5 Software–Aspekte und Implementierung

Für die diskrete Modellierung von Strömungsprozessen in geklüftet–porösen Medien sind eine Reihe von Teilschritten notwendig. In dem vorliegenden Kapitel wird eine Beschreibung der einzelnen Bausteine des softwaretechnischen Modellierungsprozesses geliefert und auf einige implementierungsrelevante Aspekte des entwickelten Verfahrens eingegangen. Für einige dieser Teilschritte wurden bestehende Module wie z.B. der Kluftgenerator FRAC3D genutzt, für andere wurden vorhandene Software–Pakete an die speziellen Anforderungen des in Kapitel 3 vorgestellten hierarchischen Gebietszerlegungsverfahrens angepaßt und erweitert.

5.1 Kluftgenerierung

Wir haben in den Abschnitten 1.1 und 1.2 festgestellt, daß die tatsächlichen Gegebenheiten im Untergrund nur eingeschränkt bekannt sind. Die Erstellung eines Kluftgeometriemodells auf der Basis von durch Messungen, Datenerhebungen und –auswertungen gewonnenen Daten ist dann sinnvoll, wenn die Qualität und Quantität der gewonnenen Daten ausreicht (siehe dazu Abschnitt 1.2 und Silberhorn–Hemminger [57]), um aus den zur Verfügung stehenden Daten ein repräsentatives Kluftnetzwerk zu rekonstruieren.
Der tatsächlichen Kluftgenerierung gehen Messungen voraus. Geometrische Strukturen werden, soweit sie an Aufschlußwänden, offenen Stollen und Bohrkernen direkt zugänglich sind, durch Ort und Orientierung der Klüfte im Raum, Abmessungen der Kluftseitenflächen, Abstände von Klüften, Kluftdichte und Kluftöffnungsweiten beschrieben. Materialparameter wie Porosität und Permeabilität werden in Labor– und Feldversuchen ermittelt. Zusätzlich erhält man mit geophysikalischen Methoden (Seismik, Geomagnetik, Geoelektrik) indirekte Daten, die im Gegensatz zu den direkt gewonnen Daten unscharf sind. Unter der Kluftgenerierung wird dann die Erstellung eines Kluftgeometriemodells auf der Grundlage der zur Verfügung stehenden Daten verstanden. Stehen ausreichend Daten zur Verfügung ist gegebenenfalls unter Einsatz geeigneter Methoden der Geoinformatik (Geostatistik, Stochastik) eine Kluftgenerierung möglich. Das so entstehende Kluftgeometriemodell wird im weiteren Verlauf als Kluftnetzwerk bezeichnet.
Für die Kluftnetzwerkgenerierung wird in dieser Arbeit der Kluftgenerator FRAC3D (siehe Silberhorn–Hemminger [57]) eingesetzt. Dieser Kluftgenerator erlaubt die Erstellung von Kluftnetzwerken auf der Grundlage des deterministischen oder eines stochastischen Ansatzes, je nach Art und Menge der vorhandenen Informationen.
Der Kluftgenerator FRAC3D erzeugt zunächst die Gesamtheit der Kluftelemente. Die Klüfte werden von FRAC3D im hier betrachteten 2D Fall als 1D Stabelemente beschrieben. Nach und nach werden einzelne Kluftelemente dem System zugefügt, bis eine vorgegebene Kluftdichte erreicht ist. Dabei muß jedes Kluftelement gewissen Vorgaben genügen. So würden z.B. zu kleine Abstände zwischen parallelen Klüften zu Problemen bei der anschließenden Netzgenerierung führen.

Weicht das Verteilungsmuster des generierten Kluftsystems zu sehr von der Eingangsverteilung ab, so kann das System verworfen oder einer Optimierung unterzogen werden.
Nach Fertigstellung des Kluftnetzwerkes werden die Informationen bezüglich gemeinsamer Schnittlinien und Schnittpunkten von Klüften durch ein Postprocessing als zusätzliche Daten für die Netzgenerierung in die Gebietsbeschreibung übernommen. Das generierte Gebiet ist dabei im allgemeinen größer als das tatsächliche Untersuchungsgebiet, das nun aus dem generierten Gebiet ausgeschnitten wird. So werden Generierungsfehler am Gebietsrand verhindert bzw. minimiert. Der Kluftgenerator legt abschließend die Daten, die das Kluftnetzwerk beschreiben, in einem für den Netzgenerator (siehe Abschnitt 5.2) lesbaren Format ab. Der Netzgenerator ist das Modul der Kluftsimulation, das das Kluftnetzwerk mit einem Gitter versieht.
Für eine ausführlichere Beschreibung des Kluftgenerators FRAC3D verweisen wir auf Silberhorn–Hemmiger [57] und Neunhäuserer [48].

5.2 Netzgenerierung

Ein Netzgenerator dient der Erstellung des Grobgitters, das im folgenden auch als Netz bezeichnet wird. Im Rahmen der diskreten Beschreibung eines geklüftet–porösen Mediums hat ein Netzgenerator die Aufgabe, Matrix und Klüfte, die von Hand oder mit einem Kluftgenerator (siehe Abschnitt 5.1) erzeugt wurden, zu vernetzen und so einer anschließenden numerischen Bearbeitung zugänglich zu machen. Die geometrische und physikalische Komplexität von Kluft–Matrix–Systemen stellt dabei hohe Ansprüche an den Netzgenerator. Die geometrische Komplexität äußert sich zum einen darin, daß die Matrix von den Klüften beliebig in einzelne Blöcke zerschnitten wird. Zum anderen sollen die Klüfte konsistent mit den angrenzenden Matrixblöcken vernetzt werden.
Es existieren eine Reihe verschiedener Ansätze (siehe z.B. Fuchs [30], Schneiders [55]) zur Netzgenerierung. Für Kluft–Matrix–Systeme werden aufgrund der komplexen Geometrie im allgemeinen Methoden verwendet, die eine Delaunay–Triangulierung beinhalten. Es gibt speziell für Kluft–Matrix–Systeme entwickelte Netzgeneratoren, da für allgemeinere Fragestellungen konzipierte Netzgeneratoren die gestellten Anforderungen nur bedingt oder gar nicht erfüllen.
Für den Fall einer niederdimensionalen Modellierung ist dabei eine Kombination von Elementen unterschiedlicher Dimension (z.B. 2D–Matrix, 1D–Kluft) notwendig, im hier betrachteten äquidimensionalen Fall hingegen müssen die Klüfte trotz ihrer ungünstigen Geometrie (Kluftausdehnung $\gg$ Kluftbreite) über die Kluftbreite vernetzt werden (siehe Abschnitte 3.1 und 3.2.1). Die physikalische Komplexität beruht auf den unterschiedlichen Eigenschaften von Kluft und Matrix hinsichtlich Durchströmung und Speicherkapazität. Diese können zu hohen Gradienten und starken Konzentrationsfronten im Bereich der Klüfte führen, so daß dort gegebenenfalls eine Verfeinerung des Berechnungsgitters erforderlich ist.

5.2.1 Netzgenerator ART

Wir haben in dieser Arbeit den vom Mathematischen Institut der Universität Stuttgart im Rahmen des SFB 404 “Mehrfeldprobleme in der Kontinuumsmechanik” entwickelten

Netzgenerator ART (Almost Regular Triangulation) verwendet. In Zusammenarbeit des Instituts für Bauingenieurwesen und des Mathematischen Instituts der Universität Stuttgart wurde ART an die besonderen Anforderungen geklüfteter Systeme angepaßt (Fuchs [29], Neunhäuserer, Fuchs, Hemminger und Helmig [49]). Auf der Grundlage einer optimierten Delaunay Triangulierung wird die Konstruktion eines möglichst regulären Netzes angestrebt. Eine ideale Triangulierung besteht in 2D aus gleichseitigen Dreiecken und jeder innere Knoten gehört zu genau sechs Dreiecken. Eine solche Triangulierung kann nur in Ausnahmefällen erreicht werden, daher versucht ART eine fast reguläre Triangulierung zu erzeugen, die eine möglichst geringe Anzahl irregulärer Knoten, also innere Knoten mit weniger oder mehr als sechs anliegenden Dreiecken, enthält.
Die Basis–Konfiguration der Knoten wird bestimmt, indem ein reguläres Gitter um den ungefähren Mittelpunkt des Gebietes konstruiert wird. Im 2D–Fall ist dies ein aus sechs gleichseitigen Dreiecken bestehendes Sechseck. Die Größe der Elemente in der Matrix wird über eine vorzugebende Dichtefunktion gesteuert, sie kann konstant gewählt werden oder z.B. eine feinere Vernetzung in der Umgebung der Klüfte fordern (siehe Abb. 5.1). Anschließend werden diejenigen Elemente des Startgitters, deren Kanten bezüglich der zugrundeliegenden Dichtefunktion zu groß sind, solange verfeinert bis sie klein genug sind. Die Klüfte werden von ART zunächst niederdimensional vernetzt, dazu werden sie als innere Ränder betrachtet. Die Randpunkte des Gebietes erhält man, indem die außerhalb des Gebietes liegenden Punkte auf den Rand projiziert werden. Weitere Knoten kommen hinzu, wenn die inneren Ränder wie z.B. Klüfte und Flächen mit den Kanten des Startgitters geschnitten werden, und die Schnittpunkte zu den inneren Knoten hinzugenommen werden. Liegen Punkte danach zu nah beieinander, wird einer von ihnen wieder gelöscht. Die Anzahl der zu optimierenden Knoten besteht dann aus den Randpunkten, den Eckpunkten des Gebietes und allen Knoten, die innerhalb des Gebietes liegen. Die Position der Knoten zueinander wird durch die Minimierung eines Penalty–Funktionals optimiert bevor die Verbindungskanten gezogen werden. Die Optimierung der Knotenpositionen zueinander hat das Ziel, die oben beschriebene Regularität der Elemente so weit wie möglich einzuhalten. Bei der abschließenden Triangulierung wird die Konformität der Elemente mit den Randkurven und –flächen sichergestellt.
Die so erzeugten Netze zeigen bei Vorgabe beliebiger innerer Ränder (Klüfte) hohe kombinatorische und geometrische Qualität (Neunhäuserer et al. [52]). Eine ausführliche Beschreibung des Netzgenerators ART findet sich in Fuchs [30].

So entsteht eine verhältnismäßig regelmäßige Netzstruktur mit qualitativ hochwertigen Elementgeometrien. Da die Qualität der Triangulierung ausschlaggebend für den Diskretisierungsfehler ist, also einen entscheidenden Faktor bei der numerischen Simulation darstellt (siehe Abschnitt 3.2), gibt ART zusätzlich Informationen über die Gesamtzahl von Knoten, die Anzahl der inneren, der äußeren sowie der irregulären Knoten, den minimalen, den maximalen und den mittleren Winkel, die Standardabweichung der Winkel von 60^o, die Standardabweichung der Kantenlängen von der mittleren Kantenlänge und die Standardabweichung des Verhältnisses Umkreis zu Inkreis von dem Wert eines gleichseitiges Dreiecks aus.
Der Netzgenerator ART ermöglicht die Vernetzung sowohl zweidimensionaler als auch dreidimensionaler Gebiete, er könnte also auch bei einer Erweiterung der vorliegenden

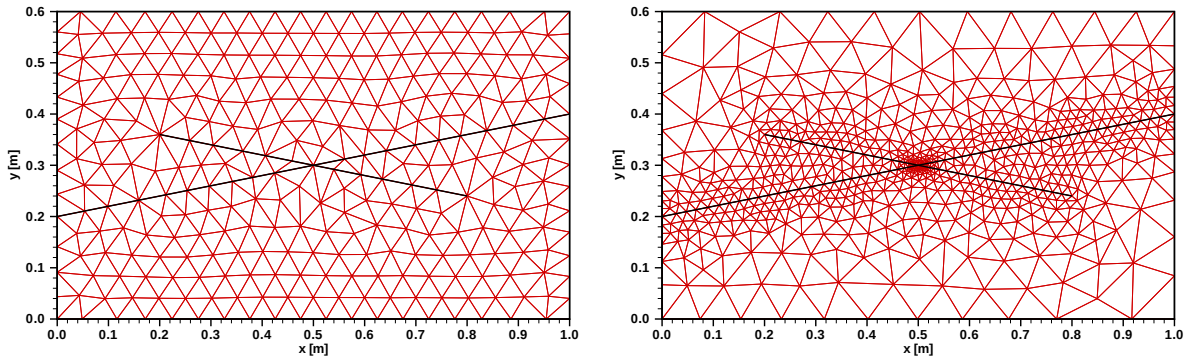

Abbildung 5.1: Vernetzung mit ART. Links: Vernetzung mit konstanter Dichtefunktion. Rechts: feinere Vernetzung entlang der Klüfte (nach Neunhäuserer [48])

Methoden auf den 3D–Fall weiterhin benutzt werden. ART vernetzt allerdings die vom Kluftgenerator erstellten Kluftnetzwerke mit niederdimensionalen Klüften. Für die in der vorliegenden Arbeit betrachteten äquidimensionalen Klüfte muß daher noch ein Zwischenschritt durchgeführt werden, der aus den niederdimensionalen Klüften äquidimensionale Klüfte macht. Das geschieht mit dem Erweiterungsmodul FRACMESH im folgenden Abschnitt 5.2.2.

5.2.2 Erweiterungsmodul FRACMESH

Für das gemeinsame DFG–Projekt des Instituts für Mathematik II der Freien Universität Berlin und der Fakultät für Bauingenieurwesen der Universität Stuttgart (siehe DFG–Arbeitsbericht [31]) wurde in Stuttgart das Erweiterungsmodul FRACMESH entwickelt. FRACMESH liest die Gebietsbeschreibung und die von dem Netzgenerator ART erzeugten Gitter ein. Als zusätzliche Information benötigt FRACMESH Angaben über die Kluftöffnungsweite. Dann werden die von FRAC3D generierten und von ART vernetzten Klüfte in Kluftrichtung aufgeschnitten, die Kluftknoten verdoppelt und bis auf die Kluftbreite auseinandergeschoben. Anschließend werden die Klüfte mit Vierecken über die gesamte Kluftbreite vernetzt (siehe Abb. 5.2). Auf diese Weise werden bei der äquidimensionalen Diskretisierung der Klüfte neue Knoten und Kanten erzeugt werden, die gegebenenfalls in die angrenzenden Matrixkanten integriert werden müssen, damit die Konsistenz des Netzes bei diesem Vorgang nicht zerstört wird. Dabei bilden die Kluftknoten den entscheidenden Faktor. In ihnen werden alle notwendigen Informationen über den angrenzenden Netzbereich gespeichert: zu wie vielen und welchen Kanten der Knoten gehört, welche Kanten davon Kluft–, Matrix oder Randkanten sind und in welcher Reihenfolge und unter welchem Winkel entgegen dem Uhrzeigersinn sortiert die Kanten anliegen und welches der neu erzeugte Knotennachbar auf der gegenüberliegenden Seite der äquidimensionalen Kluft ist.

Kluftkreuzungen werden gesondert betrachtet. FRACMESH erlaubt eine flexible Steuerung der Vernetzung von Kluftkreuzungen. Dadurch kann die Elementstruktur in der Kreuzung der Aufgabenstellung angepaßt werden und der Kreuzung gegebenenfalls ein eigener Materialkennwert (Permeabilität $\underline{\underline{K}}_0$) zugewiesen werden.

Gerade bei Klüften, die sich unter einem flachen Winkel schneiden, kann es beim Aufschneiden der Klüfte zu einer weiteren Verschlechterung der Elementgeometrien im Kreuzungsbereich

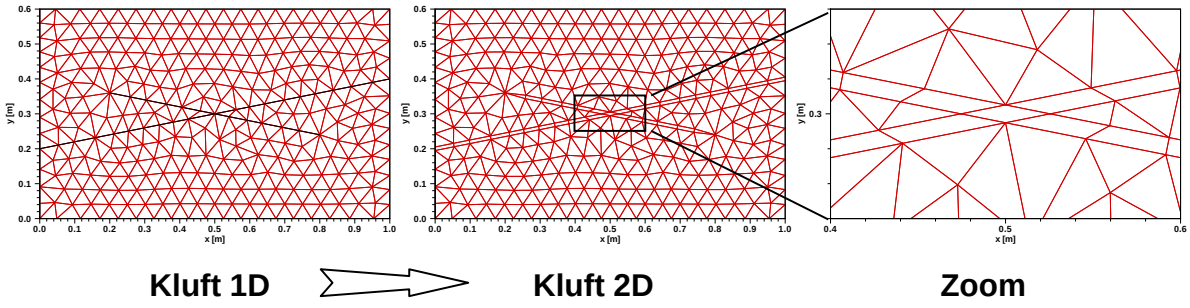

Abbildung 5.2: Vernetzung der Klüfte. Links: niederdimensional mit 1D–Elementen. Mitte, Rechts: äquidimensional mit Viereckselementen (nach Neunhäuserer [48])

kommen. Dieser Effekt kann durch eine anschließende Glättung verringert werden (siehe Neunhäuserer [48]).
Das von ART und FRACMESH erzeugte Netz bildet das Basisgitter für die verwendeten Mehrgitterverfahren (Abschnitte 3.3.1 und 3.3.3). Dafür werden von FRACMESH zwei Datenformate erstellt. Das eine wird bei der Berechnung der physikalischen Prozesse mit einem klassischen Mehrgitterverfahren verwendet. Das Gebiet wird als Ganzes betrachtet, die Zuordnung der Elemente zu Kluft oder Matrix erfolgt allein über die definierten Materialeigenschaften.
Das andere liefert zusätzliche Informationen für das Programmsystem MUFTE–UG (siehe Abschnitt 5.3), auf der die neu implementierte Datenstruktur für die hierarchische Gebietszerlegung (Abschnitt 3.4.1 und Abschnitt 5.3.3) aufbaut. Sie erlaubt eine Unterteilung des Problems in Kluft–, Interface– und Matrixproblem im Sinne einer Teilraumzerlegung (siehe Abschnitt 3.4.1).

5.3 Programmsystem MUFTE–UG

Für die Implementierung der in dieser Arbeit vorgeschlagenen Verfahren zur Lösung von Strömungsproblemen in geklüftet porösen Medien wurden die Programmsysteme MUFTE und UG in der Verbindung zu dem Softwarepaket MUFTE–UG verwendet (siehe Abbildung 5.3). UG (Unstructured Grids) ist eine Software Bibliothek zur Entwicklung von Anwendungen zum Lösen partieller Differentialgleichungen. UG wurde von der Technical Simulation Group am IWR (Interdisziplinäres Zentrum für Wissenschaftliches Rechnen) der Universität Heidelberg entwickelt (Bastian et al. (1997)[11]). Das Softwarepaket MUFTE (Multiphase Flow, Transport and Energy Model) enthält eine Reihe von mathematischen Modellen, Diskretisierungsmethoden und Anwendungsbeispiele für die Modellierung von Mehrphasen–Mehrkomponenten–Systemen der Hydrosystemmodellierung und Hydromechanik. MUFTE wurde und wird am Institut für Wasserbau, Lehrstuhl für Hydromechanik und Hydrosystemmodellierung, der Universität Stuttgart (Helmig et al. (1998)[36], Breiting et al. (2000)[17]) entwickelt.

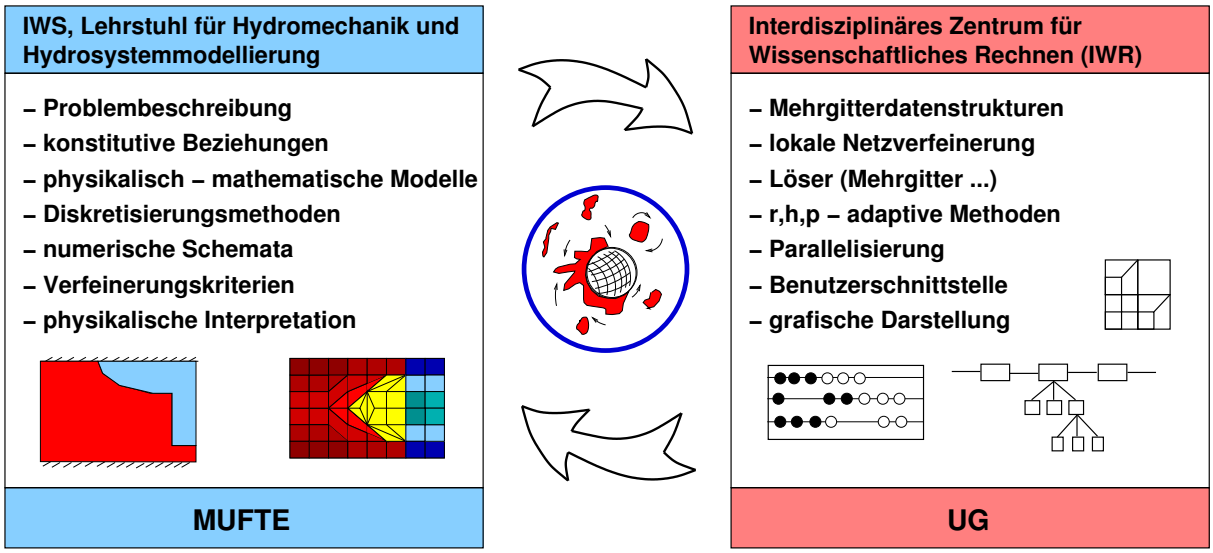

Abbildung 5.3: Struktur des Programmsystems MUFTE–UG, nach Neunhäuserer [48]

5.3.1 Softwarepaket MUFTE

MUFTE umfaßt eine Reihe von *Problem Class Libraries* und *Applications*, in denen physikalisch–mathematische Modelle, konstitutive Beziehungen, Diskretisierungsmethoden und Anwendungsbeispiele für die Modellierung von Mehrphasen–Mehrkomponenten–Systemen implementiert sind. Es stehen Module für unterschiedliche Probleme aus der Hydrosystem–Modellierung zur Verfügung, z.B. :

- isotherme Einphasen–Multikomponenten-Prozesse
  (Hinkelmann et al. (2000)[39], Hinkelmann et al. (2000)[40])
- isotherme Einphasen–Zweikomponenten–Prozesse in geklüftet–porösen Medien
  (niederdimensional) (Neunhäuserer et al. (1999)[51], (äquidimensional) Neunhäuserer, Gebauer et al. [50])
- isotherme Zweiphasen– und Dreiphasen–Prozesse ohne Phasenübergang
  (Croisé et al. (1995)[22], Croisé et al. (1995)[23])
- isotherme Dreiphasen–Dreikomponenten–Prozesse mit Phasenübergang
  (Huber et al. (1997)[43], Huber (2000)[42])
- nichtisotherme Dreiphasen–Dreikomponenten–Prozesse mit Phasenübergang
  (Emmert (1996)[28], Class (2001)[20])
- isotherme Zweiphasen–Zweikomponenten–Prozesse in geklüftet–porösen Medien
  (niederdimensional) (Bastian et al. (2000)[12], Helmig et al. (2001)[37])

Im Rahmen des DFG–Projektes (siehe DFG–Arbeitsbericht [31]) wurde die Problem Class Library für Einphasen–Zweikomponenten–Prozesse in geklüftet porösen Medien von Neunhäuserer (siehe Neunhäuserer [48] und DFG–Arbeitsbericht [31]) so modifiziert, daß

sie sowohl nieder– als auch äquidimensionale Modellierung der Klüfte erlaubt und neben klassischen Mehrgitterverfahren auch eine hierarchische Zerlegung in Kluft und Matrix ermöglicht.

5.3.2 Software–Toolbox UG

UG ist ein umfangreiches Softwaresystem, das eine Reihe effizienter Techniken, wie Mehrgitterverfahren, unstrukturierte Gitter und Adaptivität für die numerische Lösung partieller Differentialgleichungen bereitstellt. Ursprünglich wurde UG entwickelt, um die Entwicklung von Mehrgitterverfahren auf unstrukturierten Gittern und ihre Parallelisierung für praktische Anwendungen zu vereinfachen. Es stellt also die notwendigen Strukturen zur Verwaltung von Gitterhierarchien zur Verfügung. Zusätzlich werden eine Reihe von Gebietsbeschreibungen, Diskretisierungen, Lösungsverfahren und Fehlerschätzern bereitgestellt. Spezielles Gewicht wird dabei auf adaptive Netzverfeinerung, robuste Mehrgittermethoden und Parallelisierungstechniken für unstrukturierte Gitter gelegt.
Im wesentlichen sind die für diese Arbeit relevanten Teile von UG wie folgt gegliedert:

Applications: in diesem Teil von UG befinden sich die Gebietsbeschreibungen und Randbedingungen, eigene Gebietbeschreibungen können hier hinzugefügt werden. Zusätzlich finden sich in diesem Teil von UG auch die Parameterfunktionen.

Problem Class Library: enthält die Diskretisierung, zur Verfügung stehen eine Reihe verschiedener Finiter Elemente. Einige spezielle Lösungsverfahren sind ebenfalls hier zu finden. An dieser Stelle können weitere Lösungsverfahren implementiert werden, sofern UG die benötigten Datenstrukturen zur Verfügung stellt. Für die adaptive Gitterverfeinerung kann auf eine Reihe bereits implementierter Fehlerschätzer zurückgegriffen werden.

UG–Library: Dieser Teil von UG ist der Kern des Pakets. Er enthält die geometrischen und algebraischen Datenstrukturen und eine Anzahl an Gitterverfeinerungs– bzw. Vergröberungstechniken für die Verwaltung der unstrukturierten Gitter:

User–Interface: UG stellt eine Skriptsprache zur Verfügung, mit Hilfe derer die komplexen Rechenabläufe gesteuert werden können und in der mit Hilfe von Matrix–Vektor–Operationen auch einfache Verfahren implementiert werden können.

Graphics: UG bietet eine Reihe von Möglichkeiten zur Visualisierung der Gitter und Simulationsergebnisse. Zusätzlich stehen Schnittstellen für die Ausgabe der Ergebnisse in verschiedenen Formaten bekannter Graphikprogramme zur Verfügung.

Grid Manager: UG erlaubt die Verwendung von Netzen aus Drei– und Vierecken in 2D und Tetraedern, Hexaedern und Pyramiden in 3D. Der Grid Manager stellt Funktionen zur Verfeinerung, Vergröberung, Standardformfunktionen für die verschiedenen Elementtypen und Routinen zur Verwaltung der verschiedenen Gitter zur Verfügung.

Numerics: Bereits in UG implementiert sind eine Reihe von Glättungsverfahren, Quadraturverfahren und lineare und nichtlineare Lösungsverfahren.

Für die speziellen Anforderungen der in dieser Arbeit untersuchten Kluft–Matrix–Systeme wurden in den Applications und der Problem Class Library eine Reihe von Ergänzungen vorgenommen. So wurde von L. Neunhäuserer [48] eine Schnittstelle implementiert, die es erlaubt, anstelle von einer direkt in UG enthaltenen Gebietsbeschreibung, die von ART bzw. FRACMESH erzeugten Datenfiles einzulesen und zu einer für UG verwertbaren Datenstruktur zu verarbeiten. Diese zunächst nur für die niederdimensionale Diskretisierung von Klüften implementierte Schnittstelle wurde im Rahmen des DFG–Projektes mit der Fakultät für Bauingenieurwesen der Universität Stuttgart (siehe DFG–Arbeitsbericht [31]) auf äquidimensional diskretisierte Klüfte erweitert und liefert die für die Implementierung der hierarchischen Zerlegung (siehe Abschnitt 5.3.3) benötigten Zusatzinformationen.
Als Grundlage für die Implementierung der hierarchischen Datenstrukturen wurden die Problem Class Library und die Applications für Finite Elemente von Christian Wieners [66] verwendet. Sie wurden um die Diskretisierung der partiellen Differentialgleichung für die stationäre Strömung (2.17) mit hierarchischen Basisfunktionen und den benötigten Koeffizientenfunktionen ergänzt und ihre Struktur der Problem Class Library für Einphasen–Zweikomponenten–Prozesse in MUFTE (siehe Abschnitt 5.3.1) angepaßt.

5.3.3 Datenstrukturen für hierarchische Gebietszerlegungen

Das Programmsystem MUFTE–UG stellt eine große Bandbreite von Anwendungen, Lösern, Diskretisierungen, Fehlerschätzern und Glättungsverfahren zur Verfügung. Für das in Abschnitt 3.4.1 vorgestellte hierarchische Gebietszerlegungsverfahren wurden Ergänzungen und Erweiterungen in den vorhandenen Datenstrukturen vorgenommen. Dazu wurden aufbauend auf die in UG bereits vorhandenen Datenstrukturen die bei der Aufspaltung des Gesamtproblems in die Teilprobleme Kluft–, Matrix– und Interface auftretenden hierarchischen Basisfunktionen so implementiert, daß sie eine hierarchische Gebietszerlegung in die Teilprobleme Matrix–, Interface– und Kluftproblem erlauben. Für den Linienglätter im Mehrgitterverfahren des Kluftproblems (siehe Lemma 4) konnte das in UG zur Verfügung stehende Block–Gauß–Seidel–Verfahren verwendet werden, dazu wurde eine Sortierung der Freiheitsgrade in den Klüften in Linien implementiert (siehe Abschnitt Kluftproblem). Auch die in den Abschnitten 4.1.1, 4.1.2 bzw. 4.1.3 vorgestellten Fehlerschätzer waren in der vorliegenden Version von UG noch nicht implementiert. Für die adaptive Verfeinerung wurden die in Abschnitt 4.1 beschriebenen Fehlerschätzer und die in Abschnitt 4.3 erläuterte Verfeinerungsstrategie in die bereits in UG zur Verfügung stehende Gitterverwaltung mit roten, blauen und grünen Verfeinerungen eingefügt.

Assemblierung der hierarchischen Steifigkeitsmatrix

In UG stehen eine Reihe von Formfunktionen für unterschiedliche Finite Elemete Diskretisierungen zur Verfügung. Diese werden auf dem jeweiligen Referenzelement betrachtet. Eine Diskretisierung mit Hierarchischen Basen war in UG zunächst nicht vorgesehen. Die Implementierung der Hierarchische Basen Diskretisierung auf der Grundlage der in UG vorhandenen Finite Elemente Diskretisierungen stellte also die Hauptschwierigkeit

bei der programmtechnischen Umsetzung des hierarchischen Gebietszerlegungsverfahrens aus Abschnitt 3.4.1 dar.

UG liefert die Möglichkeit, das Gesamtrechengebiet in einzelne Teilgebiete, die in UG als Subdomains bezeichnet werden, (z.B. Kluft und Matrix) zu unterteilen. Diese Unterteilung ist eindeutig, ein Element bzw. ein Teil des Rechengebietes gehört immer zu genau einer Subdomain. Die Anteile Kluft, Interface und Matrix der hierarchischen Gebietszerlegung überlappen im Bereich der Klüfte. Jedes Teilproblem hat dabei innerhalb der Klüfte andere Basisfunktionen. Betrachtet man die Basisfunktionen des Interfaceproblems als

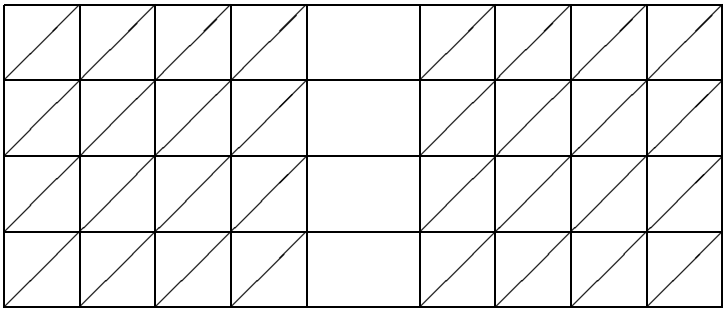

Abbildung 5.4: Gitter des Interface– und Matrixproblems

Knotenbasisfunktionen, so existiert das zugrundeliegenden Gitter im Kluftbereich zunächst nicht. Das Gitter der Kluftbasisfunktionen als Knotenbasis existiert hingegen (siehe Abbildungen 5.4 und 5.5). Für die Assemblierung der hierarchischen Gesamtsteifigkeitsmatrix haben wir die Basisfunktionen des Interfaceproblems im Bereich der Kluft auf den Kluftelementen dargestellt, die über die Kluft konstanten Basisfunktionen des Matrixproblems (siehe Abschnitt 3.3.3) innerhalb der Kluft ergeben sich dann durch eine geeignete Summierung aus den Basisfunktionen des Interfaceproblems. Für die Assemblierung der Basisfunktionen des Interfaceproblems auf den Kluftelementen wurden Informationen über die „Interface–Elemente“ aus den Kluftelementen, aus denen sich ein „Element“ des Interfaceproblems im Bereich der Kluft zusammensetzt (siehe Abbildungen 5.4 und 5.5), gewonnen. Ein „Element“

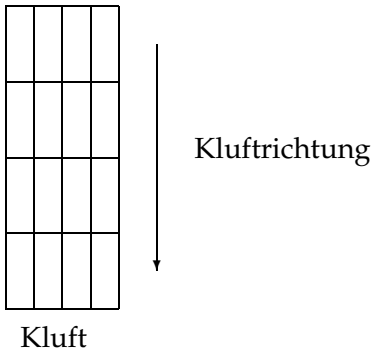

Abbildung 5.5: Das Gitter in der Kluft

des Interfaceproblems im Bereich der Kluft setzt sich aus den Kluftelementen zusammen, die durchlaufen werden, wenn man quer zur Kluftrichtung einmal die Kluftelemente durchläuft (siehe Abbildung 5.6). Kluftkreuzungen werden gesondert betrachtet. Dazu werden die zur Kluftrichtung parallelen Kanten — und damit auch die Elemente — in der Kluft quer zur Kluftrichtung (siehe Abb. 5.6) in jeweils einer doppelt verketteten Liste mit allen benötigten Informationen (Breite der Kluft, Abstand des Elementes von den Klufträndern etc. siehe Abbildung 5.7) gespeichert und an das Matrix– bzw. Interfaceproblem weitergegeben. Dies geschieht ausgehend vom Grobgitter und nutzt die von UG zur Verfügung gestellte Gitterhierarchie. Für die hierarchischen Basisfunktionen des Interfaceproblems auf den Kluftelementen (siehe Abb. 5.8) wurden dann entsprechende Formfunktionen implementiert, die auf die Informationen der Kantenliste zurückgreifen. Anschließend werden auf allen Kluftelementen einer Kantenliste mit den speziellen Formfunktionen die Einträge in die Steifigkeitsmatrix für die Knoten am Kluft–Matrix–Interface im Interface–Problem und darauf aufbauend auch die Einträge im Kluftbereich des Matrixproblems assembliert.

Die Vorgehensweise in den Kreuzungen ist ähnlich. Hier werden jeweils die Kluftelemente

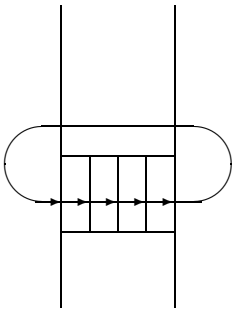

Abbildung 5.6: Verkettete Liste der Kanten quer zur Kluftrichtung

eines „Elementes“ des Interfaceproblems in einer Kluftkreuzung in einer doppelt verketteten Liste gespeichert (siehe Abbildung 5.10). Daraus erhalten wir die Informationen über die jeweils vier Knoten eines „Interface–Elementes“ in der Kluft (siehe Abbildung 5.9).

Für die Kopplung zwischen Interfaceknoten und Kluftknoten in den Nebendiagonalblöcken des hierarchischen Gleichungssystems (3.9) wird auf einem Kluftelement jeweils eine spezielle Formfunktion für die Basisfunktion des Interfaceproblems und eine Knotenbasisfunktion des Kluftproblems in der auszuwertenden Bilinearform zur Berechnung der Matrixeinträge herangezogen.

Die Basisfunktionen des Matrixproblems im Bereich der Klüfte ergeben sich als Summe der Basisfunktionen des Interfaceproblems zu gegenüberliegenden Interfaceknoten. Für sie wurden alle Knoten auf dem Interface in einer doppelt verketteten Liste mit Informationen über die gegenüberliegenden Knoten gespeichert (siehe Abbildung 5.10). An den Kluftkreuzungen werden jeweils die vier an einer Kreuzung anliegenden Knoten, deren Einträge in die Steifigkeitsmatrix für die Basisfunktionen des Matrixproblems addiert werden, in der Knotenliste gespeichert.

Für die Assemblierung des Matrixproblems haben wir auf den Knoten am Kluft–

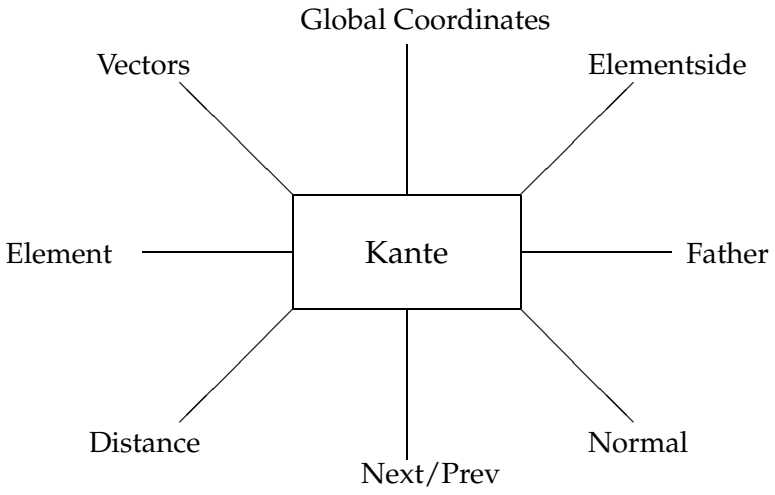

Abbildung 5.7: Struktur eines Eintrags in der Kantenliste

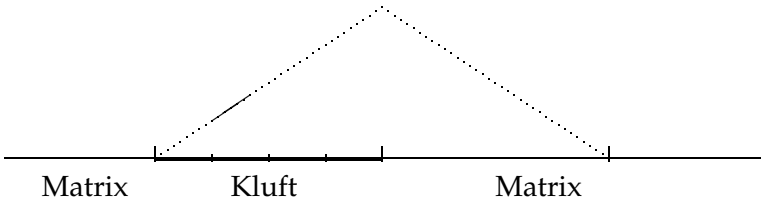

Abbildung 5.8: Basisfunktion des Interfaceproblems an der Kluft ($\cdots$), Basisfunktion des Interfaceproblems auf einem Kluftelement (—), Beispiel in 1–D

Matrix–Übergang einen zweiten Freiheitsgrad eingeführt. Dieser wird dem Matrixproblem zugewiesen. Die Funktionen des Matrixproblems sind konstant entlang der Kanten quer zur Kluftrichtung. Für die Basisfunktionen des Matrixproblems werden aus den Einträgen in die Steifigkeitsmatrix zu je zwei gegenüberliegenden Interfaceknoten p_k, q_k, die aus der Knotenliste bekannt sind, die Werte $a(\phi_j(p_k), \lambda_i) = a(\phi_j(q_k), \lambda) = a(\lambda_{p_k} + \lambda_{q_k}, \lambda_i)$ für die Einträge in die Steifigkeitsmatrix zu den Matrixbasisfunktionen in dem zusätzlichen Freiheitsgrad auf dem Interface gespeichert (siehe Abb. 5.10). Anschließend wird eine der nun doppelt in der Steifigkeitsmatrix vorhandenen Zeilen zu den Knoten p_k und q_k kondensiert, indem einer der beiden Knoten als eine einfache Verdoppelung des anderen aufgefaßt wird:

$$a_{p_k,i} = (0 \ldots 0\ 1\ 0 \ldots 0\ -1\ 0 \ldots 0).$$

$$\begin{matrix} \uparrow & \uparrow \\ p_k & q_k \end{matrix}$$

An einer Kluftkreuzung wird für die vier Knoten der Kreuzung entsprechend vorgegangen. Die Kopplung zwischen Matrix– und Interfaceknoten und Matrix– und Kluftknoten wird ebenso wie die Einträge des Matrixproblems aus den Einträgen der Steifigkeitsmatrix zu den

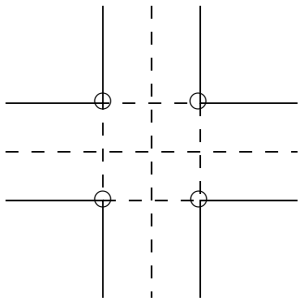

Abbildung 5.9: Knoten eines Matrixelementes in einer Kluftkreuzung

Interfaceknoten berechnet.

Auf der Grundlage der beschriebenen Listen wurden so in Anlehnung an die in der Problem

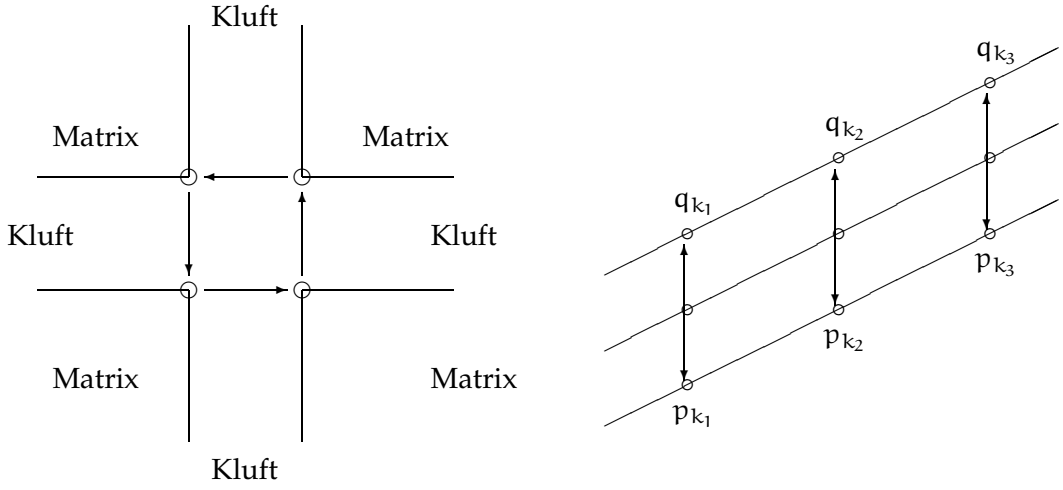

Abbildung 5.10: Liste der Eckknoten einer Kluftkreuzung (links) und der gegenüberliegenden Knoten an der Einzelkluft (rechts) für die Assemblierung der Einträge in der Steifigkeitsmatrix des Matrixproblems

Class Library (siehe Abschnitt 5.3.2) bereits vorhandenen Routinen zur Assemblierung Finiter Elemente Assemblierungsroutinen für Interface– und Matrixproblem im Kluftbereich bzw. am Kluft–Matrix–Interface und für die Kopplungsmatrizen zwischen Kluft–, Interface– und Matrixknoten entwickelt. Die Assemblierung für das Kluftproblem wurde so angepaßt, daß sie die etwas andere Struktur gegenüber der üblichen Knotenbasis berücksichtigt. Damit ist es möglich, unabhängig von der Verfeinerungstiefe in der Kluft die speziellen hierarchischen Basisfunktionen des Interfaceproblems über die gesamte Kluftbreite, die Einträge des Matrixproblems in die Steifigkeitsmatrix auf dem Interface und die Kopplung zwischen Kluft, Interface und Matrix in den Nebendiagonalblöcken der Gesamtsteifigkeitsmatrix zu assemblieren.

Nach Aufstellen des hierarchischen Gleichungssystems konnte der Teilraumkorrektur–Algorithmus, der sich aus der hierarchischen Gebietszerlegung ergibt, mit Hilfe der

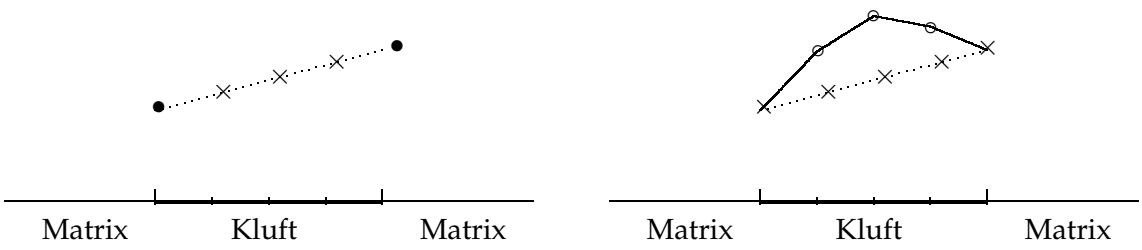

Abbildung 5.11: Links: × Interpolationswerte, • Matrixwerte, rechts: × Matrix– und Interpolationswerte, ○ Gesamtlösung

```
/* Liste der gegen"uberliegenden Knoten am Interface */
INT InitKnoten(NP_NL_ASSEMBLE *np, INT level);
INT Knotenupdate (NP_NL_ASSEMBLE *np, KNOTEN *KnListe, INT level, INT knl);

/* Liste der Kanten f"ur die Assemblierung, Listensortierung, Gitterverwaltung */
INT InitKanten(NP_NL_ASSEMBLE *np, INT level);
INT Kantenupdate (NP_NL_ASSEMBLE *np,KANTE *KanListe, INT level, INT kanl);

/* Liste der Knoten innerhalb der Kreuzung */
INT KreuzInit(NP_NL_ASSEMBLE *np, INT level);
INT Kreuzupdate (NP_NL_ASSEMBLE *np,KREUZ *KrListe, INT level, INT krl);

/* Sortierung von Freiheitsgraden in Linien in Kluftrichtung und Kreuzungsbl"ocke*/
INT InitLinie(NP_NL_ASSEMBLE *np, INT level);
INT InitKreuzblock(NP_NL_ASSEMBLE *np, INT level);
```

Abbildung 5.12: Übersicht der Listen–Funktionen

Skriptsprache von UG im Skriptfile implementiert (siehe Abbildung 5.15) werden.
Die implementierten Listen stellen den zentralen Faktor bei der Integrierung der hierarchischen Gebietszerlegung in die vorhandenen Datenstrukturen des Softwarepaketes UG dar. Eine Gesamtübersicht der für die hierarchische Gebietszerlegung implementierten Listen ist in Abbildung 5.12 aufgeführt, die entsprechenden Datenstrukturen sind in Abbildungen 5.13 und 5.14 zu sehen.
Die nun zur Verfügung stehenden Informationen über die Kluftkreuzungen werden auch bei der Umsortierung der Freiheitsgrade der Kluftkreuzungen in Blöcke genutzt (siehe Abschnitt Kluftproblem). Die vorgestellten Listen der Kluftkanten ermöglichen später die Linienbildung für den Linien–Glätter im Mehrgitter des Kluftproblems und werden auch für die adaptive Verfeinerungsstrategie wieder verwendet.
Für die Darstellung der in der hierarchischen Basis berechneten Ergebnisse in der bekannten Knotenbasis müssen die Ergebnisse von Matrix– und Interfaceproblem in die Knoten im Inneren der Klüfte linear interpoliert werden (Abbildung 5.11). Auch bei dieser Interpolation werden die Informationen aus der Datenstruktur „Kluftkante“ verwendet. Sie liefert Informationen darüber welche Knoten des Kluftproblems geeignet gewichtete Werte der Interface– und Matrixknoten zugewiesen bekommen.

```
typedef struct Knoten {
  ELEMENT *t; /* anliegendes Element */
  VECTOR *v; /* Vektor des Knoten */
  struct Knoten *prev, *next; /* verkettete Liste von Knoten */
  DOUBLE c0, c1; /* Koordinaten */
  INT index; /* Index innerhalb der Knotenliste*/
} KNOTEN;

typedef struct Kante {
  ELEMENT *t; /* Element zu dem die Kante gehoert */
  struct Kante *prev, *next; /* verkettete Liste von Kanten */
  INT seite; /* Nummer der Kante innerhalb des Elementes t */
  INT n0,n1; /* Nummer der "passenden" Knoten am Kluft-Matrix-Uebergang */
  DOUBLE ldist,rdist; /* Entfernung zum Kluftrand */
  DOUBLE c0[DIM], c1[DIM]; /* Koordinaten der Kantenendpunkte */
  INT cn0,cn1; /* Nummern der Vektoren zu den Kantenendpunkten */
  VECTOR *v[MAX_NODAL_VECTORS]; /* Liste von Vektoren auf der Kante */
  INT index; /* Index innerhalb der Kantenliste */
  DOUBLE b; /* Elementbreite zwischen n=0 und n=2, ist 0.0 falls t Matrixelement */
  DOUBLE_VECTOR normal; /* Normale auf die Kanten */
  INT x; /* Kluftrichtung */
  INT vater; /* Index des Vaterelements in der Liste auf level-1 */
}KANTE;

typedef struct Kreuz {
  ELEMENT *t; /* Element in der Kreuzung */
  ELEMENT *t0,*t1,*t2, *t3; /* die anliegenden Matrixelemente */
  INT m0,m1,m2,m3; /* die Nr. des Matrixelementknotens an der Kreuzung */
  struct Kreuz *next, *prev; /* verkettete Liste der Elemente der Kreuzung */
  DOUBLE c0[DIM],c1[DIM],c2[DIM],c3[DIM]; /* Koordinaten der Eckpunkte */
  INT index; /* Index in der Kreuzliste */
  VECTOR *v[MAX_NODAL_VECTORS]; /* Liste der Vektoren der Eckpunkte */
  DOUBLE labstandx,labstandy,rabstandx,rabstandy; /* Abstand Element-KrRand */
  INT x,y; /* x1 oder x2 Richtung */
  DOUBLE swx,swy; /*Schrittweiten des Parallelogramms */
  DOUBLE_VECTOR normal0,normal1,normal2,normal3; /*Normalen auf die Elementkanten*/
  INT vater; /* Index des Vaterelements in der Liste auf level = level-1 */
  struct Kreuz *nextx,*prevx; /* Nachbarelement in x-Richtung */
  struct Kreuz *nexty,*prevy; /* Nachbarelement in y-Richtung */
} KREUZ;
```

Abbildung 5.13: Datenstrukturen für Kanten, Knoten

```
typedef struct Linie{
   struct Linie *next, *prev;
   INT index; /* Index in der Linie */
   INT indexkante; /* Index der zugehoerigen Kante */
   INT start; /* Anfang der Linie */
} LINIE;

typedef struct Kblock{
    struct Kblock *next, *prev;
   ELEMENT *t; /* zugehoeriges Element */
   INT index; /* Index in der Liste */
   VECTOR *v; /* Vektor */
   DOUBLE c0[DIM]; /* Koordinaten */
} KBLOCK;
```

Abbildung 5.14: Datenstrukturen für Kreuzungblöcke und Linien

Matrixproblem

In UG besteht die Möglichkeit, die (Teil–) Gebiete und ihre Ränder durch Zuweisung unterschiedlicher Parts in verschiedene Gruppen zu unterteilen. Dadurch können diese Gruppen gesondert behandelt werden. So können nach erfolgreicher Assemblierung der Gesamtsteifigkeitsmatrix Berechnungen auf Teilen des Gesamt–Gleichungssystems (3.9) durchgeführt werden. Die Lösung des Matrixproblems stellt dann also einen Mehrgitter–V–Zyklus mit Gauß–Seidel–Glätter auf dem Subproblem dar, das entsteht, wenn man den Diagonalblock der Matrixproblemknoten des Gesamtgleichungssystems betrachtet:

$$A_{MM} u_M = B_M.$$

Die benötigten Routinen für das Mehrgitter–Verfahren und der Gauß–Seidel–Glätter stehen in UG zur Verfügung und können ohne weitere Ergänzungen genutzt werden.

Interfaceproblem

Ebenso wie im Matrixproblem betrachten wir beim Interfaceproblem einen Teil des Gesamtgleichungssystems (3.9):

$$A_{II} u_I = B_I.$$

Wir lösen dieses Teilproblem direkt. Dazu steht in UG eine LU–Zerlegung zur Verfügung.

Kluftproblem

Der in Abschnitt 3.3.1 erwähnte Löser für anisotrope Probleme ist ein Mehrgitter–verfahren mit Linienglätter für das Kluftproblem

$$A_{KK} u_K = B_K.$$

Der Glätter stellt eine spezielle Form eines Block–Gauß–Seidel–Verfahrens dar. In der vorliegenden Version von UG steht ein Block–Gauß–Seidel–Glätter für die UG–Datenstruktur BLOCKVECTOR zur Verfügung. Die Nutzung dieser Datenstruktur für das Block–Gauß–Seidel Verfahren erfordert die Umsortierung der ansonsten meist lexikographisch sortierten Freiheitsgrade des gesamten Gleichungssystems. Mit Hilfe der Datenstruktur der Kluftkanten und –kreuzungen zur Assemblierung der hierarchischen Basisfunktionen und der Auszeichnung spezieller Knoten durch die Zuweisung eines eigenen Parts, wie z.B. der Kluftknoten am Interface, konnten die Kluft–Freiheitsgrade in Blöcke bestehend aus Linien parallel zur Kluftrichtung und Kluftkreuzungen in BLOCKVECTORs sortiert werden. Anschließend konnte das vorhandene Block–Gauß–Seidel–Verfahren in UG als Glätter im Mehrgitterverfahren des Kluftproblems genutzt werden.

Adaptive Verfeinerung

In der Problem Class Library für Finite Elemente in UG steht eine Auswahl an Fehlerschätzern und Verfeinerungs–Algorithmen zur Verfügung. Auf dieser Grundlage wurden die Residuen–Fehlerschätzer für isotrope (Abschnitt 4.1.1) und anisotrope (Abschnitt 4.1.2) Elemente implementiert. Die adaptiven Verfeinerungsalgorithmen mußten den Datenstrukturen der hierarchischen Zerlegung angepaßt werden und die Verfeinerungsstrategie so implementiert werden, daß trotz der Unterteilung in ein Kluft–, Interface– und ein Matrixproblem bzw. der entsprechenden Subdomains von UG das Gesamtgitter verfeinert wird und dabei die Gitterstruktur, die für die hierarchische Zerlegung notwendig ist, erhalten bleibt.
Die anisotrope (blaue) Verfeinerung von Vierecken war in der vorliegenden Version von UG vorgesehen und weitestgehend implementiert. Sie wurde ergänzt, so daß nun zwei Kanten eines Vierecks zur Verfeinerung markiert und die Elemente entsprechend verfeinert werden können. Die in Abbildung 5.6 dargestellten Listen von Kanten quer zur Kluftrichtung werden auch bei der anisotropen (blauen) Verfeinerung der Kluft quer zur Kluftrichtung verwendet, indem immer zwei in der Kantenliste aufeinander folgende Kanten eines Elementes in der Kluft beim Durchlaufen der Liste an die blaue Verfeinerung als markiert übergeben werden.
Aufbauend darauf wurde der in Abschnitt 4.4 vorgestellte Verfeinerungsalgorithmus implementiert, der über einen Fehlerschätzer geeignete Elemente in Kluft und Matrix zur Verfeinerung auswählt und rot, grün oder blau verfeinert. Bei der adaptiven Verfeinerung werden wieder die für die Assemblierung implementierten Kantenlisten genutzt. Es wir dafür Sorge getragen, daß mit der in Abschnitt 4.3 vorgestellten Verfeinerungsstrategie ein Gitter für das gesamte Gebiet bestehend aus den Subdomains Kluft und Matrix ohne hängende Knoten am Interface entsteht. Die Kantenliste liefert zu einem Kluftelement immer das anliegende Matrixelement. Wird das Kluftelement markiert, so wird auch das Matrixelement markiert und umgekehrt werden, wenn ein Matrixelement markiert wird, alle Kluftelemente der zugehörigen Kantenliste blau markiert, wenn die Kluftelemente noch nicht markiert sind. Sind die Kluftelemente schon rot markiert, dann wurden alle Elemente der Kluft markiert und das Gitter ist konsistent, sobald die entsprechenden anliegenden Matrixelemente markiert sind.

Erweiterung: Kluftenden Für die in Abschnitt 3.5 vorgestellte Zweilevel–Variante des vorgestellten hierarchischen Verfahrens für Kluft–Matrix–Systeme haben wir auch eine

```
repeat { #Schleife ueber die level
  npexecute assembleM ; #Assemblierung auf den Dreiecken in der Gesteinsmatrix
  npexecute myassembleM $km; #Assemblierung der hierarchischen Basis
  npexecute assembleK $kluft; #Assemblierung der Kluftbasisfunktionen
  npexecute assembleMK $MKI; #Assemblierung der Nebendiagonalbloecke
  npexecute transferM $s; #Interpolation fuer das Mehrgitter des Matrixproblems
  @lexorder; #lexikographische Sortierung der Freiheitsgrade
  lineorder $A GMAT $kℓuft Ksol $kr KRsol $ass myassembleM; #Linien fuer Line−GS
  repeat{ # Schleife fuer die Iteration auf einem level
    npexecute Mmgs $i $d $r $s $p; # MMat*Msol = Mrhs
    matmulminus $x Krhs $y Msol $m KMMAT; #Update der rechten Seite
    matmulminus $x MRrhs $y Msol $m IMMAT;
    npexecute Imgs $i $d $r $s $p; #IMat*Isol = Irhs
    copyvector $f Isol $t Mhelp $ass myassembleM $MI;
    add $x Msol $y Mhelp;
    add $x Mhsol $y Msol; # Update der Loesung
    matmulminus $x Krhs $y MRsol $m KIMAT;
    matmulminus $x MNrhs $y MRsol $m MIMAT;
    if(l>=1)   {
      npexecute Kmgs $r $i $s $p $d; #KMat*Ksol = Krhs
      add $x Khsol $y Ksol;
      matmulminus $x Mrhs $y Ksol $m MKMAT;
      matmulminus $x Irhs $y Ksol $m IKMAT;
      npexecute li; # lineare Interpolation
      add $x Khsol $y Ksol;
    }
    iter= iter+1;
    if(iter>=VITER) break;
  } #repeat−Schleife fuer die Iteration auf einem level
  if (l >= MAXSTEP) break;
  l = l + 1;
  npexecute MKerror;
  refine $g;
  npexecute transferM $s $N;
} # Schleife ueber die level
```

Abbildung 5.15: Teilraumkorrektur–Algorithmus

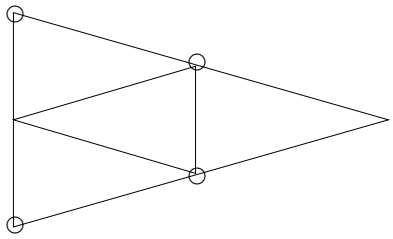

Abbildung 5.16: Knoten eines Matrixelementes in einem Kluftende

Lösung für Kluftenden implementiert. Die entsprechenden Grobgitter wurden von ART und FRACMESH erzeugt. Dabei erweisen sich an den Enden der Klüfte im Gebiet die unterschiedlichen Schrittweiten in der Kluft und der Matrix als ein Problem. Die Schrittweite in der Kluft kann quer zur Kluftachse die Kluftbreite nicht übersteigen, ein so feines Gitter in der Matrix würde aber zu einer starken Vergrößerung der Anzahl der Knoten in der Matrix und damit zu erheblich längeren Rechenzeiten führen. Soll eine Anpassung der Schrittweite in der Matrix im Kluftendbereich an die Kluftbreite vermieden werden, ergeben sich im Netzgenerator ART, der ursprünglich für 1D Klüfte konzipiert ist und erst mit dem Erweiterungsmodul FRACMESH für 2D Klüfte genutzt werden kann, mit Blick auf die hierarchische Zerlegung grundsätzlich zwei Möglichkeiten am Kluftende (siehe Abbildung 5.17).

Eine Möglichkeit ist, die Kluft mit einem langen spitz zulaufenden Dreieck abzuschließen.

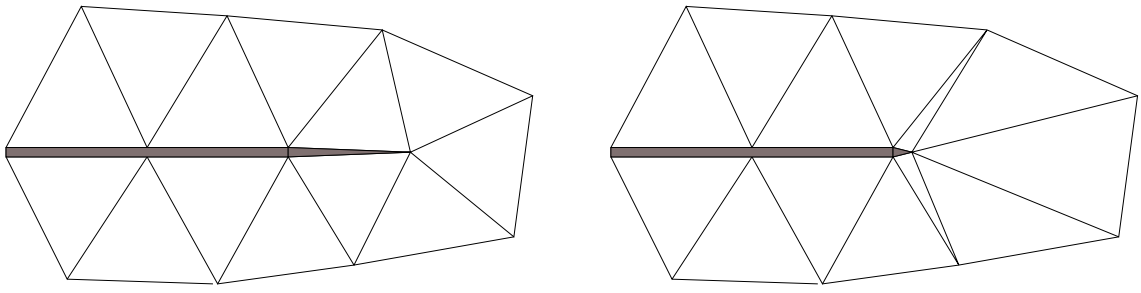

Abbildung 5.17: Verschiedene Möglichkeiten eines Kluftendes: langes Dreieck (links) kurzes Dreieck (rechts)

Dann bleibt die von ART für die niederdimensionale Kluft gelieferte Elementstruktur in der Matrix im wesentlichen erhalten, nur in der Kluft tritt ein Element mit einem Innenwinkel α auf, der für verschwindende Kluftweite ($\varepsilon \to 0$) ebenfalls gegen Null geht. Dabei ist der Sprung in den Koeffizienten der Permeabilität $K(x)$ genau im Winkel α.

Die andere Möglichkeit für äquidimensionale Kluftenden ist es, die Kluft mit einem sehr kurzen Dreieck abzuschließen, so daß am Kluftende kein spitzer Winkel entsteht. Dabei entstehen zwangsläufig in der Matrix Elemente mit kleinen Winkeln. Es hat sich generell als

weniger problematisch erwiesen, wenn die sehr kleinen Winkel in der Matrix auftreten, da dann der Koeffizientensprung nicht in spitzen Winkeln auftritt. Wir haben in dieser Arbeit daher diese Variante gewählt. In beiden Fällen sind Dreiecke mit spitzen Winkeln nicht zu vermeiden. Günstiger wäre eine der Geometrie angepaßte Vernetzung, die um das Kluftende in der Matrix feiner ist, eine Verdichtung der Elemente in der Gesteinsmatrix um das Kluftende, so daß in der Gesteinsmatrix keine langen schmalen Elemente auftreten.
Eine weitere denkbare Möglichkeit für die Kluftenden ist, die Klüfte in einem Viereck enden zu lassen. Dabei würde in der Matrix wieder ein spitzer Winkel auftreten. Diese Vorgehensweise ähnelt daher der Variante mit einem kurzen Dreieck am Kluftende ohne Koeffizientensprung in den spitzen Winkeln. Die Variante mit Vierecken als Kluftende erfordert allerdings eine von der in dieser Arbeit verwendeten und implementierten Struktur des hierarchischen Verfahrens abweichende Vorgehensweise.
Die Verfeinerung der Kluftenden stellt ein nicht ganz unerhebliches Problem dar, wenn die Möglichkeit die Klüfte anisotrop (also nur in eine Richtung) zu verfeinern beibehalten werden soll. Eine Erläuterung der Probleme und Vorschlag zur Lösung sind in Kapitel 7 zu finden.