ergebnisse in den praktischen Simulationen bei weitem nicht erreicht werden können.

Um den Einfluß der Netztopologie auf die Berechnungszeiten einer numerischen Simulation zu demonstrieren, wird ein Beispiel auf zwei verschiedenen Rechnernetzen gerechnet. Dabei wird eine einfache Bus-Topologie mit einer Binärbaum-Topologie verglichen.

Die Binärbaum-Topologie ist ein hierarchisch strukturiertes Rechnernetz und ermöglicht eine parallele Kommunikation in verschiedenen Teilbäumen, die in Kapitel 8.3 bei der Analyse der Verteilungsstrategien zugrunde gelegt wurde. In einer einfachen Bus-Topologie finden dagegen alle Kommunikationen in einem Verbindungsstrang statt und behindern sich gegenseitig. Mit wachsender Anzahl von Rechnern nehmen die Kollisionen der Datenpakete im Rechnernetz zu und verhindern die Skalierbarkeit der Parallelisierung. Bei der Binärbaum-Topologie ist die Kommunikationszeit abhängig von der Baumtiefe und damit wesentlich besser skalierbar.

Es wurde das Wärmeleitungsproblem HEAT1 mit uniformer Verfeinerung und der Teilungsstrategie der Bisektion auf beiden Topologien gerechnet. Die Abbildung 84 zeigt die Zeiten des Laufes mit 8192 Elementen bei verschiedener Anzahl von Prozessoren. In allen Fällen ist die Baumtopologie schneller. Ab vier Prozessoren werden bei einem Problem dieser Größe die Rechenzeiten auf einer einfache Bus-Topologie wieder langsamer.

Um die Beschleunigung einer Parallelisierung messen zu können, d.h. um ein Maß für die Qualität der Parallelisierung geben zu können, definiert man den Geschwindigkeitsfaktor (speed-up) und die Effizienz (s. [35]).

- speed-up: $s = \frac{Berechnungszeit\ auf\ einer\ Rechnereinheit}{Berechnungszeit\ auf\ n\ Rechnereinheiten}$
- Effizienz: $e = \frac{Speed{-}Up}{Anzahl\ der\ Prozessoren}$

Im Gegensatz zu den theoretischen Überlegungen in Kapitel 8.3 konnte in den praktisch durchgeführten Simulationen keine Beschleunigung gegenüber dem direkten Verfahren erreicht werden. Der Grund dafür liegt in der Implementierung von ARESO. Bei ARESO handelt es sich um einen Prototypen, dessen Ziel es ist, prinzipiell die praktische Umsetzbarkeit der vorgestellten Konzepte nachzuweisen. Es war nicht die Aufgabe der Arbeit, in diesem Prototypen eine Optimierung bzgl. der Berechnungsgeschwindigkeit zu erzielen, da eine derartige Optimierung den Rahmen dieser Arbeit bei weitem überschritten hätte.

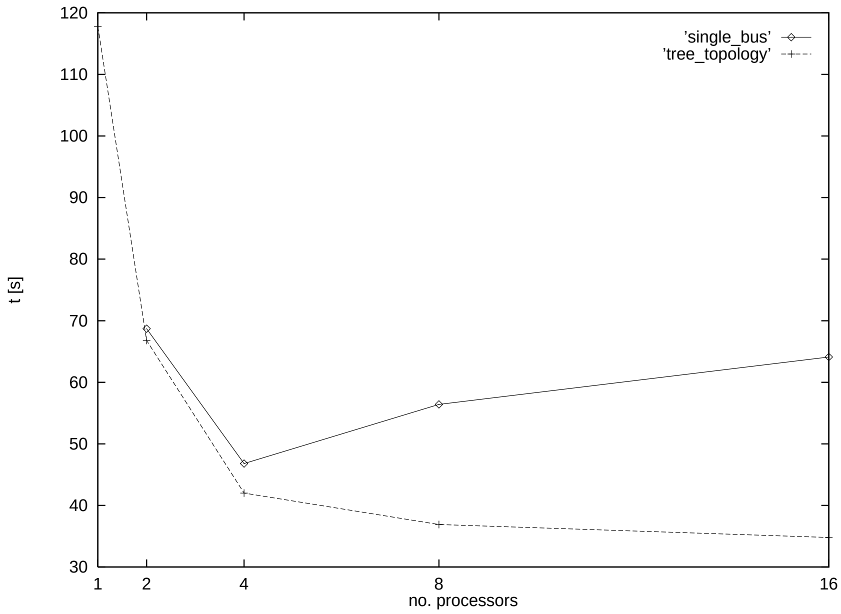


Abbildung 84: Rechenzeiten von uniformen Beispiel mit Bisektion auf verschiedenen Netztopologien

Dieser Nachteil wirkt sich besonders stark bei dem iterativen Verfahren aus. Im direkten Verfahren werden in einem W-Zyklus auf vollbesetzten Matrizen zwar mehr arithmetische Operationen durchgeführt, diese sorgen aber durch optimierte Routinen aus Standard-Bibliotheken für lineare Gleichungssysteme für eine sehr gute Prozessorauslastung. Das iterative Verfahren muß mehrere Baumdurchläufe durchführen, die zwar deutlich weniger arithmetische Operationen erfordern, diese werden aber nicht mit optimierte Routinen ausgeführt. Zudem erfordert jeder Baumdurchlauf in der aktuellen Implementierung einen Verwaltungsaufwand der zum Teil den Aufwand für die arithmetische Operationen übersteigt.

Ein weiterer großer Nachteil für das iterative Verfahren ist die Notwendigkeit, mehr Kommunikationsschritte durchzuführen. Die Datenpakete sind zwar deutlich kleiner, aber die begrenzten Kommunikationsraten von lose gekoppelten Workstations führt hier zu langen Gesamtrechenzeiten.

Um mit dem Prototyp ARESO effiziente parallele numerische Simulationen mit dem iterativen Verfahren durchzuführen, müßte die Programmstruktur vollkommen überarbeitet werden, um den Verwaltungsaufwand auf das notwendige Mindestmaß zu reduzieren. Auch eine Umstellung auf schnellere Softwarepakete zur Unterstützung der rechnerübergreifenden Kommunikation, wie z.B. PVM, ist dringend erforderlich. Da der Schwerpunkt der vorliegenden Arbeit aber die Entwicklung des neuen Lösungsverfahrens sowie der theoretische Nachweis der Vorzüge gegenüber dem direkten Verfahren war, konnten diese Optimierungen im gegebenen zeitlichen Rahmen nicht durchgeführt werden.

In den Abbildungen 85 und 86 sind die Rechenzeiten für das Wärmeleitungs-

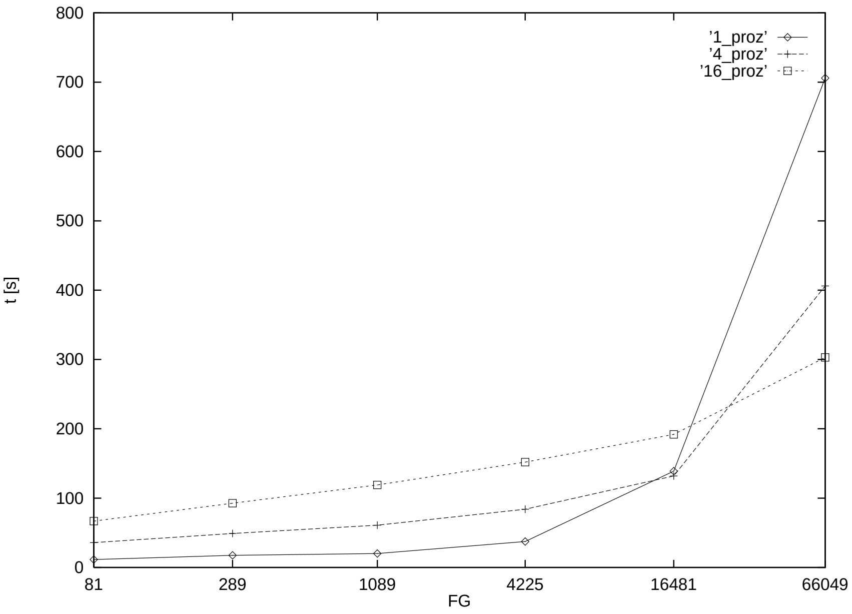


Abbildung 85: Rechenzeiten von Beispiel HEAT2 (uniform) mit iterativen Lösungsalgorithmus für verschiedene Anzahl von Workstations

problem HEAT2 mit uniformer Verfeinerung und der Teilungsstrategie der Viertelung zu sehen. Die Berechnungen wurden auf der Baumtopologie mit bis zu 16 Rechnern, unter Verwendung der Verteilungsstrategie VS3, durchgeführt. Es zeigt sich, daß die sequentielle Berechnungszeit des iterativen Verfahrens für das Wärmeleitungsproblem HEAT2 mit uniformer Verfeinerung genauso schnell ist wie die des direkten Verfahrens ist. Die parallelen Berechnungsläufe werden aber langsamer, so daß der speed-up von 6.7 beim direkten auf 2.3 beim iterativen Verfahren bei 16 Prozessoren sinkt. Auch bei 4 Prozessoren sinkt der speed-up von 2.1 auf 1.7 und die Effizienz von 0.54 auf 0.43.

Um zu demonstrieren, wieviel Zeit ein W-Zyklus des iterativen Algorithmus bei der aktuellen Implementierung von ARESO benötigt, sind in Abbildung 87 die Berechnungszeiten von einem Beispiel mit einem bilinearen Lösungsverlauf auf dem Rand des Berechnungsgebietes zu sehen. In diesem Beispiel wird die korrekte Lösung in einem Iterationsschritt, d.h. einem W-Zyklus erreicht, da die Werte auf dem Rand durch die Hierarchisierung sofort die korrekte Lösung im Gebietsinneren erzeugen. Bei diesem Beispiel sind die Rechenzeiten deutlich geringer wie beim direkten Verfahren. Aber auch hier ist der speed-up mit 2.65 und die Effizienz mit 0.17 deutlich schlechter als beim direkten Algorithmus.

Das iterative Verfahren ist zwar wegen der Implementierung von ARESO nicht schneller als das direkte Verfahren aber es können damit Probleme gelöst werden, die bisher in unseren Testumgebungen nicht berechenbar waren. In Abbildung 88 sind die Rechenzeiten für das Beispiel HEAT1 mit mit wachsender Anzahl von Freiheitsgraden zu sehen. Als Testumgebung diente ein Worksta-

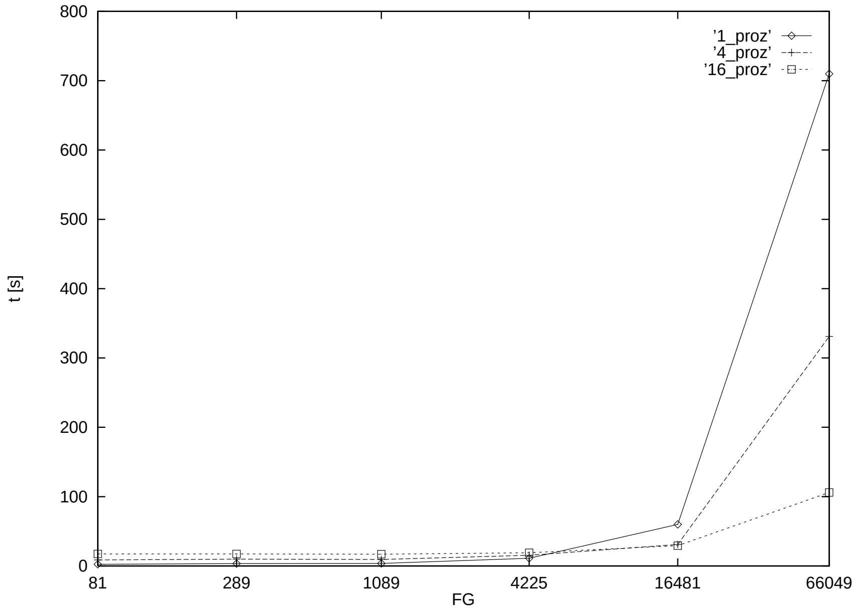


Abbildung 86: Rechenzeiten von Beispiel HEAT2 (uniform) mit direkten Lösungsalgorithmus für verschiedene Anzahl von Workstations

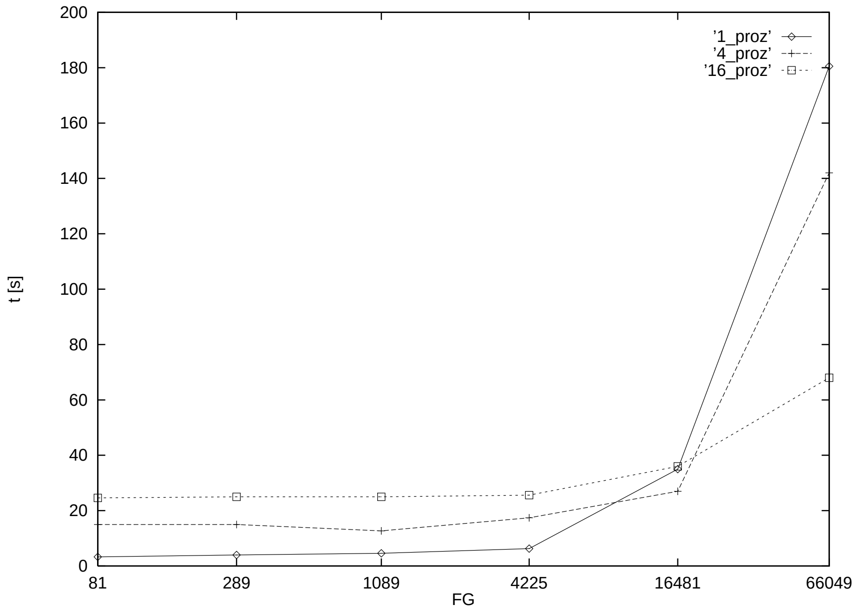


Abbildung 87: Rechenzeiten des iterativen Lösungsalgorithmus bei einem Beispiel mit bilinearen Lösungsverlauf

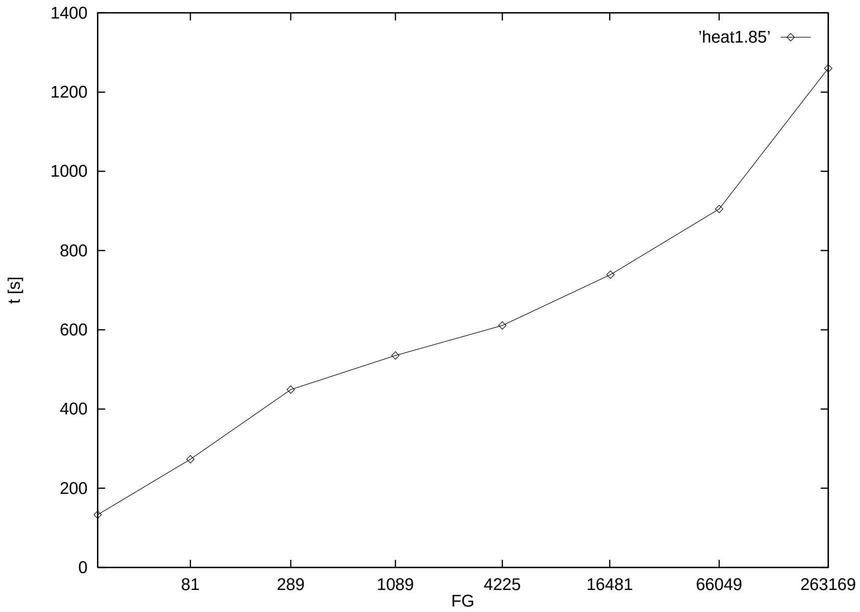


Abbildung 88: Rechenzeiten von Beispiel HEAT1 bei iterativem Lösungsalgorithmus auf 85 Workstations

tioncluster mit 110 HP9720 mit 32 MByte Hauptspeicher. Davon wurden 85 ausgewählt und mit der Verteilungsstrategie VS2 dem Substrukturierungsbaum zugewiesen. Die berechnete Problemgröße entspricht einem uniform verfeinerten Netz mit $m = 9$. Der theoretisch erreichbare Wert von $m = 10$ für 85 Workstation und Verteilungsstrategie VS2 (s. Kap. 8.3) kann wegen dem zusätzlichen Speicher für Datenstrukturen zur Modellierung, Zerlegung, Assemblierung, etc. auf der Testumgebung nicht erreicht werden. Man erkennt in der Abbildung an dem annähernd linearen Anstieg der Berechnungszeiten bei uniform wachsender Problemgröße die Skalierbarkeit des Algorithmus. Der Anstieg liegt sogar unter dem Faktor mit dem die Netze wachsen, da sich bei zunehmender Problemgröße das Verhältnis von Rechenzeit zu Kommunikationszeit verbessert.

Da die praktisch erzielbaren Berechnungszeiten des iterativen Verfahrens keine Beschleunigung bezüglich des direkten Lösers ergeben, wird auf die Darstellung von adaptiven, parallelen Berechnungsläufen verzichtet. Für das direkten Verfahren der rekursiven Substrukturierung konnte gezeigt werden, daß auch bei adaptiven Netzen mit einer geeigneten Zuordnung der Rechner zu dem Substrukturierungsbaum ähnliche Beschleunigungen wie bei uniformen Netzen erzielt werden können ([39]).

10 Zusammenfassung und Ausblick

Im Standard FE-Verfahren herrscht entweder keine oder nur eine einstufige hierarchische Struktur in der Modellierung und Berechnung von technischen Problemen. Das Prinzip der rekursiven Substrukturierung unterstützt dagegen eine baumartige Problembeschreibung und vollzieht den Schritt vom Detail zum Gesamtproblem in mehreren Stufen. Diese hierarchische Vorgehensweise ermöglicht effiziente numerische Algorithmen und eine Parallelisierung der FEM.

Der in der Arbeit entwickelte iterative Lösungsalgorithmus verwendet hierarchische Basisfunktionen und benutzt damit das hierarchische Prinzip auch in der Darstellung der Lösung. In diesem konsequent hierarchisch strukturiertem System erfolgt die Lösung von FE-Problemen mit einem Speicherbedarf von $O(n)$ und einem Aufwand an arithmetischen Operationen von $O(n \cdot log(n)^2)$.

Da der iterative Algorithmus explizit in hierarchischen Basen rechnet, kann er mit der Methode der adaptiven Relaxation kombiniert werden, was eine zusätzliche Einsparung an Operationen bewirkt. Als Ergebnis erhält man ein vollständig adaptives Verfahren, das den algebraischen Fehler und den Diskretisierungsfehler mit den gleichen Mitteln reduziert.

Im Gegensatz zur direkten Methode der rekursiven Substrukturierung, bei der ein fester Prozentsatz vom Aufwand im Bereich der oberen Baumknoten liegt, verlagert sich hier der Hauptaufwand in die unteren Baumebenen. Da diese Baumebenen ein höheres Potential an Parallelität besitzen, steigt damit die Zahl der effizient einzusetzenden Rechner in einer numerischen Simulation.

Mit den hier entwickelten Algorithmen wurden einige einfache technische Gebilde modelliert und auf Workstationnetzen berechnet. Dabei wurde die Anwendbarkeit der Vorgehensweise auf adaptive FE-Berechnungen von Wärmeleitungs- und Elastizitätsproblemen mit bilinearen und biquadratischen Ansatzfunktionen bestätigt.

Eine weitere Verbesserung des Verfahrens scheint möglich. Dabei müßte insbesondere untersucht werden, wie sich die Integration der Idee des Erzeugendensystems von [29] auf die Konvergenzgeschwindigkeit der Gleichungslösung auswirkt. Mit der zu erwartenden konstanten Anzahl von Iterationsschritten zur Lösung der Gleichungssysteme (unabhängig von der Anzahl der Unbekannten) verbessert sich auch die praktische Berechnung von technischen Objekten deutlich. Auch die Ausdehnung der adaptiven Relaxation auf baumartig strukturierte Separatoren könnte zusätzliche Ersparnis am Rechenaufwand bewirken.

In dem iterativen Lösungsalgorithmus wurde ausschließlich das Gauß-Seidel-Verfahren als Glätter eingesetzt. Die Verwendung des Jacobi-Verfahrens würde lediglich die Diagonalelemente der Matrizen zur Korrektur der Lösung verwenden und eine weitere Vereinfachung des Algorithmus bewirken. Es bleibt die Frage, wie sich ihr Einsatz auf die Konvergenz des Gesamtverfahrens auswirkt.

Bisher wurden lediglich lineare FE-Berechnungen durchgeführt. Ein interessantes Anwendungsgebiet stellen aber auch nichtlineare Problemstellungen dar. Durch den lokalen Charakter des vollständig adaptiven Verfahrens können die dort auftretenden lokalen Effekte besonders gut erfaßt werden.

Das mehrstufig strukturierte System mit vielen zum Teil unabhängigen Teilproblemen stellt ein ideales Testbeispiel für die Entwicklung von Lastverteilungsverfahren (z.B. [5]) dar. Besonders bei stark adaptiven Netzen sollten solche Verfahren in Zukunft zum Einsatz kommen.

Mit dem Programm ARESO sind numerische Simulationen wahlweise mit direktem oder iterativem Verfahren möglich. Eine Kombination von beiden Verfahren, z.B. der Einsatz vom direkten Löser in kleineren Teilgebieten und eine iterative Lösung der oberen Baumebenen, ist möglich, müßte aber untersucht werden. Eine ähnliche Möglichkeit ist die Verwendung von noch effizienteren iterativen Verfahren auf Teilgebieten mit regelmäßigen Geometrien, wie z.B. das Verfahren der dünnen Gitter ([72]) und die Zusammensetzung dieser Teilgebiete mit dem Prinzip der rekursiven Substrukturierung.

Literatur

[1] L.M. Adams, R.G. Voigt, *A Methodology For Exploiting Parallelism In The Finite Element Process*, ICASE, NASA Langley Research Center, 1984.

[2] G.S. Almasi, A. Gottlieb, *Highly Parallel Computing*, The Benjamin/Cummings Publishing Company, Inc., Redwood City, California, 1994.

[3] I. Babuska, W. C. Rheinboldt, *Error Estimates For Adaptive Finite Element Computations*, SIAM J. Num. Anal., Vol. 15, pp. 736-754, 1978.

[4] I. Babuska, A. Miller, *A-Posteriori Error Estimates and Adaptive Techniques for Finite Element Method*, Univ. of Maryland, Inst. f. Physic. Sci. and Tech., Tech. Note, BN-968, 1981.

[5] M. Backschat, *Entwicklung eines hierarchisch organisierten Maklersystems zur dynamischen Lastverteilung von Divide&Conquer-artigen Problemen*, Diplomarbeit, TU München, 1995.

[6] R. E. Bank, T. F. Dupont, H. Yserentant, *The Hierarchical Basis Multigrid Method*, Numer. Math., Vol. 52, p 427-458., 1988.

[7] R. Barrett et al, *TEMPLATES for the Solution of Linear Systems: Building Blocks for Iterative Methods*, erhältlich via Internet in http://www.netlib.org/templates, 1994.

[8] P. Bastian, *Parallel adaptive Mehrgitterverfahren*, Naturwissenschaftlich-Mathematische Gesamtfakultät der Rupprecht-Karls-Universität Heidelberg, Dissertation, 1994.

[9] K. J. Bathe, *Finite-Elemente-Methoden*, Springer Verlag, Berlin, 1990.

[10] A. Brandt, *Multigrid Techniques: 1984 Guide with Applications to Fluid Dynamics*, Ges. für Mathematik und Datenverarbeitung, Bonn, GMD-Studien Nr. 85, Darmstadt, 1984.

[11] H. Bungartz, *Higher Order Finite Elements on Sparse Grids*, Institut für Informatik, Technische Universität München, SFB-Bericht Nr. 342/01/95 A, 1995.

[12] D. S. Burnett, *Finite Element Analysis*, Addison Wesley Publishing Company, 1987.

[13] S.W. Cae, K. J. Bathe, *On Automatic Mesh Construction and Mesh Refinement in Finite Element Analysis*, Computer & Structures, Vol. 32, No. 3/4, pp. 911-936, Pergamon Press, 1989.

[14] T. F. Chan, B. F. Smith, *Domain Decomposition and Multigrid Algorithms for Elliptic Problems on Unstructured Meshes*, In D. E. Keyes, J. Xu (eds.): 7th Int. Symp. on Domain Decomposition, SIAM, Philadelphia, pp. 175-188, 1994.

[15] B. Davoudzadeh, *PARCON : Steuerungsprogramm zur dynamischen Lastverteilung auf einem Workstationnetz*, Diplomarbeit, TU München, Institut für Informatik, 1994.

[16] R.H. Dodds, L.A. Lopez, *Substructuring in Linear And Nonlinear Analysis*, Int. J. Num. Meth. Engng., Vol. 15, pp. 583-597, 1980.

[17] M. Dryja, O.B. Widlund, *Multilevel Additive Methods for Elliptic Finite Element Problems*, New York University, Courant Institute, Department of Computer Science, Report 507, 1990.

[18] M. Dryja, B.F. Smith, O.B. Widlund, *Schwarz Analysis of Iterative Substructuring Algorithms for Elliptic Problems in Three Dimensions*, New York University, Courant Institute, Department of Computer Science, Report 638, 1993.

[19] *DUDEN Informatik*, Dudenverlag Verlag, Mannheim, Wien, Zürich, 1988.

[20] R. Ebner, *Neuimplementierung einer funktionalen Sprache zu Parallelisierung numerischer Algorithmen*, Diplomarbeit, TU München, Institut für Informatik, 1994.

[21] O. Egeland, P.O. Araldsen, *SESAM-69 - A General Purpose Finite Element Method Program*, Computer & Structures, Vol. 4, pp. 41-68, Pergamon Press, 1974.

[22] EMRC, (Engineering Mechanics Research Corporation), *NISA II, User's Manual, Handbuch, Troy (Michigan), USA*, 1989.

[23] C. Farhat, F. X. Roux, *Implicit parallel processing in structural mechanics*, Computational Mechanics Advances, Vol. 2, Elsevier Science, Niederlande, 1994.

[24] A. George, *Nested Dissection of a Regular Finite Element Mesh*, SIAM J. Numer. Anal., Vol. 10, No. 2, 1973.

[25] A. George, J.W.H. Liu, *An Automatic Nested Dissection Algorithm For Irregular Finite Element Problems*, SIAM J. Numer. Anal., Vol. 15, No. 5, 1978.

[26] A. George, J.W.H. Liu, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice-Hall Series in Computional Mechanics, 1981.

[27] M. Griebel, *On the combination of the ideas of multilevel solvers using hierarchical bases and the substructuring technique for the finite element method*, Institut für Informatik, Technische Universität München, Bericht I8709, 1987.

[28] M. Griebel, *Zur Lösung von Finite-Differenzen- und Finite-Element-Gleichungen mittels der Hierarchischen Transformations Mehrgitter Methode*, Institut für Informatik, Technische Universität München, SFB-Bericht Nr. 342/4/90 A, 1990.

[29] M. Griebel, *Multilevelmethoden als Iterationsverfahren über Erzeugendensysteme*, Teubner Skripten zur Numerik, Stuttgart, 1994.

[30] M. Griebel, *Domain-oriented multilevel methods*, Contemporary Mathematics, Vol. 180, pp. 223-229, 1994.

[31] M. Griebel, T.Neunhoeffer, *A Domain-Oriented Multilevel Algorithm - Implementation and Parallelization*, Institut für Informatik, Technische Universität München, SFB-Bericht Nr. 342/18/94 A, 1994.

[32] M. Griebel, P.Oswald, *On the abstract theory of additive and multiplicative Schwarz algorithms*, Numer. Math. Vol. 70, pp. 163-180, Springer-Verlag, Berlin, 1995.

[33] G. Haase, U. Langer, A. Meyer, S.V. Nepomnyaschikh, *Hierarchical Extension and Local Multigrid Methods in Domain Decomposition Preconditioners*, Fakultät für Mathematik, Technische Universität Chemnitz-Zwickau, Preprint-Reihe der DFG-Forschergruppe Scientific Parallel Computing, Bericht Nr. SPC 94-8, 1994.

[34] W. Hackbusch, *Iterative Lösung großer schwachbesetzter Gleichungssysteme*, Teubner Studienbücher Mathematik, Stuttgart, 1993.

[35] H. U. Heiss, *Prozessorzuteilung in Parallelrechnern*, Reihe Informatik, Band 98, BI Wissenschaftsverlag, Mannheim, 1994.

[36] G. L. Hennigan, S. Castillo, E. Hensel, *Using Domain Decomposition To Solve Symmetric, Positive-Definite Systems On The Hypercube Computer*, Int. J. Num. Meth. Engng. Vol. 33, pp. 1941-1954, 1992.

[37] M.R. Hestens, E. Stiefel, *Methods of conjugate gradients for solving linear systems*, J. Res. Nat. Bur. Standards 49, 1952.

[38] W. Huber, R. Hüttl, M. Schneider, Chr. Zenger, *Distributed Numerical Simulation On Workstation Networks*, in Proceedings Of The Symposium On HPSC High Performance Scientific Computing, Munich, June 17-18 1993, Notes On Numerical Fluid Mechanics, Teubner, 1994.

[39] R. Hüttl, M. Schneider, *Parallel Adaptive Numerical Simulation* , Institut für Informatik, Technische Universität München, SFB-Bericht Nr. 342/01/94 A, 1994.

[40] Ikoss GmbH, *Description of ASKA (Saab-Scania/Ikoss): Features and Analysis-Capabilities*, Produktbeschreibung, Stuttgart, 1989.

[41] S. L. Johnsson, K. K. Mathur, *Data Structures And Algorithms For The Finite Element Method On A Data Parallel Supercomputer*, Int. J. Num. Meth. Engng., Vol. 29, pp. 881-908, 1990.

[42] K. Knothe, H. Wessels, *Finite Elemente*, Springer Verlag, Berlin, 1991.

[43] R. Krause, *Erweiterung eines Pre-und Postprozessors für die Substrukturierung in der Finite-Element-Methode*, Diplomarbeit, TU München, Institut für Informatik, 1994.

[44] U. Langer, *Substrukturtechnik und Schwarzsche Methoden*, GAMM-Miteilungen, Heft 1, pp. 86-103, 1992.

[45] H. X. Lin, *Analysis and Implementation of DD Methods for Parallel FE Computations*, In D. E. Keyes, J. Xu (eds.): 7th Int. Symp. on Domain Decomposition, SIAM, Philadelphia, pp. 407-412, 1994.

[46] MSC/NASTRAN, *Handbook for Superelement Analysis*, The MacNeal-Schwendler Corporation, U.S.A., 1982.

[47] B. Mühlhaus, *Parallele, rekursive Substrukturierung bei der 'finite Element Methode' als Multiuser-System auf Rechnernetzen*, Diplomarbeit, TU München, Institut für Informatik, 1992.

[48] C. Müller, *FEMANI: Ein Postprozessor zu ARESO bei der Substrukturierung in der Finite-Element-Methode*, Fortgeschrittenenpraktikum, TU München, 1993.

[49] C. Müller, *Visualisierung der Bottom-Up-Konstruktion technischer Systeme gemäß dem Baukastenprinzip*, Diplomarbeit, TU München, 1994.

[50] S. Nölting, *Nonlinear Adaptive Finite Element Systems on Distributed Memory Computers*, Distributed Memory Computing, Proceedings of the 2nd European Conference EDMCC2, Munich, Lecture Notes in Computer Science, Arndt Bode (Ed.), vol. 487, Springer Verlag, Berlin, 1991.

[51] A.K. Noor, H.A. Kamel, R.E. Fulton, *Substructuring Techniques - Status And Projections*, Computers & Structures Vol. 8 pp. 621-632, 1977.

[52] J. T. ODEN, A. PATRA, Y. FENG, Domain Decomposition for Adaptive hp Finite Element Methods, In D. E. Keyes, J. Xu (eds.): 7th Int. Symp. on Domain Decomposition, SIAM, Philadelphia, pp. 295-301, 1994.

[53] A. PFAFFINGER, *Parallel Communications on Workstation Networks with Complex Topologies*, Institut für Informatik, Technische Universität München, SFB-Bericht Nr. 342/08/95 A, 1995.

[54] C. PFLAUM, U. RÜDE, *Gauß' Adaptive Relaxation for the Multilevel Solution of Partial Differential Equations on Sparse Grids*, Institut für Informatik, Technische Universität München, SFB-Bericht Nr. 342/13/93 A, 1993.

[55] L. PLANK, *Netzadaption und Mehrgitterverfahren für die numerische Behandlung von Faltwerken*, Fachbereich Bauingenieur- und Vermessungswesen der Universität Hannover, Dissertation, 1990.

[56] J. S. PRZEMIENIECKI, *Matrix Structural Analysis of Substructures*, AIAA Journal, Vol. 1, No. 1, pp. 138-147, 1963.

[57] E. RANK, *A-posteriori-Fehlerabschätzungen und adaptive Netzverfeinerung für Finite-Element- und Randintegralmethoden*, Mitteilungen aus dem Institut für Bauingenieurwesen I, TU München, 1985.

[58] W. REDANZ U.A., *ISARES, Integrated Structure Analysis Research System*, Lehrstuhl für Statik der TU München, 1995.

[59] U. RÜDE, *On the Multilevel Adaptive Iterative Method*, Institut für Informatik, Technische Universität München, Bericht Nr. TUM-INFO-05-92-I16-300/1.-FMI, 1992.

[60] U. RÜDE, *On the Robustness and Efficiency of the Fully Adaptive Multigrid Method*, In A. Quarteroni, J. Periaux, Y.A. Kuznetsov, O.B. Widlund (eds.): 6th Int. Symp. on Domain Decomposition, SIAM, Philadelphia, pp. 121-126, 1992.

[61] U. RÜDE, *Mathematical and Computional Techniques for Multilevel Adaptive Methods*, Frontiers in Applied Mathematics, vol. 13, SIAM. Philadelphia, 1993.

[62] W. RUST, *Mehrgitterverfahren und Netzadaption für lineare und nichtlineare statische Finite-Elemente-Berechnungen von Flächentragwerken*, Fachbereich Bauingenieur- und Vermessungswesen der Universität Hannover, Dissertation, 1991.

[63] M. SCHNEIDER, *Verteilte adaptive numerische Simulation auf der Basis der Finite-Elemente-Methode*, Technische Universität München, Dissertation, 1995.

[64] H. R. SCHWARZ, *Methode der Finiten Elemente*, Teubner Studienbücher Mathematik, Stuttgart, 1991.

[65] J. R. SHEWCHUK, O. GHATTAS, *A Compiler for Parallel Finite Element Methods with Domain-Decomposed Unstructured Meshes*, In D. E. Keyes, J. Xu (eds.): 7th Int. Symp. on Domain Decomposition, SIAM, Philadelphia, pp. 445-450, 1994.

[66] R. SOUTHWELL, *Relaxation Methods in Engineering Science - a Treatise in Approximate Computation*, Oxford University Press, 1940.

[67] J. STOER, *Numerische Mathematik 1*, Springer Verlag, Berlin, 1989.

[68] J. STOER, *Numerische Mathematik 2*, Springer Verlag, Berlin, 1990.

[69] B. WAHLER, *Implementierung einer graphischen Oberfläche in Motif zur Substrukturierung in der Finite-Element-Methode*, Diplomarbeit, TU München, 1992.

[70] W. WUNDERLICH, G. BREITZKE, K. LEIMBACH, *Substar, Ein Finite-Element-Programm mit Mehrebenen-Substrukturtechnik zur linearen statischen Berechnung von Tragwerken*, Ruhr-Universität Bochum, Institut für konstruktiven Ingenieurbau, Technisch-wissenschaftliche Mitteilungen, 1986.

[71] H. YSERENTANT, *On the Multi-Level Splitting of Finite Element Spaces*, Numer. Math., Vol. 49, p 379-412., 1986.

[72] C. ZENGER, *Sparse Grids*, Institut für Informatik, Technische Universität München, SFB-Bericht 342/18/90 A, 1990.

[73] C. ZENGER, *Parallele Berechnung von Ingenieurproblemen auf baumartigen Netzwerken von Arbeitsplatzrechnern*, Tagungsband des 124. WE-Hereaus-Seminar, Parallele Algorithmen in der Festkörpermechanik, Bad Honnef, 1994.

[74] O. C. ZIENKIEWICZ, J. Z. ZHU, *A simple error estimator and adaptive procedure for practical engineer analysis*, Int. J. Num. Meth. Engng., Vol. 24, p 337-357., 1987.

[75] O. C. ZIENKIEWICZ, J. Z. ZHU *Superconvergence recovery technique and a posteriori error estimators*, Int. J. Num. Meth. Engng., Vol. 30, pp. 1323-1339, 1990.

[76] O. C. ZIENKIEWICZ, J. Z. ZHU *The three R's of engineering analysis and error estimation and adaptivity*, Comp. Meth. in Appl. Mech. and Engng., pp. 95-113, 1990.

[77] O. C. Zienkiewicz, J. Z. Zhu *Adaptivity And Mesh Generation*, Int. J. Num. Meth. Engng., Vol. 32, pp. 783-810, 1991.

[78] O. Zone, R.Keunings, D. Roose, *Parallel Algorithms for the Direct Solution of Finit Element Equations on a Distributed Memory Computer*, Distributed Memory Computing, Proceedings of the 2nd European Conference EDMCC2, Munich, Lecture Notes in Computer Science, Arndt Bode (Ed.), vol. 487, Springer Verlag, Berlin, 1991.

Index