Numerische Lösung und Modellierung eines inversen Problems zur Assimilation digitaler Bilddaten

I n a u g u r a l - D i s s e r t a t i o n

zur

Erlangung des Doktorgrades der

Mathematisch-Naturwissenschaftlichen Fakultät

der Heinrich-Heine-Universität Düsseldorf

vorgelegt von

Stefan Henn

aus Düsseldorf

Februar 2001

Gedruckt mit der Genehmigung der

Mathematisch-Naturwissenschaftlichen Fakultät der

Heinrich-Heine-Universität Düsseldorf

Referent:	Prof. Dr. K. Witsch
Korreferentin:	Prof. Dr. M. Hochbruck
Korreferent:	Prof. Dr. F. Jarre

Tag der mündlichen Prüfung: 08.02.2001

Internetadresse: http://www.ulb.uni-duesseldorf.de/diss/mathnat/2001/henn.htm
Email: henn@uni-duesseldorf.de

Inhaltsverzeichnis

Kapitel 1

Einleitung

Die vorliegende Arbeit beschäftigt sich mit der Identifikation von physikalischen Größen, auf die man z.B. durch direkte Messungen keinen Zugang hat, sondern auf die man durch Beobachtung anderer verfügbarer Größen schließen muss. Hierbei wurden im Wesentlichen aus der Medizin bekannte Aufgabenstellungen betrachtet, die sich aber leicht auf andere naturwissenschaftliche Probleme übertragen lassen.

In der medizinischen Hirnforschung wird mit Hilfe histologischer Bildsequenzen die Funktionsweise des menschlichen Gehirns erforscht. Bei der Herstellung histologischer Serienschnitte treten unvermeidlich Deformationen auf. Deshalb ist es erforderlich, die histologische Bildinformation mit der Information anderer nicht deformierter Bilddaten, die z.B. mit der Magnetischen Resonanz-Tomographie (MRT) erzeugt wurden, räumlich zu überlagern. Ziel einer solchen Anpassung ist daher, einerseits den Nachteil makroskopischer Auflösung der MRT-Bilder durch die mikroskopische Information zu ergänzen und andererseits den Nachteil nichtlinearer Deformationen in histologischen Bildsequenzen durch die nahezu unverzerrten MRT-Aufnahmen als Referenz zu kompensieren. Die Anwendung von Anpassungsverfahren wird in der Medizin auch bei der Klärung von Fragestellungen zur Organisation und Variation des menschlichen Gehirns notwendig, wie etwa bei der Berechnung eines menschlichen Referenzgehirns. Hierbei wird die durch ein bildgebendes Verfahren (MRT) erzeugte Information verschiedener Individuen überlagert.
In der Meteorologie sind wichtige Größen bei der kurz- und mittelfristigen Wettervorhersage die Windrichtung und die Windgeschwindigkeit. Durch die Assimilation von zu verschiedenen Zeitpunkten von einem Satelliten gesendeten Wolkenbildern

kann auf die Verschiebung der Wolken und damit auch auf die gesuchten Winddaten geschlossen werden.

Mathematisch fallen die Probleme in die Klasse inverser Probleme (engl. inverse problems), da man Aussagen über Größen (die Verschiebungen bzw. Deformationen) treffen will, die für Messungen unzugänglich sind. Deshalb wird in diesen Fällen durch die Beobachtung anderer verfügbarer Größen (die Bildinformation) auf die gesuchte Verschiebung zurückgeschlossen.

Zur Lösung der beschriebenen Aufgabenstellung im Bereich der Medizin sind in den letzten Jahren eine große Menge von Lösungsansätzen publiziert worden. Hierbei werden speziell ausgezeichnete Informationen der Bilddaten entsprechend aufeinander transformiert. Die wichtigsten Ansätze sind hierbei punkt-, kurven- und oberflächenbasierte Anpassungsverfahren. Bei den punkt- und kurvenbasierenden Anpassungsverfahren (vgl. z.B. [10]) werden vom Mediziner eingetragene Strukturen (Punkte oder Kurven) aufeinander transformiert, während man bei oberflächenbasierten Anpassungsverfahren (vgl. z.B. [35]) die vorgegebenen Gehirnoberflächen anpasst. Die berechneten Verschiebungen werden hierbei durch eine geeignete Interpolation auf den gesamten Bilddatensatz übertragen.

In Kapitel 2 dieser Arbeit erfolgt die mathematische Modellierung des Problems nicht für spezielle Strukturen der Datensätze, sondern durch die gesuchten Deformationen $\phi(x, u(x)) = x - u(x)$ mit der Verschiebung $u(x)$ für alle Bildelemente von zwei vorgegebenen Signalen (Bilder). Das erste Bild, das sogenannte Referenzbild $R(x)$, enthält hierbei keinerlei Deformationen, während das zweite Bild $T(\phi(x, u(x)))$ als deformiert angenommen wird. Das Problem wird als Identifikationsproblem beschrieben. Bei gegebenem Input T und Output R sollen die Systemparameter $u(x)$ so bestimmt werden, dass die Zustandsgleichung

$$T(\phi(x, u(x))) = R(x)$$

möglichst gut erfüllt ist. Die Zustandsgleichung liefert eine Vorschrift zu Transformation der Datensätze in der die Information aller Bildpunkte einfließt und nicht nur die spezieller Strukturen.

Unglücklicherweise vereint das Identifikationsproblem nicht alle Eigenschaften, die man im Sinne von Hadamard von einem gut gestellten (engl. well-posed) Problem

erwartet. Lösungen, wenn solche existieren, sind weder eindeutig noch stabil, so dass es sich um ein sogenanntes schlecht gestelltes (engl. ill-posed) Problem handelt. Es zeichnet sich dadurch aus, dass vollkommen unterschiedliche Systemparameter zu beinahe oder vollständig identischem Output führen können. Umgekehrt können kleinste Datenfehler dazu führen, dass der Output von weit entfernten Systemparametern viel besser reproduziert wird als von den ursprünglich berechneten Systemparametern.
Zudem existieren im Allgemeinen sehr viele Parameter $u(x)$, die die Zustandsgleichung lösen. Die meißten Parameter sind jedoch hochgradig unstetig, wie zum Beispiel solche die wahllos (Grau-)Werte von T auf die selben Werte von R überführen. Solche Parameter sind als Lösung des Problems uninteressant, und werden deshalb durch eine modellgerechte Regularisierung in Abschnitt 2.3 ausgeschlossen.

Grundlegende Arbeiten für die Berechnung von Verschiebungsvektoren für digitale Bilddaten sind von Horn und Schunk ([24], 1981) zur Berechnung des optischen Flusses und die Erweiterung des Verfahren auf mehrere Bildauflösungsstufen durch Dengler und Schmidt ([11], 1988). Die erste bekannte Arbeit, die die Bestimmung von Verschiebungsvektoren zur Anpassung medizinischer Bilddaten beschreibt, stammt von Bajcsy und Kovacic ([4], 1989).
Amit et al. ([1], 1994) berechnet für Bilddaten einer Hand die Verschiebungsvektoren mit Hilfe des Gradientenabstiegs im Fourierraum, der pseudo-spektral Methode und der Wavelet Methode. In der Arbeit von Christensen et al. ([8], 1996) werden die digitalen Bilddaten als Flüssigkeit modelliert und die Grauwertdifferenz durch ein iteratives Abstiegsverfahren minimiert, welches auf der Berechnung der Abstiegsrichtung und deren Länge durch eine gewöhnliche Differentialgleichung basiert.

In dieser Arbeit wird in Abschnitt 2.2 der Begriff der Lösung verallgemeinert, um für die Zustandsgleichung die Existenz einer Lösung zu erzwingen. Als Lösung akzeptiert werden deshalb alle Verschiebungen $u(x)$, die die Defektnorm

$$D(u(x)) = ||T(\phi(x, u(x))) - R(x)||_{L_2}^2$$

minimieren. $D(u(x))$ wird auch als Defektfunktional oder Zielfunktional bezeichnet.

Instabilitäten wird, wie bei vielen inversen Problemen, durch Regularisierung begegnet. Hierzu werden in Kapitel 3 zunächst allgemeine Abstiegsverfahren zur Lösung des Problems eingeführt. Die Minimierung des Defektfunktionals kann

hiermit zwar durchgeführt werden, dies führt aber zu keinen geeigneten Ergebnissen. Deshalb werden die Abstiegsverfahren auf zweierlei Weise regularisiert. Beide Ansätze führen auf ein System partieller gekoppelter linearer Differentialgleichungen mit Dirichlet-Randbedingungen, deren eindeutige Lösungen die Abstiegsrichtung des Defektfunktionals vorgeben.
Bei der Verwendung des klassischen Regularisierungsverfahrens von Tikhonov wird dem Defektfunktional ein Glättungs- oder Strafterm, der durch einen positiven Regularisierungsparameter α gewichtet wird, hinzugefügt. Im so erzeugten Tikhonov-Funktional wird der von Tikhonov durch die kanonische Norm definierte Glättungsterm durch eine symmetrisch, positiv semidefinite Bilinearform ersetzt. Sie misst die potentielle Energie einer elastischen Verschiebung und modelliert die im Bild dargestellten Objekte wie ein elastisches Medium. Auf der Suche nach einer Minimallösung des Tikhonov-Funktionals mit festem Regularisierungsparameter stellt man fest, dass die Lösungen auch Lösungen einer nichtlinearen Variationsgleichung sind, die die schwache Form einer nichtlinearen partiellen Differentialgleichung ist.

In Kapitel 4 werden die partiellen Differentialgleichungen geeignet diskretisiert. Hierzu werden die digitalisierten Bilddaten als stückweise konstante Funktionen aufgefasst. Man erhält eine Zerlegung, die auf achsenparallelen Teilgebieten mit konstanter Breite definiert sind. Das Diskretisierungsgitter für die Verschiebungsvektoren wird so ausgerichtet, dass die Gitterpunkte gerade in der Mitte der Bildelemente postiert sind. Die so erhaltene Diskretisierung der Verschiebungsvektoren wird allgemein als cell centered Diskretisierung bezeichnet.
Die Approximation der Gleichung erfolgt durch die Finite Differenzen Methode (FDM). Hierbei werden die auftretenden Ableitungen in den partiellen Differentialgleichungen durch Differenzenquotienten zweiter Ordnung ersetzt.
Die numerische Lösung erfolgt durch Mehrgitterverfahren, deren Grundidee die Lösung nach verschiedenfrequenten Fehlerkomponenten auf verschiedenen Auflösungsstufen im Folgenden kurz dargestellt wird. Sie zeichnen sich gegenüber anderen Verfahren zur iterativen Lösung partieller Differentialgleichungen dadurch aus, dass ihre Konvergenzrate im Allgemeinen unabhängig von der gewählten Schrittweite h ist und das ihr asymptotischer Aufwand ($\mathcal{O}(N)$) proportional zur Anzahl der Gitterpunkte N ist. Außerdem werden für die linearen und nichtlinea-

ren Gleichungen geeignete Glättungsverfahren sowie Transferoperatoren für den Übergang zwischen den verschiedenen Gitterebenen beschrieben. Hierdurch wird ein deutlicher Zeitvorteil bei der Berechnung der Verschiebungsvektoren gegenüber anderen Anpassungsverfahrenerfahren, wie etwa [4] mit dem Jacobi Gesamtschrittverfahren oder [8] mit dem sukzessiven Überrelaxationsverfahren erreicht. Man ist somit in der Lage zweidimensionale Bilddaten mit hohem Informationsgehalt (feiner Abtastschrittweite) und dreidimensionale Bilddatensätze zu verarbeiten.

Neben der Analyse des Problems und dessen Lösungsverfahren spielt auch die effiziente Lösung eine entscheidende Rolle. In Kapitel 5 werden die Methoden zur Minimierung des Defektfunktionals mit den Mehrgitterverfahren kombiniert. Zunächst werden die Verfahren, die auf der Verfolgung von Abstiegsrichtungen des Defektfunktionals basieren, beschrieben. Die Verfahren benutzen in jedem Iterationsschritt ein lineares MGV zur Berechnung der Abstiegsrichtung.
Ein weiterer Ansatz zur Bestimmung einer Näherungslösung ist die Minimierung des nichtlinearen Tikhonov-Funktionals. Das Problem hierbei ist eine geeignete Wahl des Regularisierungsparameters α. Durch den Parameter α wird Einfluss auf das Verhältnis der Defektnorm und die Glattheit der regularisierten Lösungen genommen. Unter geeigneten Voraussetzungen sollte sich mit fallendem α auch der Wert der Defektnorm verringern. Für kleine α wird die numerische Berechnung einer Lösung mit einem nichtlinearen Mehrgitterverfahren instabil. Andererseits können die Lösungen für große α numerisch stabil berechnet werden. Die Lösungen werden hierdurch jedoch so stark geglättet, dass die Defektnorm kaum kleiner wird.
Der optimale Regularisierungsparameter hängt von der Lösung selbst ab und kann daher nicht anhand der Bilddaten a priori bestimmt werden. Zur näherungsweisen Bestimmung optimaler Regularisierungsparameter werden zwei verschiedene Ansätze verfolgt. Zum einen wird α innerhalb des nichtlinearen Mehrgitterverfahren so gewählt, dass sich der Wert der Defektnorm auf den verschiedenen Gitterebenen geeignet reduziert. Der zweite Ansatz nutzt aus, dass die mit großem α berechneten Lösungen zwar zu glatt sind, um die Defektnorm entscheidend zu reduzieren, aber eine geeignete Startnäherung für die Minimierung des Tikhonov-Funktionals mit verkleinertem α sind. Es wird deshalb eine Folge von Minimierungsproblemen von Tikhonov-Funktionalen mit abnehmendem α betrachtet, bei der die näherungsweise

berechnete Lösung der vorherigen Iterierten als Startwert des aktuellen Minimierungsproblems fortgesetzt wird.

Bei genauer Analyse der Minimierungsverfahren erkennt man, dass hierbei zwei Probleme auftreten. Einerseits werden bei einer kleinen Bildabtastschrittweite die kleinen Strukturen (hohe Bildfrequenzen) bevorzugt angepasst, jedoch größere vernachlässigt. Das führt dazu, dass die Minimierungsverfahren in vielen Fällen eine unbrauchbare lokale Minimalstelle berechnen. Außerdem sind zum Erreichen dieser Minimalstelle viele Iterationschritte der Minimierungsverfahren notwendig, so dass der Minimierungsprozess gerade auf feinen Auflösungsstufen sehr lange dauert. Deshalb wird in Abschnitt 5.5 ein Mehrgitteransatz zur Berechnung geeigneter Startwerte der Minimierungsverfahren vorgestellt. Hierzu wird das Defektfunktional (d.h. die digitalen Bilddaten) auf gröbere Bildauflösungen transformiert. Das Funktional wird hier mit den jeweiligen Verfahren minimiert und die Lösung auf die nächst feinere Gitterstufe übertragen. Dabei werden die verschieden frequenten Bildstrukturen auf den jeweiligen Bildauflösungsstufen angepasst. Abschließend werden die Verfahren an synthetischen und realen Bilddaten getestet.

Der Informationsgehalt der Bilddaten ist in vielen Anwendungen enorm. Typischer Weise enthalten in der Medizin zweidimensionale Schnittbilder bis zu 512×512 pixel (engl. picture elements) und dreidimensionale Volumendatensätze $256 \times 256 \times 128$ voxel (engl. volume picture elements). Aufnahmen der Erdoberfläche können bis zu 4096×4096 Bildelemente enthalten. Wegen dieser Datenmengen wird in Kapitel 6 der Arbeit das Parallelisierungspotential der Minimierungsverfahren diskutiert und ein geeigneter paralleler Algorithmus entwickelt. Die parallele Implementierung der Gitteralgorithmen auf Parallelrechnern mit verteiltem Speicher basiert auf der Methode des Grid Partitioning. Hierbei werden die digitalen Bilddaten in verschiedene Teilbilder zerlegt und auf entsprechend viele Prozessoren verteilt. Die Berechnung der gesuchten Verschiebungen erfolgt auf den einzelnen Prozessoren. Die verschiedenen Gitterebenen werden hierbei hierarchisch nacheinander abgearbeitet, während die Behandlung einzelner Gitterpunkte auf einer Gitterebene parallel durchgeführt wird.

Das Abschlußkapitel 7 faßt die in verschiedenen Einzelschritten gewonnenen Ergebnisse zusammen, wertet sie aus und bestimmt ihren Stellenwert für die allgemeine

Forschungslage.

Die Programme zur Implementierung der vorgestellten Algorithmen sind mit den Programmiersprachen C bzw. C++ geschrieben. Aus Platzgründen werden die Programmzeilen hier nicht aufgeführt. In Anhang A wird jedoch ein kurzer Überblick über die Struktur der verwendeten C++-Klassen gegeben.

Kapitel 2

Inverse Problemstellung

In diesem Kapitel wird zunächst ein grober Überblick über inverse Fragestellungen gegeben. Anschließend wird das in dieser Arbeit vorliegende Problem als Identifikationsproblem eingeordnet und diskutiert.

2.1 Inverse Probleme

In der Literatur werden inverse Probleme häufig als Umkehrung von direkten Problemen motiviert. Eine genaue Beschreibung von direkten und inversen Problemen ist jedoch nicht bekannt. In [5] wird der Begriff des inversen Problems präzisiert. Ausgehend von einem mathematischen Modell eines physikalischen Prozesses mit eindeutig definierten Eingangs- und Ausgangsdaten, wird das den Prozess beschreibende Modell durch

- die Eingangsgrößen (Input),
- die Ausgangsgrößen (Output),
- und die Systemparameter

charakterisiert. Sei X der Raum der Eingabegrößen, Y der Raum der Ausgabegrößen und P der Raum der Systemparameter, dann ist das mathematische Modell eines physikalischen Prozesses durch die Gleichung:

$$y = A(p)x, \quad p \in P$$

gegeben. In [5] werden drei Arten von Problemen unterschieden:

(P1) das direkte Problem berechnet mit gegebener Eingabe $x \in X$ und gegebenem Systemparameter $p \in P$ die Ausgabe $y \in Y$,

(P2) das Rekonstruktionsproblem berechnet die Eingabe $x \in X$ bei gegebener Ausgabe $y \in Y$ und gegebenem Parameter $p \in P$,

(P3) das Identifikationsproblem bestimmt den Systemparameter $p \in P$, bei gegebener Ein- und Ausgabe.

Die Probleme (P2) und (P3) sind inverse Probleme. Das direkte Problem (P1) beschreibt einen Ursache-Wirkung-Effekt. Das Rekonstruktionsproblem versucht im Gegensatz dazu aus einer bekannten Wirkung die Ursachen zu bestimmen. Das in dieser Arbeit betrachtete Problem wird im folgenden Kapitel als Identifikationsproblem beschrieben, bei dem die Informationen in Form von Messergebnissen vorliegen und aus denen die Systemparameter identifiziert werden sollen.
Interessanterweise sind ein Teil der in dieser Arbeit verwendeten Daten ihrerseits Ergebnisse eines Identifikationsproblems. In der medizinischen Diagnostik dienen Identifikationsaufgaben zur nichtinvasiven Beschaffung von Informationen über das Körperinnere. Dies geschieht durch äußere Messungen von Strahlungsintensitäten oder Magnetfeldern, mit denen dann durch Integraltransformationen auf die gesuchten Parameter (Bilder) zurückgeschlossen wird.

2.2 Problemstellung

Die in dieser Arbeit betrachtete Aufgabenstellung ist in der Medizin als Matching oder Registrierung bekannt. In der medizinischen Bildverarbeitung ist hiermit die Veränderung oder Transformation von digitalen Bilddaten gemeint. Diese wird vorgenommen, um einerseits die Bildinformation besser vergleichen zu können und andererseits Informationen über die Transformation zu erhalten. In der medizinischen Anwendung unterscheidet man im Wesentlichen vier verschiedene Gebiete, die im Folgenden kurz dargestellt werden. Einen weiterreichenden Überblick erhält man zum Beispiel aus [29].

- Durch die Registrierung von Bildinformationen, die durch verschiedene Sensoren aufgenommen wurden (multimodale Bildinformation), werden unterschiedliche

Informationen aus histologischen Bildsequenzen und/oder verschiedenen neuen bildgebenden Verfahren wie

- der Röntgen-Computertomographie (CT),
- der Magnetischen Resonanz-Tomographie (MRT) und
- der Positron-Emissiontomographie (PET)

integriert und hierdurch der Informationsgehalt der resultierenden Bilddaten erhöht. Von medizinischem Interesse ist auch die Integration von Bilddaten mit funktionellem und strukturellem, also anatomischem Inhalt, (vgl. [13] für Anpassungen auf der Basis von PET- und MRT- Bilddaten).

- Bei der Integration von Bilddaten einer Modalität (monomodaler Bildinformation) wird eine interindividuelle Anpassung von Bilddaten gleicher Modalität durchgeführt. Zu den praktischen Anwendungen gehört auch die Rekonstruktion von Unterschieden in den Bilddaten, die zu unterschiedlichen Zeitpunkten gemessen wurden.

- Weitere Anwendungen, deren Realisierung nicht das direkte Ziel dieser Arbeit waren, ist das sogenannte Template Matching. Hierbei werden Bilddaten von Patienten mit den Bildern aus einer Bilddatenbank (sogenannter Atlas) verglichen. Der Atlas besteht aus Bildern von Referenzpatienten, aus künstlich erstellten Bildern oder aus Bildern statistischer Untersuchungen.

- Beim physikalischen Matchen werden Koordinaten der Bilddaten auf die physikalischen Koordinaten im Körper eines Patienten transformiert. Eine bekannte Anwendung hierfür ist das computergestützte Operieren (engl. computer assisted surgery (CAS)), das in den letzten Jahren sehr stark entwickelt wurde.

Im C. und O. Vogt Institut für Hirnforschung der Heinrich-Heine-Universität Düsseldorf wird durch die Zusammenführung der mikroskopischen Information deformierter histologischer Bildsequenzen und der exakten Lage von MR-Bildsequenzen mit makroskopischer Auflösung ein digitales Bild erzeugt, das die gemeinsame Information beider Bilddatensätze enthält. Hierdurch wird eine Informationsverdichtung erreicht,

die in der aktuellen medizinischen Forschung (vgl. z.B. [16]) große Bedeutung für das Verständnis der Funktionsweise des menschlichen Gehirns hat.

Zur Klärung von Fragestellungen zur Organisation und Variation des menschlichen Gehirns wird in der Hirnforschung ein menschliches Referenzgehirn berechnet, bei dem Strukturen verschiedener Individuen des menschlichen Gehirns überlagert werden müssen. Hierzu ist eine interindividuelle Anpassung verschiedener MRT-Bilddatensätze notwendig.

Die in dieser Arbeit betrachtete Problemstellung besteht in der Identifikation der Deformationen u, die zwei verschiedene Bilddatensätze in geeigneter Weise aufeinander transformiert. Da die direkte Bestimmung der Deformationen (etwa durch Messungen) nicht möglich ist, wird durch Beobachtung anderer verfügbarer Größen das Problem invers gestellt. Durch diese Aufgabenstellung ergibt sich auf natürliche Weise das folgende Optimierungsproblem:

Bestimme u aus einem Parameterraum $\mathcal{X}$, so dass das durch u modellierte mathematische Problem möglichst gut mit dem beobachteten Verhalten des realen (physikalischen) Systems übereinstimmt.

Die mathematische Modellierung des Problems erfolgt durch die kontinuierlichen Signale (Bilder) $T, R : \Omega \to I\!R$, wobei das sogenannte Referenzbild $R(x)$ keine Deformation enthält. Die Deformation eines Objekts wird durch die Abbildung

$$\begin{aligned} \phi : \Omega \times \mathcal{X} &\to \Omega \\ \phi : (x, u) &\mapsto x - u(x) \end{aligned}$$

beschrieben. Sie transformiert ein Bildelement in eine eindeutig bestimmte Position $\phi(x, u(x))$. Durch Betrachtung von $R(x)$ und des durch $\phi(x, u(x))$ deformierten Templatebildes $T(x)$, erhalten wir die folgende Zustandsgleichung

$$T(x - u(x)) \;=\; R(x), \tag{2.1}$$

aus der auf geeignete Weise auf die Verschiebung $u(x)$ geschlossen werden soll.
In der Praxis ist die Erfassung dieses physikalischen Gesamtzustands nicht möglich, deshalb werden die Bilder digitalisiert, so dass $T^h, R^h \in I\!B^h$ mit

$$I\!B^h : \Omega_h \to [0, g_{max}] \subset I\!N$$

mit maximalem Grauwert g_{max} (typisch $g_{max} = 255$ bei Grauwertbildern). Für numerische Berechnungen wird die Grauwertquantisierung in ein Intervall $[0,1] \subset \mathbb{R}$ übertragen. Das Diskretisierungsgitter Ω_h wird häufig durch das durch ein Gitter überdeckte Einheitsquadrat $[0,1]^2$ bzw. den Einheitswürfel $[0,1]^3$ für zwei- bzw. dreidimensionale Probleme ersetzt.

Vor der Entwicklung effizienter numerischer Lösungsverfahren für das beschriebene Problem steht die Klärung einiger grundlegender Fragen, die den Informationsgehalt des inversen Problems festlegen.
Durch den nichtlinearen Operator $F(u(x)) := T \circ \phi(x,u)$ kann das Problem als Identifikationsproblem formuliert werden. Gesucht wird der Systemparameter $u \in \mathcal{X}$, so dass für $F(u(x)) = R(x)$ gilt. Die berühmte These von Hadamard aus dem Jahr 1923 (HADAMARD,[20]) besagt, dass ein Problem korrekt (gut, sachgemäß) gestellt ist, falls

(1) zu jedem Datensatz eine Lösung existiert,

(2) die Lösung eindeutig bestimmt ist,

(3) die Lösung stetig von den Daten abhängt.

Probleme, die nicht eine oder mehrere Bedingungen 1 bis 3 erfüllen, nennt man inkorrekt oder schlecht gestellt. In diesem Zusammenhang sei betont, dass ein inkorrekt gestelltes Problem nicht bedeutet, dass die mathematische Modellierung des physikalischen Problems falsch ist, d.h. schlecht oder nicht exakt vorgenommen wurde, sondern die schlechte Problemstellung eine natürliche Eigenschaft des mathematischen Modells ist.
In diesem Sinn ist das oben beschriebene Identifikationsproblem schlecht gestellt, weil im Allgemeinen weder die Existenz (Surjektivität), Eindeutigkeit (Injektivität) noch die Stetigkeit des Umkehroperators gilt. Hierzu einfache Beispiele

Beispiel 1: Problem besitzt nicht immer eine Lösung.
Hat der gemessene Referenzdatensatz eine Struktur (etwa ein Quadrat) mit Grauwert g_1 und der zweite Datensatz die gleiche Struktur mit Grauwert $g_2 \neq g_1$, dann gibt es kein $u \in \mathcal{X}$, so dass $T(x-u) = R(x)$. Das heißt, das Problem (2.1) ist in diesem Fall nicht lösbar.

Beispiel 2: Problem besitzt nicht immer eine eindeutige Lösung.
Wie in Beispiel (1) hat der gemessene Referenzdatensatz ein Quadrat mit Grauwert g_1. Der Templatedatensatz sei der um 45^o gedrehte Referenzdatensatz. Hier gibt es neben der Möglichkeit, den Datensatz zurück zu drehen, auch die Möglichkeit, ihn um 45^o vor zu drehen. Beide Lösungen bringen die Datensätze zur Deckung.

Um eine geeignete Näherungslösung des Problems finden zu können, wird der Begriff der Lösung verallgemeinert. Hierzu wird mit

$$h(u) := T(x - u(x)) - R(x) = F(u(x)) - R(x)$$

der $L_2(\Omega)$-Abstand

$$D(u) = \langle h(u), h(u) \rangle = ||h(u)||^2_{L_2(\Omega)} = \int_\Omega h(u)^2 \, d\Omega \tag{2.2}$$

zwischen den beiden Datensätzen gemessen und eine Lösung des Minimierungsproblems

$$\min_{u \in \mathcal{X}} D(u) \tag{2.3}$$

gesucht. Im Allgemeinen existiert kein Minimalelement u_{min} des Problems (2.3). Da das Defektfunktional nicht negativ wird, existiert jedoch ein Infimum

$$\inf_{u \in \mathcal{X}} D(u) = \inf_{u \in \mathcal{X}} ||F(u) - R(x)||_2^2 \geq 0$$

als untere Grenze aller zulässigen Defektnormen. Somit existieren zum vorgegebenen Wert $\eta > 0$ stets Elemente u^η_{min} mit

$$D(u^\eta_{min}) \leq \inf_{u \in \mathcal{X}} D(u) + \eta.$$

Die Lösung der Zustandsgleichung (2.1) ist im Allgemeinen nicht eindeutig. Deshalb macht es Sinn, durch Vorgabe eines Elements u^* aus dem Parameterraum $\mathcal{X}$ nach speziell ausgezeichneten Lösungen zu suchen. Für nichtlineare Probleme wird ein weiteres Kriterium eingeführt, das die Menge der Minimallösungen einschränken soll und im günstigsten Fall sogar eindeutig gestaltet. In [14] wird das Element $u^\dagger \in \mathcal{X}$

als u^*-Minimum-Norm-Lösung der Zustandsgleichung (2.1) bezeichnet, wenn für ein gegebenes Referenzelement $u^* \in \mathcal{X}$

$$||u^\dagger - u^*||_{\mathcal{X}} = \min \left\{ ||u - u^*||_{\mathcal{X}} \quad \middle| \quad F(u(x)) = R(x) \right\}$$

gilt. Hierdurch wird die Auswahl des Minimalelements gesteuert. Das Element u^* dient hierbei als Auswahlkriterium für die gesuchte Lösung.

Bei einer solchen inversen Problemstellung ist die Korrektheit der Datensätze von grundlegender Bedeutung. In der Zustandsgleichung (2.1) ist die Problemstellung idealisiert, da von einer exakten rechten Seite $R(x)$ ausgegangen wird.

In der Praxis können jedoch während der Messung (Bildaufnahme) die Datensätze gestört werden (Messfehler) oder zusätzlich beim Vergleich zweier Datensätze aus unterschiedlichen bildgebenden Verfahren ganz anders interpretiert werden (Modellfehler). Hinzu kommen naturgemäß Fehler bei der numerischen Berechnung der Lösung (Verfahrensfehler).

Bei vielen inversen Problemstellungen ist das Datenfehlerniveau $\delta > 0$ bekannt. Bei den in dieser Arbeit verwendeten Daten ist diese Größe jedoch quantitativ nicht zu erfassen. Weil man bei der Verallgemeinerung des Lösungsbegriffs durch die Minimierung der Defektnorm ohnehin nicht davon ausgeht, dass zu vorgegebenem Datensatz $R(x)$ die Lösung $u(x)$ ein exaktes Bild des Funktionals F ist, wird zur Berechnung der Lösung $R(x)$ als exakt angenommen. Um jedoch die Mess- und Modellfehler in der Praxis zu reduzieren, werden die Datensätze durch Bildverarbeitungsalgorithmen (Grauwertanpassungen, lineare- und nichtlineare Filter) so gut wie möglich vorverarbeitet.

Idealerweise würde man nach den Modifikationen der Problemstellung erwarten, dass jeder Parameterwert eindeutig aus dem zugehörigen Datensatz $R(x)$ rekonstruierbar ist. Da aber schon kleinste Mess- oder Modellfehler zu Datensätzen $R(x)$ führen können, die von weit entfernten Parametern erzeugt werden (vgl. Beispiel 3), ergibt sich die Forderung, dass leicht gestörte Daten nur zu leicht gestörten Lösungen führen.

Beispiel 3: Instabilität des Minimierungsproblems.

Gegeben seien zwei übereinanderliegende Quadrate mit numerischen Grauwerten $g_1,\ g_2 \in [0,1]$, so dass $u = 0$ eine Lösung des Minimierungsproblems (2.3)

ist. Dann können schon kleinste Störungen der Grauwerte g_2 dazu führen, dass die Lösung weit von der ursprünglichen Lösung entfernt liegt.

Als Ausweg wird im nächsten Abschnitt die Tikhonov-Regularisierung vorgestellt.

2.3 Regularisierung inverser Probleme

Durch die Methode der Regularisierung können schlecht gestellte inverse Probleme näherungsweise stabil gelöst werden. Einen Überblick über Regularisierungsmethoden für inverse Probleme findet man in [14] und [26]. Die in dieser Arbeit verwendeten Verfahren zur numerischen Lösung des inversen Problems werden durch die von Tikhonov Anfang der sechziger Jahre publizierte Methode regularisiert und im folgenden Abschnitt erläutert.

2.3.1 Motivation der Tikhonov-Regularisierung

Im Allgemeinen ist die Identifikationsaufgabe nichts anderes als die Minimierung der Defektnorm $D(u(x))$ auf der Parametermenge $\mathcal{X}$. Bei der Lösung des so gestellten nichtlinearen Minimierungsproblems (2.3) kann es, wie im vorherigen Abschnitt beschrieben wurde, schon bei kleinsten Mess- oder Modellfehlern zu Lösungen kommen, die weit von der eigentlich richtigen Lösung entfernt liegen. Zusätzlich sind bei dieser Problemstellung Aussagen über die Existenz und Eindeutigkeit einer Lösung nur unter zusätzlichen Bedingungen möglich.
Dieser Instabilität begegnet man bei vielen inversen Problemen durch die Methode der Regularisierung. Bei dem klassischen Regularisierungsverfahren von Tikhonov (Tikhonov,[32], [33] und [34]) wird das regularisierte Problem

$$\min_{u \in \mathcal{X}} T_\alpha(u) \tag{2.4}$$

mit dem sogenannten Tikhonov-Funktional

$$T_\alpha(u) := D(u) \; + \; \alpha G(u)$$

mit einem Glättungs- oder Strafterm G betrachtet. Hierbei wird dem zu minimierenden Funktional $D(u)$ ein Regularisierungsterm $G(u)$ mit geeigneten Eigenschaften

hinzugefügt. Tikhonov benutzte ursprünglich die kanonische Norm $||\cdot||_{\mathcal{X}}$ für den Parameterraum $\mathcal{X}$. In vielen Anwendungen wird die kanonische Norm jedoch durch eine geeignete Energienorm (vgl. Abschnitt 2.3.2) ersetzt.

Der Wahl des Regularisierungsparameters $\alpha \in I\!R^{>0}$ kommt hierbei eine fundamentale Bedeutung zu. Große Werte von α unterdrücken Datenfehler und geben dem Regularisierungsterm (Glättungsterm) mehr Gewicht. Umgekehrt gewinnt bei kleinen Werten von α die Fehlerfunktion $D(u)$ an Einfluss, so dass kleinste Datenfehler große Auswirkungen auf die resultierende Lösung haben.
Die Funktionale G und D stehen für zwei verschiedene Regularisierungsparameter in folgender Beziehung:

SATZ 2.3.1. *(Monotonie)*
Angenommen für $\alpha_2 > \alpha_1 > 0$ können Lösungen u_{α_1} und u_{α_2} von (2.4) berechnet werden, dann gilt

$$G(u_{\alpha_2}) \leq G(u_{\alpha_1}) \quad \textit{und} \quad D(u_{\alpha_1}) \leq D(u_{\alpha_2}).$$

Beweis. Für die Funktionale T_{α_1} und T_{α_2} gilt:

$$T_{\alpha_1}(u_{\alpha_1}) = D(u_{\alpha_1}) + \alpha_1\, G(u_{\alpha_1}) \leq D(u_{\alpha_2}) + \alpha_1\, G(u_{\alpha_2}) = T_{\alpha_1}(u_{\alpha_2}) \tag{2.5}$$

$$T_{\alpha_2}(u_{\alpha_2}) = D(u_{\alpha_2}) + \alpha_2\, G(u_{\alpha_2}) \leq D(u_{\alpha_1}) + \alpha_2\, G(u_{\alpha_1}) = T_{\alpha_2}(u_{\alpha_1}). \tag{2.6}$$

Mit (2.6) und (2.5) folgt

$$(\alpha_2 - \alpha_1)G(u_{\alpha_2}) \leq (\alpha_2 - \alpha_1)G(u_{\alpha_1}),$$

also $G(u_{\alpha_2}) \leq G(u_{\alpha_1})$. Aus Gleichung (2.5) folgt sofort $D(u_{\alpha_1}) \leq D(u_{\alpha_2})$. □

In der Praxis ist, außer für spezielle Datensätze, die Ermittlung einer globalen Minimallösung von $D(u)$, wegen der Unstetigkeit vieler Lösungen, weder erreichbar noch erwünscht. Deshalb geht man dazu über eine geeignete Minimallösung eines Tikhonov-Funktionals näherungsweise so zu bestimmen, dass sie der tatsächlichen Deformation sehr nahe kommt. Hierfür ist es zum einen wichtig den Regularisierungsparameter geeignet zu wählen und zum andern wegen des enormen Rechenaufwands effektive Algorithmen zur Berechnung der Lösungen zu entwickeln.

2.3.2 Regularisierung durch eine Bilinearform

Bei vielen Problemstellungen in Medizin, Technik und Naturwissenschaften (z.B. in [1], [2], [3], [9], [28] und [31]) werden anstelle der klassischen Tikhonov-Regularisierung andere, der Problemstellung entsprechende Glättungsterme G verwendet. Hierbei wird der durch die kanonische Norm $||\cdot||_{\mathcal{X}}$ definierte Glättungsterm $||u||_{\mathcal{X}}^2$ in T_α durch die Energienorm $||u||_e := a(u,u)^{1/2}$, mit einer Bilinearform $a(u,u)$, ersetzt. Dies führt zur Minimierung des Funktionals

$$T_\alpha(u) = D(u) + \alpha||u||_e^2 = D(u) + \alpha a(u,u). \tag{2.7}$$

Um die elastische Deformationen eines Körpers (z.B. menschlicher Organe) zu modellieren, wird die folgende symmetrische, positiv semidefinite Bilinearform

$$a(u,v) := \int_\Omega \sum_{i,j=1}^{d} (\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i})(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i})\, d\Omega, \tag{2.8}$$

eingeführt. Die Bilinearform a misst die potentielle Energie einer elastischen Verschiebung (vgl. [37]) und modelliert einen Körper wie ein elastisches Medium ohne Querkontraktion.

2.4 Erweiterte Problemstellung

In Kapitel 2.2 wurde eine in der aktuellen medizinischen Forschung relevante Aufgabenstellung als Identifikationsproblem formuliert. Identifikationsprobleme treten in vielen Bereichen der Naturwissenschaften, Medizin und Technik auf. Das Ziel einer Identifikationsaufgabe besteht in der Identifikation eines oder mehrerer Parameter, die nicht direkt messbar sind, aus anderen gegebenen Größen. Diese Größen sind in dieser Arbeit digitale Bilddaten. In der medizinischen Aufgabenstellung sind die Ein- und Ausgabedaten Schnittbilder innerer Organe (in dieser Arbeit Gehirne), aus denen Parameter berechnet werden sollen, die das Defektfunktional $D(u)$ möglichst minimieren. Die gesuchten Parameter sind hierbei entweder die bei der Herstellung histologischer Serienschnitte erzeugten Deformationen oder die Verschiebung zur Überlagerung interindividueller Bilddaten.

In Kooperation mit Herrn Holmlund der Firma Eumetsat wurde die Aufgabenstellung auf ein Problem aus der Meteorologie übertragen. Wichtige Parameter bei der

kurz- und mittelfristigen Wettervorhersage sind die Windrichtung und die Windgeschwindigkeit, zu deren zuverlässigen Bestimmung es bisher noch keine geeigneten Verfahren gibt. Verfügbar sind jedoch von einem Satelliten aufgenommene Wolkenbilder zu unterschiedlichen Zeitpunkten, die bei der Formulierung der Aufgabe als Identifikationsproblem als Ein- bzw. Ausgabedaten dienen. Die gesuchten Parameter in dieser Modellierung sind die Windrichtung und Windgeschwindigkeit, auf die durch geeignete Verfahren geschlossen werden soll.

Auf der Suche nach geeigneten Lösungen wird, wie in Kapitel 2.2 beschrieben, das Defektfunktional $D(u)$ minimiert. Hiermit sollen zwei zu unterschiedlichen Zeiten aufgenommene Wolkenbilder zur Deckung gebracht werden. Aus der Zeitdifferenz und den Verschiebungen für die einzelnen Bildpunkte wird anschließend in geeigneter Weise auf die Richtung und die Geschwindigkeit des Windes zurückgerechnet.

Kapitel 3

Unrestringierte Minimierungsverfahren für das Defektfunktional

Die in diesem Abschnitt vorgestellten Verfahren erzeugen iterativ Folgen $\{u^{(k)}\}_{k\in I\!N} \subset \mathcal{X}$, die mit wachsendem Index k das Defektfunktional $D(u)$ immer weiter reduzieren. Das bedeutet, dass die Abstiegsbedingung

$$D(u^{(k+1)}) < D(u^{(k)}) \quad \text{für} \quad k = 0, 1, ...$$

für nichtoptimale $u^{(k)}$ gesichert wird.

3.1 Allgemeine Abstiegsverfahren zur Lösung des Identifikationsproblems

3.1.1 Verfahren des steilsten Abstiegs / Landweber-Iteration

Ein erster Ansatz zur Lösung des nichtlinearen Minimierungsproblems (2.3) ist die Taylorentwicklung erster Ordnung des Defektfunktionals D um die Verschiebung u

$$D(u + w) \approx D(u) + (\nabla D(u), w). \tag{3.1}$$

Gesucht ist also ein $w \in \mathcal{X}$, so dass $D(u+tw) < D(u)$ mit einem Parameter $t > 0$. Aus (3.1) folgt hiermit direkt $(\nabla D(u), w) < 0$. Für die Richtung des steilsten Abstiegs $w = -\nabla D(u)$ gilt

$$(\nabla D(u), -\nabla D(u)) = -||\nabla D(u)||^2 < 0$$

und man erhält ein iteratives Verfahren. Hierbei bestimmt man im k-ten Iterationsschritt die nächste Approximation $u^{(k+1)}$ als

$$u^{(k+1)} := u^{(k)} + t_k \cdot \nabla D(u^{(k)}) \tag{3.2}$$

mit Relaxationsparameter t_k. Im Kontext mit inversen Problemen wird das Verfahren des steilsten Abstiegs (steepest decent) (3.2) bei Anwendung auf least-squares Funktionale auch Landweber-Iteration genannt [14].

3.1.2 Verfahren zweiter Ordnung zur Minimierung des Defektfunktionals

Das Newton Verfahren zur Minimierung des Defektfunktionals

Das Newton Verfahren zur Minimierung des Defektfunktionals basiert auf der Taylorentwicklung zweiter Ordnung von $D(u^{(k)})$ an der Stelle $u^{(k)} \in \mathcal{X}$:

$$D(u^{(k)} + w) \approx D(u^{(k)}) + (J_D(u^{(k)}), w) + \frac{1}{2} w^T H(u^{(k)}) w \tag{3.3}$$

mit $w \in \mathcal{X}$ und der Hesse Matrix $H(u)$ an der Stelle $u^{(k)}$. Das Minimum der rechten Seite von (3.3) ist Minimum des quadratischen Funktionals

$$g(w) = (J_D(u^{(k)}), w) + \frac{1}{2} w^T H(u^{(k)}) w. \tag{3.4}$$

Ein stationärer Punkt $w^{(k)}$ von Gleichung (3.4) ist Lösung des linearen Systems

$$H(u^{(k)}) w = -J_D(u^{(k)}). \tag{3.5}$$

w wird Newton-Richtung genannt, und die iterative Anwendung ergibt das Newton-Verfahren.

Das Gauß-Newton Verfahren zur Minimierung des Defektfunktionals

Der Gradient J_D und die Hessematrix H des least-squares Defektfunktionals $D(u)$ haben die folgende spezielle Struktur

$$J_D(u) = 2 \cdot J_h^T(u) \cdot h(u) \tag{3.6}$$

$$H(u) = J_h^T(u) J_h(u) + Q(u), \tag{3.7}$$

mit der Zerlegung der Hessematrix in $J_h^T(u)J_h(u)$ und einen quadratischen Anteil $Q(u)$. Eingesetzt in Gleichung (3.5) erhält man die Newton Richtung

$$\left(J_h^T(u^{(k)})J_h(u^{(k)}) + Q(u^{(k)}) \right) w = -J_D(u^{(k)}). \tag{3.8}$$

Durch das Vernachlässigen des quadratischen Anteils Q der Hesse Matrix H erhält man das Gauß-Newton Verfahren

$$J_h^T(u^{(k)})J_h(u^{(k)})w = -J_D(u^{(k)}). \tag{3.9}$$

Die Gauß-Newton Abstiegsrichtung ergibt sich als Lösung des linearen Ausgleichsproblems

$$\min_{w \in \mathcal{X}} ||J_h(u^{(k)})w + h(u^{(k)})||_2^2.$$

Die iterative Anwendung ergibt die Gauß-Newton Iteration.

3.2 Regularisierte Abstiegsverfahren

In diesem Abschnitt werden zwei iterative Verfahren zur Minimierung des Defektfunktionals $D(u)$ vorgestellt, in denen zusätzlich die Abstiegsrichtungen regularisiert werden.

3.2.1 Das Levenberg-Marquardt Verfahren

Die Levenberg-Marquardt (LM) Iteration ist eine Variante der Gauß-Newton Iteration zur Minimierung des Defektfunktionals $D(u)$. Hierbei wird die Linearisierung von $h(u)$ um eine aktuelle Näherungslösung $u^{(k)}$ betrachtet. Mit

$$h(u^{(k)} + w) = h(u^{(k)}) + J_h(u^{(k)}) \cdot w + \mathcal{O}(w^2)$$

erhält man eingesetzt in Darstellung (2.2) das quadratische Funktional

$$I_k(w) = ||h(u^{(k)}) + J_h(u^{(k)}) \cdot w||_{L_2(\Omega)}^2.$$

Durch die Regularisierung der Abstiegsrichtung w erhält man das Tikhonov-Funktional

$$\overline{I}_k(w) = ||h(u^{(k)}) + J_h(u^{(k)}) \cdot w||_{L_2(\Omega)}^2 + \alpha_k ||w||_{\mathcal{X}}^2 \tag{3.10}$$

mit dem Regularisierungsparameter $\alpha_k > 0$. Da $\overline{I}_k(w)$ ein quadratisches Funktional ist, kann die Minimallösung $w = w^{(k)}$ über die entsprechende Normalengleichung

$$w^{(k)} = (J_h^T(u^{(k)})J_h(u^{(k)}) + \alpha_k I)^{-1} J_h^T(u^{(k)})h(u^{(k)})$$

bestimmt werden. Das entspricht der Gauß-Newton Abstiegsrichtung (3.8), bei der der quadratische Anteil der Hesse-Matrix $Q(u)$ durch ein Vielfaches der Einheitsmatrix $\alpha_k I$ ersetzt wird. Für $\alpha_k \to \infty$ ist $w^{(k)}$ die Richtung des steilsten Abstiegs, für $\alpha_k = 0$ entspricht $w^{(k)}$ der Gauß-Newton Abstiegsrichtung.

Die Regularisierung von (3.10) durch die Bilinearform $a(\cdot,\cdot)$, gemäß Kapitel 2.3.2, führt auf die Minimierung des Funktionals

$$\begin{aligned}
\hat{I}_k(w) &= ||h(u^{(k)}) + J_h(u^{(k)}) \cdot w||^2_{L_2(\Omega)} + \alpha_k a(w,w) \\
&= \langle c_1 + c_2 \cdot w, c_1 + c_2 \cdot w\rangle + \alpha_k a(w,w) \\
&= \underbrace{\alpha_k a(w,w) + \langle c_2 w, c_2 w\rangle}_{=:b_{\alpha_k}(w,w)} + \langle c_1, c_1\rangle + 2 \cdot \underbrace{\langle c_1, c_2 \cdot w\rangle}_{=:-l(w)} \\
&= b_{\alpha_k}(w,w) + \langle c_1, c_1\rangle - 2 \cdot l(w)
\end{aligned}$$

nach w, wobei

$$c_1 := h(u^{(k)}) \quad \text{und} \quad c_2 := J_h(u^{(k)}).$$

Gesucht ist also ein $w \in \mathcal{X}$, so dass

$$\hat{I}_k(w) = \min_{v \in \mathcal{X}} \hat{I}_k(v). \tag{3.11}$$

Nimmt $\hat{I}_k$ sein Minimum für ein w^* an, so ist mit beliebigem $\varphi \in \mathcal{X}$ und $t \in I\!R$ stets

$$\hat{I}_k(w^* + t \cdot \varphi) \geq \hat{I}_k(w^*).$$

$\hat{I}_k(w^* + t \cdot \varphi)$ ist eine quadratische Funktion in t also differenzierbar und nimmt als Funktion sein Minimum für $t = 0$ an. Notwendige Bedingung für eine Minimalstelle ist das Verschwinden der sogenannten ersten Variation:

$$\frac{\partial}{\partial t} \hat{I}_k(u^* + t \cdot \varphi)|_{t=0} = 0.$$

Die Bilinearform $b_{\alpha_k}(\cdot,\cdot)$ ist symmetrisch, positiv und semidefinit, deshalb ist die Lösung der Variationsgleichung

$$b_{\alpha_k}(w,\varphi) = l(\varphi) = \langle c_2^T \cdot c_1, \varphi\rangle_{L_2(\Omega)} \quad \forall \varphi \in \mathcal{X} \tag{3.12}$$

äquivalent zur Lösung des Minimierungsproblems in Gleichung (3.11). Für

$$\mathcal{X} = \left(H_0^1(\Omega)\right)^n = \underbrace{H_0^1(\Omega) \times \cdots \times H_0^1(\Omega)}_{n-mal}$$

ist das die schwache Formulierung des folgenden Randwertproblems mit Dirichlet-Randbedingungen

$$\begin{array}{rcll} \alpha_k(-\Delta w(x) - \nabla(\nabla w(x))) + c_2 \cdot w(x) \cdot c_2^T & = & f(u^{(k)}(x)) & \text{für } x \in \Omega, \\ w(x) & = & 0 & \text{für } x \in \partial\Omega \end{array} \tag{3.13}$$

wobei $f(u^{(k)}) = c_2^T \cdot c_1 = J_h(u^{(k)}) \cdot h(u^{(k)})$. Für den Term $c_2 \cdot w(x) \cdot c_2^T$ gilt hierbei

$$\begin{aligned} c_2 \cdot w(x) \cdot c_2^T = J_h(u^{(k)}) \cdot w(x) \cdot J_h(u^{(k)})^T &= \\ \nabla T(x - u^{(k)}) \cdot w(x) \cdot \nabla T(x - u^{(k)})^T &= \\ (T_x \cdot T_x w_1 + T_x \cdot T_y w_2, T_x \cdot T_y w_1 + T_y \cdot T_y w_2)^T. & \end{aligned}$$

Im zweidimensionalen Fall erhält man die lineare gekoppelte partielle Differentialgleichung:

$$\begin{pmatrix} \alpha_k(-\Delta - \frac{\partial}{\partial x \partial x}) + T_x \cdot T_x & -\alpha_k \frac{\partial}{\partial x \partial y} + T_x \cdot T_y \\ -\alpha_k \frac{\partial}{\partial x \partial y} + T_x \cdot T_y & \alpha_k(-\Delta - \frac{\partial}{\partial y \partial y}) + T_y \cdot T_y \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} = \begin{pmatrix} f_1(u^{(k)}(x)) \\ f_2(u^{(k)}(x)) \end{pmatrix}. \tag{3.14}$$

Auf eine ähnliche Gleichung zur Minimierung eines Funktionals zur Berechnung des optischen Flusses kamen (HORN UND SCHNUCK, [24]). Unglücklicherweise ist die numerische Lösung der Gleichung durch ein iteratives Verfahren problematisch. Die Koeffizienten der Funktion w der linken Seite von (3.14) können abhängig von den Werten von T sehr viel kleiner oder auch sehr viel größer als null sein.
Dies zerstört im Allgemeinen die Diagonaldominanz der Diskretisierungsmatrix (vgl. Kapitel 4.1.2) von Gleichung (3.14) und gefährdet damit die Konvergenz der in Kapitel 4 vorgestellten iterativen Lösungsverfahren. Um die Diagonaldominanz zu erhalten, müsste der Parameter α_k entsprechend groß gewählt werden. Hierdurch wird jedoch die Länge der Abstiegsrichtung im Defektfunktional $D(u)$ entsprechend klein, so dass der Wert von $D(u)$ kaum reduziert wird. Deshalb wird im folgenden Abschnitt ein auf dem Verfahren des steilsten Abstiegs basierendes Verfahren zur Minimierung des Defektfunktionals $D(u)$ betrachtet.

3.2.2 Das regularisierte Landweber-Verfahren

Die Linearisierung des LM Verfahrens erfolgte durch Linearisierung von $h(u)$ um eine aktuelle Näherung $u^{(k)}$. Im Folgenden wird $D(u)$ durch Linearisierung um eine gegebene aktuelle Lösung $u^{(k)}$ ersetzt:

$$D(u^{(k)} + v) = D(u^{(k)}) + \langle J_D(u^{(k)}), v\rangle_{L_2(\Omega)} + \mathcal{O}(v^2).$$

Durch Regularisierung der Abstiegsrichtung v durch die symmetrisch positiv semidefinite Bilinearform a gemäß Gleichung (2.8) erhält man

$$I_k(v) := \langle J_D(u^{(k)}), v\rangle_{L_2(\Omega)} + \alpha_k a(v, v).$$

Mit der Einheitsmatix I gilt

$$\begin{aligned} J_D(u^{(k)}) &= 2 \cdot J_h(u^{(k)}) \cdot h(u^{(k)}) \\ &= 2 \cdot \underbrace{J_T(\phi(x, u^{(k)}))}_{=\nabla T(\phi(x,u^{(k)}))} \cdot \underbrace{J_\phi(u^{(k)})}_{=-I} \cdot h(u^{(k)}) = -2 \cdot \nabla T(\phi(x, u^{(k)})) \cdot h(u^{(k)}). \end{aligned}$$

Gesucht ist also ein $v \in \mathcal{X}$, so dass

$$I_k(v) = \min_{w \in \mathcal{X}} I_k(w). \tag{3.15}$$

Weil $a(\cdot, \cdot)$ eine symmetrisch positiv semidefinite Bilinearform ist, ist die Lösung der linearen Variationsgleichung

$$a(v, \varphi) = \underbrace{\langle -\frac{2}{\alpha_k} J_h(u^{(k)}) \cdot h(u^{(k)}), \varphi\rangle_{L_2(\Omega)}}_{=: f_{\alpha_k}(u^{(k)})} \quad \forall \varphi \in (H_0^1(\Omega))^2$$

äquivalent zur Minimallösung von $I_k(v)$. Das ist die schwache Form des Randwertproblems:

$$\begin{aligned} -\Delta v(x) - \nabla(\nabla v(x)) &= f_{\alpha_k}(u^{(k)}(x)) \quad \text{für} \quad x \in \Omega \\ v(x) &= 0 \qquad\qquad\quad \text{für} \quad x \in \partial\Omega. \end{aligned} \tag{3.16}$$

Im zweidimensionalen Fall erhält man das lineare gekoppelte partielle Differentialgleichungssystem

$$\begin{pmatrix} -\Delta - \frac{\partial}{\partial x \partial x} & -\frac{\partial}{\partial x \partial y} \\ -\frac{\partial}{\partial x \partial y} & -\Delta - \frac{\partial}{\partial y \partial y} \end{pmatrix} \begin{pmatrix} v_1(x) \\ v_2(x) \end{pmatrix} = \begin{pmatrix} f_{\alpha_k,1}(u^{(k)}(x)) \\ f_{\alpha_k,2}(u^{(k)}(x)) \end{pmatrix}. \tag{3.17}$$

3.3 Variationsansatz zur Minimierung des Tikhonov-Funktionals

Neben der Möglichkeit, den Wert des Defektfunktionals durch einen linearisierten Ansatz zu reduzieren, existiert die Möglichkeit, direkt das Tikhonov-Funktional

$$T_\alpha(u) = D(u) + \alpha G(u)$$

bzgl. u zu minimieren. Das Tikhonov-Funktional ist nichtlinear, so dass im Allgemeinen die Frage nach der Existenz und Eindeutigkeit einer Lösung offen ist. Aussagen über die Eindeutigkeit einer Minimallösung sind, wie für die nicht regularisierte Problemstellung (2.3), bei einem nicht konvexen Defektfunktional $D(u)$ nicht zu formulieren. Durch eine hinreichend große Wahl des Regularisierungsparameters α kann bei einem streng konvexen Funktional G die Eindeutigkeit der Lösung von (2.4) erzwungen werden. Hierbei wird jedoch im Allgemeinen dem Funktional T_α durch zu starke Gewichtung des Regularisierungsterms G dessen Konvexität aufgezwungen. Dies hat zur Folge, dass die charakteristischen Eigenschaften des ursprünglich zu minimierenden Funktionals D weitgehend verloren gehen.

Notwendige Bedingung für eine Minimallösung

Zur Herleitung einer notwendigen Bedingung für eine Minimalstelle von T_α wird die Einführung der Gâtaux-Ableitung eines Funktionals $\mathcal{F}$

$$\mathcal{F}_u(v) = \lim_{t \to 0} \frac{\mathcal{F}(u + tv) - \mathcal{F}(u)}{t}$$

notwendig.

SATZ 3.3.1. *(Notwendige Bedingung einer Minimallösung)*
Die Bilinarform a sei symmetrisch und positiv semidefinit. $D_u(v)$ sei die Gâteaux-Ableitung des Defektfunktionals $D(u)$. Dann sind die Lösungen des Minimierungsproblems

$$T_\alpha(u) = D(u) + \alpha a(u,u) \to min \quad \forall u \in \mathcal{X} \tag{3.18}$$

auch Lösungen der Variationsgleichung

$$\alpha a(u,\varphi) + \frac{1}{2} D_u(\varphi) = 0 \quad \forall \varphi \in \mathcal{X}. \tag{3.19}$$

Beweis. Sei u^* eine Lösung von (3.18) und $\varphi \in \mathcal{X}$ beliebig aber fest gewählt, dann gilt

$$T_\alpha(u^* + t\varphi) \geq T_\alpha(u^*) \quad \forall t \in I\!R. \tag{3.20}$$

Da $a(\cdot,\cdot)$ eine symmetrische Bilinearform ist, gilt

$$a(u + tw, u + tw) = a(u,u) + 2ta(u,w) + t^2 a(w,w).$$

Mit $t > 0$ und (3.20) gilt

$$\begin{aligned} \frac{T_\alpha(u^* + t\varphi) - T_\alpha(u^*)}{t} &= \frac{\alpha a(u^* + t\varphi, u^* + t\varphi) - \alpha a(u^*, u^*) + D(u^* + t\varphi) - D(u^*)}{t} \\ &= 2 \cdot \alpha a(u^*, \varphi) + \frac{D(u^* + t\varphi) - D(u^*)}{t} + \mathcal{O}(t) \geq 0. \end{aligned}$$

Für $t \to 0$ folgt also

$$\alpha a(u^*, \varphi) + \frac{1}{2} D_{u^*}(\varphi) \geq 0,$$

das gilt für $\pm\varphi$. Daraus folgt Gleichung (3.19) und damit die Behauptung. □

Die Variationsgleichung (3.19) ist mit

$$D_u(\varphi) = \langle \underbrace{J_h(u(x)) \cdot h(u(x))}_{=: f(u(x))}, \varphi \rangle$$

die schwache Form der folgenden nichtlinearen Randwertaufgabe mit Dirichlet-Randbedingungen

$$E(u(x)) = \begin{cases} \alpha(-\Delta u(x) - \nabla(\nabla u(x))) - f(u(x)) &= 0 \quad \text{für} \quad x \in \Omega \\ u(x) &= 0 \quad \text{für} \quad x \in \partial\Omega. \end{cases} \tag{3.21}$$

Im zweidimensionalen Fall erhält man für das Einheitsquadrat $\Omega = (0,1) \times (0,1)$ die nichtlineare gekoppelte partielle Differentialgleichung

$$\alpha \begin{pmatrix} -\Delta - \frac{\partial}{\partial x \partial x} & -\frac{\partial}{\partial x \partial y} \\ -\frac{\partial}{\partial x \partial y} & -\Delta - \frac{\partial}{\partial y \partial y} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} - \begin{pmatrix} f_1(u_1, u_2) \\ f_2(u_1, u_2) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}. \tag{3.22}$$

Existenz und Eindeutigkeit einer Minimallösung

Die Eindeutigkeit einer Lösung ist nur gesichert, wenn das Defektfunktional $D(u)$ ein schwach konvexes Funktional ist, d.h. für beliebige $u, v \in \mathcal{X}$ gilt:

$$D((1-t)u + tv) \leq (1-t)D(u) + tD(v) \quad \forall t \in [0,1].$$

Mit der strengen Konvexität des Energieterms $a(u,u)$ folgt hieraus die strenge Konvexität des Tikhonov-Funktionals T_α und deshalb für Minimallösungen u und v von T_α mit $T_\alpha(u) = T_\alpha(v) = c$

$$T_\alpha((1-t)u + tv) < (1-t)c + tc = c$$

mit Widerspruch dazu, dass c Minimum ist. Das Defektfunktional ist in der Regel hochgradig nicht konvex. Deshalb kann man im Allgemeinen nicht mit einer eindeutigen Minimallösung des Funktionals T_α rechnen.
Die Betrachtung kontinuierlicher Funktionale wird hier nicht fortgesetzt, weil die Lösungen in der Praxis auf einem endlich dimensionalen Raum H_h (vgl. Kapitel 4) mit dem diskreten Funktional $T_\alpha^h : H_h \times H_h \to \mathbb{R}^{\geq 0}$ berechnet werden. Die diskreten Funktionale sind positiv und stetig. Darüber hinaus gilt $T_\alpha^h(u) \to \infty$ falls $||u||_{H_h} \to \infty$. Die Existenz mindestens einer Minimallösung für das diskrete Funktional folgt hieraus sofort.

Kapitel 4

Numerische Lösung der Randwertaufgaben

Auf der Suche nach einer Minimallösung des Defektfunktionals durch die im vorherigen Abschnitt beschriebenen linearen und nichtlinearen Ansätze ist es erforderlich, die entsprechenden linearen oder nichtlinearen Randwertprobleme zu lösen. Zur numerischen Lösung werden die Randwertprobleme in diesem Kapitel zunächst geeignet diskretisiert. Die Approximation der Differentialoperatoren erfolgt durch Differenzenoperatoren zweiter Ordnung. Abhängig von der Anzahl der Bildelemente der digitalisierten Bilder führt diese Vorgehensweise auf sehr große Gleichungssysteme, die im Folgenden iterativ durch Mehrgitterverfahren gelöst werden.

4.1 Diskretisierung der Randwertaufgaben

Für die im vorherigen Kapitel beschriebenen Abstiegsverfahren wurden die Funktionen T, R, u und f als kontinuierlich angenommen. Für numerische Berechnungen müssen die Funktionen geeignet diskretisiert werden.

4.1.1 Uniforme Bildabtastung und Quantisierung der Grauwerte

Um kontinuierliche Bilder $B(x_1, x_2)$ in einer geeigneten Form darzustellen, so dass sie mit dem Computer weiterverarbeitet werden können, müssen sie räumlich abgetastet (Bildabtastung) und die Grauwerte durch eine endliche Skala ersetzt werden (Grauwert-Quantisierung).

Hierfür wird ein zweidimensionales kontinuierliches Bild $B(x_1, x_2)$ durch jeweils gleich große Bildzellen in der Form einer $N_1 \times N_2$ Bildmatrix:

$$B(x_1, x_2) \approx \begin{bmatrix} B(0,0) & B(0,1) & \cdots & B(0, N_2-1) \\ B(1,0) & B(1,1) & \cdots & B(1, N_2-1) \\ \vdots & \vdots & & \vdots \\ B(N_1-1,0) & B(N_1-1,1) & \cdots & B(N_1-1, N_2-1) \end{bmatrix} = B^{\delta}(x_1, x_2) \tag{4.1}$$

und ein dreidimensionales Bild $B(x_1, x_2, x_3)$ entsprechend in der Form einer $N_1 \times N_2 \times N_3$ Bildmatrix approximiert. Bei der Digitalisierung wird die Bildinformation an den Gitterpunkten $x_{i,j} = (ix_1, jx_2)$ abgetastet oder in einer Umgebung von $x_{i,j}$ gemittelt. Die Werte $B(x_1, x_2)$ entsprechen einer Grauwertquantisierung (vgl. z.B. [25]). Die rechte Seite von Gleichung (4.1) wird hierbei auch digitales Bild genannt, die Elemente der Bildmatrix werden im zweidimensionalen pixel (engl.: picture element) und im dreidimensionalen voxel (engl.: volume picture element) genannt.
Durch die beschriebene Abtastung des kontinuierlichen Bildsignals $B(x_1, x_2)$ und dessen Grauwertquantisierung entsteht das mit Fehlern behaftete diskrete Bildsignal $B^{\delta}(x_1, x_2)$. Die allgemeine Wirkung dieser Operationen ist eine Glättung des Bildes, hierbei gehen die feinen Details des Bildes verloren. Im Fourierraum führt diese Vorgehensweise zu einer Dämpfung hoher Wellenzahlen (Bandbegrenzung). Bei vielen Aufgaben im Bereich der digitalen Bildverarbeitung ist es nicht notwendig, auf der durch die Diskretisierung gegebenen Bildauflösung zu arbeiten. Im Folgenden werden zwei Techniken zur Reduzierung der Bildinformationen beschrieben, die in der Praxis zur Beschleunigung von Bildverarbeitungsalgorithmen verwendet werden.

Bildvergröberung

Die Bildvergröberung ist eine Einschränkung der Bildinformation durch Zusammenfassen mehrerer Bildelemente in ein Bildelement einer kleineren Bildmatrix. Dies geschieht für alle Bildelemente in gleicher Art und Weise, so dass diese Operation einer Glättung entspricht. Im Fourierraum entspricht dies der Unterdrückung hoher Wellenzahlen oder feinen Details (Tiefpassfilter). In der praktischen Anwendung werden Operationen auf vergröberten Bildern durchgeführt, um den Rechenaufwand zu reduzieren.

Nicht uniforme Bildabtastung

Eine weitere Möglichkeit zur Einschränkung der Bildinformation ist eine nicht äquidistante Bildabtastung. Hierbei kann die Größe der Bildelemente bei der oben beschriebenen Diskretisierung des kontinuierlichen Bildes abhängig von der Bildinformation geeignet gewählt werden. Diese Art der Bildvergröberung kann beispielsweise für große Bildteile mit gleichem Grauwert, etwa dem Hintergrund bei der Darstellung von Gehirnschnitten, genutzt werden.

4.1.2 Diskretisierung der Verschiebungsvektoren

Durch die uniforme Bildabtastung erhält man eine Zerlegung des quadratischen Bildbereichs in $N_1 \times N_2$ achsenparallele Teilgebiete. In den meisten Anwendungen der digitalen Bildverarbeitung ist hierbei $N_1 = N_2 = 2^k$ mit $k \in \mathbb{N}$. Die digitalen Bilddaten können hierdurch als stückweise konstante Funktionen aufgefasst werden.

Um die gesuchten Verschiebungen geeignet zu diskretisieren, wird der Wert für die Verschiebung eines d-dimensionalen Bildelements an der Stelle $(i_1, i_2, ..., i_d)$ in dessen Mitte gespeichert. Die so diskretisierten Verschiebungsvektoren sind ebenfalls stückweise konstante Funktionen, die auf den gleichen achsenparallelen Teilgebieten mit der Breite $h_1 = h_2 = 1/N_1 = 1/N_2$ definiert sind. Diese Art der Diskretisierung der Verschiebungsvektoren wird cell centered Diskretisierung (vgl. [38]) genannt, und die Teilgebiete der Verschiebungsvektoren werden als cell (Zelle) bezeichnet. Man erhält allgemein das folgende d-dimensionale Gitter

$$G_h = \left\{ x \in \mathbb{R}^d \middle| x = (j-s)h, j = \begin{pmatrix} j_1 \\ \vdots \\ j_d \end{pmatrix}, h = \begin{pmatrix} h_1 \\ \vdots \\ h_d \end{pmatrix}, s = \frac{1}{2}\begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}, \begin{array}{l} j_\alpha = 1, .., N_\alpha, \\ h_\alpha = 1/N_\alpha, \\ \alpha = 1, .., d \end{array} \right\},$$

in dessen Gitterpunkten die einzelnen Komponenten des Verschiebungsvektors für das entsprechende Bildelement abgespeichert werden. Die Menge der Gitterpunkte am Rand von G_h wird mit Γ_h bezeichnet. Hierbei ist formal zu beachten, dass die Punkte in Γ_h nicht auf dem Rand von Ω liegen, jedoch auf die Bildelemente am Rand von Ω_h zugreifen.

Für $d = 2$ wird die Diskretisierung der Verschiebungsvektoren in dieser Arbeit auch pixel centered Diskretisierung genannt (vgl. Abbildung 3.2 und 3.3). Es resultiert das

folgende zweidimensionale Gitter

$$G_h = \Big\{ x \in I\!R^2 : x = ((i-1/2)\cdot h, (j-1/2)\cdot h),\ i = 1,2,..,N_1,\ j = 1,2,..,N_2 \Big\}. \quad (4.2)$$

Man beachte, dass durch das Gitter G_h die Positionen der Mittelpunkte der verschiedenen Zellen zusammengefasst werden. Die Komponenten der Verschiebungen u wie auch die der rechten Seite f werden für die numerische Berechnung als Gitterfunktionen mit $N_1 \times \cdots \times N_d$ Gitterpunkten

$$\mathcal{F} : G_h \to I\!R^{N_1 \times \cdots \times N_d}$$

behandelt. D.h. für die zweidimensionale Problemstellung besteht der Verschiebungsvektor u und die rechte Seite f aus zwei Gitterfunktionen

$$\begin{array}{rcl} u_h(x_h) & = & (u_{1,h}, u_{2,h})^t \in \mathcal{F}(G_h) \times \mathcal{F}(G_h) \quad \text{bzw.} \\ f_h(x_h) & = & (f_{1,h}, f_{2,h})^t \in \mathcal{F}(G_h) \times \mathcal{F}(G_h). \end{array}$$

Im Folgenden werden vektorielle Gitterfunktionen mit der Dimension $d \geq 2$ durch

$$(\mathcal{F}(G_h))^d := \underbrace{\mathcal{F}(G_h) \times \cdots \times \mathcal{F}(G_h)}_{d-mal}$$

dargestellt. Die Diskretisierung der linearen partiellen Differentialgleichung (3.13) führt also auf

$$\begin{array}{rcll} -\Delta_h u_h - \nabla_h(\nabla_h u_h) & = f_h & \text{für} & (x,y) \in G_h \\ u_h & = 0 & \text{für} & (x,y) \in \Gamma_h \end{array}$$

und die Diskretisierung der nichtlinearen partiellen Differentialgleichung (3.22) auf

$$E_h(u_h) = \left\{ \begin{array}{rcl} -\Delta_h u_h - \nabla_h(\nabla_h u_h) - f_h(u_h) & = & 0 \quad \text{für} \quad (x,y) \in G_h \\ u_h & = & 0 \quad \text{für} \quad (x,y) \in \Gamma_h \end{array} \right.$$

mit den diskreten Differentialoperatoren Δ_h und ∇_h.

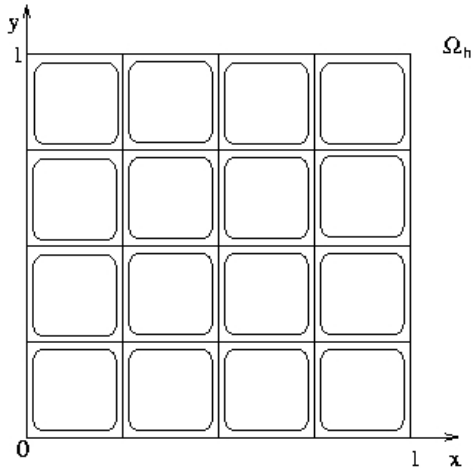


Abbildung 4.1: Bildzellen dargestellt im Gebiet $\Omega = [0,1]^2$.

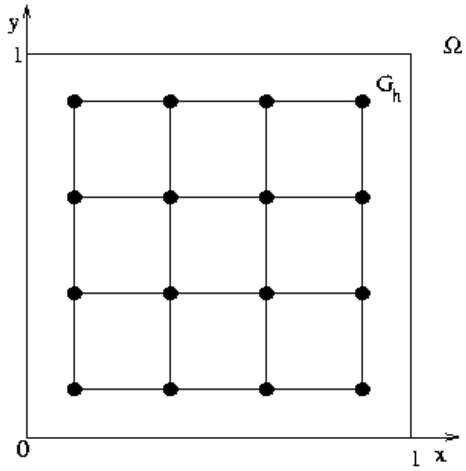


Abbildung 4.2: Diskretisierungsgitter Ω_h für die Verschiebungen u und die rechte Seite f.

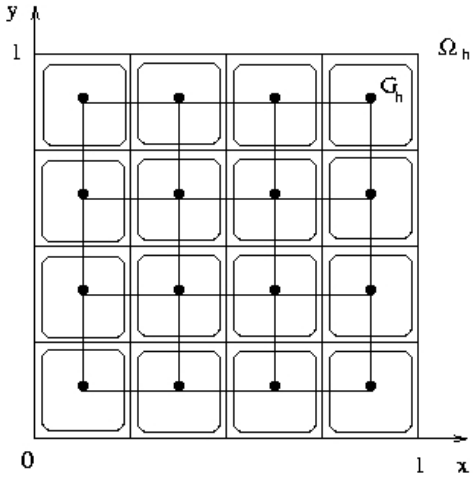


Abbildung 4.3: Lage des Gitters G_h.

4.2 Approximation der Randwertaufgaben durch die Finite Differenzen Methode

Für die Diskretisierung der betrachteten linearen und nichtlinearen partiellen Differentialgleichungssysteme werden Differenzenverfahren verwendet. Differenzenverfahren sind dadurch charakterisiert, dass die auftretenden Ableitungen durch Differenzenquotienten ersetzt werden. Wegen der zur Regularisierung verwendeten Bilinearform enthalten die partiellen Differentialgleichungen alle den Differentialoperator

$$L := -\Delta \cdot -\nabla(\nabla\cdot).$$

Die Approximation des Differentialoperators L erfolgt durch die Differenzenoperatoren

$$\begin{aligned}
-u_{xx} &= \tfrac{1}{h^2}\begin{bmatrix} -1 & +2 & -1 \end{bmatrix} u + \mathcal{O}(h^2) \\
-u_{yy} &= \tfrac{1}{h^2}\begin{bmatrix} -1 \\ +2 \\ -1 \end{bmatrix} u + \mathcal{O}(h^2) \\
-\Delta u &= \tfrac{1}{h^2}\begin{bmatrix} & -1 & \\ -1 & +4 & -1 \\ & -1 & \end{bmatrix} u + \mathcal{O}(h^2) \\
-u_{xy} &= \tfrac{1}{4h^2}\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix} u + \mathcal{O}(h^2)
\end{aligned}$$

zweiter Ordnung mit Schrittweite h. Insgesamt ergibt sich der folgende Differenzenoperator

$$L_h(u_{1,h}, u_{2,h}) = (-\Delta_h - \nabla_h(\nabla_h))(u_{1,h}, u_{2,h}) =$$

$$\begin{cases}
\tfrac{1}{h^2}\begin{bmatrix} 0 & -1 & 0 \\ -2 & 6 & -2 \\ 0 & -1 & 0 \end{bmatrix} u_{1,h} + \tfrac{1}{4h^2}\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix} u_{2,h} = f_{1,h} \quad \text{in} \quad \Omega_h \\
\tfrac{1}{h^2}\begin{bmatrix} 0 & -2 & 0 \\ -1 & 6 & -1 \\ 0 & -2 & 0 \end{bmatrix} u_{2,h} + \tfrac{1}{4h^2}\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix} u_{1,h} = f_{2,h} \quad \text{in} \quad \Omega_h \\
u_{1,h} = u_{2,h} = 0 \quad \text{in} \quad \Gamma_h.
\end{cases}$$

Zur Approximation der rechten Seite in (3.13) und (3.16) bzw. des nichtlinearen Funktionals $f(u)$ in (3.22)

$$\begin{aligned} f_h(u_h(x_h)) = J_D(u_h(x_h)) &= J_T(\phi(x_h, u_h(x_h))) \cdot J_\phi(u_h(x_h)) \cdot (T(x_h - u_h(x_h)) - R(x_h)) \\ &= -\nabla T(\phi(u_h)) \cdot (T(x_h - u_h(x_h)) - R(x_h)) \end{aligned} \tag{4.3}$$

werden die ersten Ableitungen von T an der Stelle $\phi(u_h(x_h)) = x_h - u_h(x_h)$ durch die symmetrischen Differenzenoperatoren

$$T_x = \frac{1}{2h} \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} T_h + \mathcal{O}(h^2)$$

bzw.

$$T_y = \frac{1}{2h} \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} T_h + \mathcal{O}(h^2)$$

zweiter Ordnung ersetzt. Für einen Gitterpunkt $(x_i, x_j) \in G_h$ gilt

$$f_h(u_h(x_i, x_j)) = \frac{1}{2}\Big(\frac{T^h_{i+1,j} - T^h_{i-1,j}}{h}, \frac{T^h_{i,j+1} - T^h_{i,j-1}}{h}\Big)^t (T^h_{i,j} - R^h_{i,j})$$

mit $T^h_{i,j} = T^h(x_i - u_{1,h}(x_i, x_j), x_j - u_{2,h}(x_i, x_j))$ und $R^h_{i,j} = R^h(x_i, x_j)$.

Mit dem diskreten nichtlinearen Funktional $f_h(u_h)$ erhält man den Differenzenoperator

$$N_h(u_{1,h}, u_{2,h}) = L_h(u_{1,h}, u_{2,h}) + f(u_{1,h}, u_{2,h}) =$$

$$\begin{cases} \frac{1}{h^2} \begin{bmatrix} 0 & -1 & 0 \\ -2 & 6 & -2 \\ 0 & -1 & 0 \end{bmatrix} u_{1,h} + \frac{1}{4h^2} \begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix} u_{2,h} - f_{1,h}(u_{1,h}, u_{2,h}) = 0 \quad \text{in} \quad \Omega_h \\ \\ \frac{1}{h^2} \begin{bmatrix} 0 & -2 & 0 \\ -1 & 6 & -1 \\ 0 & -2 & 0 \end{bmatrix} u_{2,h} + \frac{1}{4h^2} \begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix} u_{1,h} - f_{2,h}(u_{1,h}, u_{2,h}) = 0 \quad \text{in} \quad \Omega_h \\ \\ u_{1,h} = u_{2,h} = 0 \quad \text{in } \Gamma_h \end{cases}$$

für die zweidimensionale Problemstellung.

4.3 Mehrgitterverfahren zur Lösung der Randwertaufgaben

Die Approximation von Randwertproblemen durch die Methode der finiten Differenzen führt, in Abhängigkeit von der gewählten Diskretisierung, auf große schwachbesetzte Gleichungssysteme, für die im Folgenden geeignete numerische Lösungsverfahren vorgestellt werden. Zunächst werden mit den Relaxationsverfahren klassische

Iterationstechniken zur Lösung linearer Gleichungssysteme betrachtet. Diese konvergieren in den ersten Iterationsschritten sehr schnell, jedoch im Laufe der Iterationen verlieren sie deutlich an Konvergenzgeschwindigkeit, so dass sich eine schlechte asymptotische Konvergenzrate einstellt.
Trotzdem sind herkömmliche Iterationsverfahren ein wichtiger Bestandteil der im weiteren betrachteten Mehrgitterverfahren. Hierbei werden verschiedene Gitterebenen $G_l := G_{h_l}, l = 1, .., L$, mit den Schrittweiten $h_1 > h_2 > ... > h_l$ in den Lösungsprozess mit einbezogen.

4.3.1 Relaxationsverfahren

Die Konvergenz von Relaxationsverfahren ist abhängig von der Diskretisierungsschrittweite h (h-Abhängigkeit). Für sehr feine Diskretisierungsgitter, wie sie bei digitalen Bilddatensätzen mit kleiner Abtastschrittweite (also mit hohem Informationsgehalt) benötigt werden, geht die Konvergenzrate ρ der Relaxationsverfahren (typisch $\rho = 1 - \mathcal{O}(h^2)$) gegen eins. Deshalb sind die in diesem Abschnitt betrachteten Relaxationsverfahren zur numerischen Lösung der untersuchten Randwertprobleme ungeeignet.

Relaxationsverfahren werden innerhalb der Mehrgitteriteration zur Glättung der hochfrequenten Fehleranteile eingesetzt und nicht zur Lösung des Problems. Die Untersuchungen beschränken sich deshalb auf die Glättungseigenschaften und nicht auf die davon oft sehr stark abweichenden Konvergenzeigenschaften der Relaxationsverfahren. Wegen der schnellen Reduktion der hochfrequenten Fehleranteile kann die Anzahl der Relaxationsschritte auf den verschiedenen Gitterebenen innerhalb der Mehrgitteriteration sehr klein gewählt werden. Erfahrungsgemäß genügen 1-3 Iterationsschritte auf jeder Gitterebene um die hochfrequenten Fehlerkomponenten vernünftig zu glätten.

Spezielle Relaxationsverfahren

Im Folgenden werden typische Relaxationsverfahren (Jacobi Gesamtschrittverfahren (GSV), Gauß-Seidel Einzelschrittverfahren (ESV) und sukzessive Überrelaxationsverfahren (SOR(ω)) mit Relaxationsparameter ω) als Komponente des Mehrgitterverfahrens zur Lösung der linearen Randwertaufgabe (3.16) aufgeführt. Der Glättungsfaktor der verschiedenen Verfahren hängt nicht nur von dem verwendeten Relaxa-

tionsverfahren ab, sondern auch von der Nummerierung der Gitterpunkte.
In dieser Arbeit werden lexikographische und schachbrettartige Nummerierungen eingesetzt. Bei der lexikographischen Nummerierung wird die Nummerierung der Gitterpunkte übernommen, d.h. zuerst werden alle Punkte der ersten Zeile spaltenweise durchlaufen, dann die der zweiten Zeile solange, bis die letzte Zeile erreicht ist. Die schachbrettartige Nummerierung zerlegt die Gitterpunkte in schwarze und weiße Punkte. Hierbei werden zunächst alle weißen Punkte relaxiert und anschließend alle schwarzen Punkte unter Benutzung der neuen weißen Punkte.
Bei der zu lösenden Gleichung (3.17) handelt es sich um ein System von Differentialgleichungen. Bei Systemen kann man zusätzlich zwischen Verfahren unterscheiden, die zunächst alle Punkte eines Gitters vollständig relaxieren und anschließend die Gitterpunkte der übrigen Gitter, wie zum Beispiel bei dem Block-Gauß-Seidel Verfahren mit lexikographischer Nummerierung der Gitterpunkte:

- Relaxation sämtlicher Gitterpunkte des ersten Gitters mit einem Gesamtschrittverfahren und lexikographischer Nummerierung

$$v_{1,h}^{(n+1)} = \tfrac{1}{6}\left(h^2 f_{1,h} \;+\; \begin{bmatrix} & 1 & \\ 2 & & 2 \\ & 1 & \end{bmatrix} v_{1,h}^{(n)} \;+\; \tfrac{1}{4} \begin{bmatrix} -1 & & 1 \\ & & \\ 1 & & -1 \end{bmatrix} v_{2,h}^{(n)} \right)$$

- Relaxation aller Gitterpunkte des zweiten Gitters unter Verwendung der gerade neu berechneten Gitterpunkte des ersten Gitters

$$v_{2,h}^{(n+1)} = \tfrac{1}{6}\left(h^2 f_{2,h} \;+\; \begin{bmatrix} & 2 & \\ 1 & & 1 \\ & 2 & \end{bmatrix} v_{2,h}^{(n)} \;+\; \tfrac{1}{4} \begin{bmatrix} -1 & & 1 \\ & & \\ 1 & & -1 \end{bmatrix} v_{1,h}^{(n+1)} \right).$$

Hierbei kann die Art der Nummerierung der Gitterpunkte, das Relaxationsverfahren für die verschiedenen Gitter und die Verwendung der neuen oder alten Werte bei den verschiedenen Blöcken beliebig variiert werden.
In der Praxis zeigt sich doch sehr deutlich, dass die vollständige Behandlung der verschiedenen Gitter nicht nur schlechtere Konvergenzraten, sondern auch schlechtere Glättungsraten für das Mehrgitterverfahren bedeuten. Günstiger ist es, die verschiedenen Gitter gleichzeitig zu durchlaufen und die neuberechneten Gitterpunkte sofort wieder zu verwenden, zum Beispiel die Gauß-Seidel Relaxation mit lexikographischer

Nummerierung der Gitterpunkte:

$$v_{1,h}^{(n+1)} = \tfrac{1}{6}\Bigg(h^2 f_{1,h} + \begin{bmatrix} 2 & \\ & 1 \end{bmatrix} v_{1,h}^{(n+1)} + \begin{bmatrix} & 1 \\ & 2 \end{bmatrix} v_{1,h}^{(n)} + \tfrac{1}{4}\begin{bmatrix} & \\ 1 & -1 \end{bmatrix} v_{2,h}^{(n+1)} + \tfrac{1}{4}\begin{bmatrix} -1 & 1 \\ & \end{bmatrix} v_{2,h}^{(n)} \Bigg)$$

$$v_{2,h}^{(n+1)} = \tfrac{1}{6}\Bigg(h^2 f_{1,h} + \begin{bmatrix} 2 & \\ & 1 \end{bmatrix} v_{2,h}^{(n+1)} + \begin{bmatrix} & 1 \\ & 2 \end{bmatrix} v_{2,h}^{(n)} + \tfrac{1}{4}\begin{bmatrix} & \\ 1 & -1 \end{bmatrix} v_{1,h}^{(n+1)} + \tfrac{1}{4}\begin{bmatrix} -1 & 1 \\ & \end{bmatrix} v_{1,h}^{(n)} \Bigg).$$

Auch hierbei kann die Nummerierung der Gitterpunkte und die Wahl der Relaxationsverfahren beliebig variiert werden. Zur Verbesserung der Glättungsraten innerhalb der Mehrgitterverfahren hat sich eine leichte Überrelaxation der Relaxationsverfahren (SOR(ω) mit $\omega > 1$) bewährt.

4.3.2 Lineare Mehrgitterverfahren MG-CS

Untersucht man die Relaxationsverfahren mit einer Fourieranalyse, so erkennt man, dass die hochfrequenten Anteile des Fehlers unabhängig von der Gitterweite h stark reduziert werden, während niedrigfrequente Anteile kaum gedämpft werden (vgl. etwa [17]). Der Grundgedanke der Mehrgitterverfahren ist einfach und nutzt aus, dass die niedrigfrequenten Anteile des Fehlers fast ohne Informationsverlust auch auf gröberen Gitterebenen repräsentiert werden können, hochfrequente Anteile des Fehlers jedoch nicht. Das Mehrgitterprinzip beruht auf der Kombination aus Relaxationen auf feinen und groben Gittern sowie Gittertransferoperatoren auf einer Folge immer gröber werdender Gitter.

Die hierfür notwendigen Operatoren werden für zwei Räume diskreter Gitterfunktionen $(\mathcal{F}(G_h))^d$ und $(\mathcal{F}(G_H))^d$ mit $h < H$ aus Kapitel 4.1 wie folgt

Relaxationsoperator	$\mathcal{R}_h : (\mathcal{F}(G_h))^d \to (\mathcal{F}(G_h))^d$
Grobgitteroperator	$L_H : (\mathcal{F}(G_H))^d \to (\mathcal{F}(G_H))^d$
Restriktionsoperator	$I_h^H : (\mathcal{F}(G_h))^d \to (\mathcal{F}(G_H))^d$
Interpolationsoperator	$I_H^h : (\mathcal{F}(G_H))^d \to (\mathcal{F}(G_h))^d$

eingeführt.

Das Zweigitterverfahren

Eine grundlegende Rolle in der Mehrgittertechnik spielt die Tatsache, dass der Fehler $e_h^{(n)} = u_h - u_h^{(n)}$ die Defektgleichung

$$L_h e_k^{(n)} = L_h(u_h - u_h^{(n)}) = f_h - L_h u_h^{(n)} = d_h^{(n)}$$

für einen linearen Operator L_h erfüllt. Wegen

$$u_h = e_h^{(n)} + u_h^{(n)}$$

kann der Fehler $e_h^{(n)}$ also als Korrektur für die aktuelle Näherung $u_h^{(n)}$ verwendet werden. Ein Iterationsschritt eines Zweigitterverfahrens lässt sich mit gegebener aktueller Näherungslösung $u_h^{(n)}$ folgendermaßen formulieren:

(ZGV 1) Vorglättung durch ν_1 Iterationen eines Relaxationsverfahrens

$$\overline{u}_h^{(n)} = \mathcal{R}_h^{\nu_1}(u_h^{(n)}, L_h, f_h)$$

(ZGV 2) Berechnung des Defekts

$$d_h = f_h - L_h(\overline{u}_h^{(n)})$$

(ZGV 3) Restriktion des Defekts auf das gröbere Gitter

$$d_H = I_h^H(d_h)$$

(ZGV 4) Berechnung der exakten Lösung v_H der Grobgitterkorrekturgleichung

$$L_H v_H = d_H$$

(ZGV 5) Interpolation der Grobgitterkorrektur auf das feine Gitter

$$v_h = I_H^h(v_H)$$

(ZGV 6) Berechnung der korrigierten Näherung

$$\tilde{u}_h^{(n)} = \overline{u}_h^{(n)} + v_h$$

(ZGV 7) Nachglättung durch ν_2 Iterationen eines Relaxationsverfahrens

$$u_h^{(n+1)} = \mathcal{R}_h^{\nu_2}(\tilde{u}_h^{(n)}, L_h, f_h).$$

Das Zweigitterverfahren zur Lösung linearer Randwertaufgaben benutzt das grobe Gitter G_H ausschließlich zur Korrektur von Feingitterlösungen, deshalb wird es auch Correction Scheme (CS) genannt.

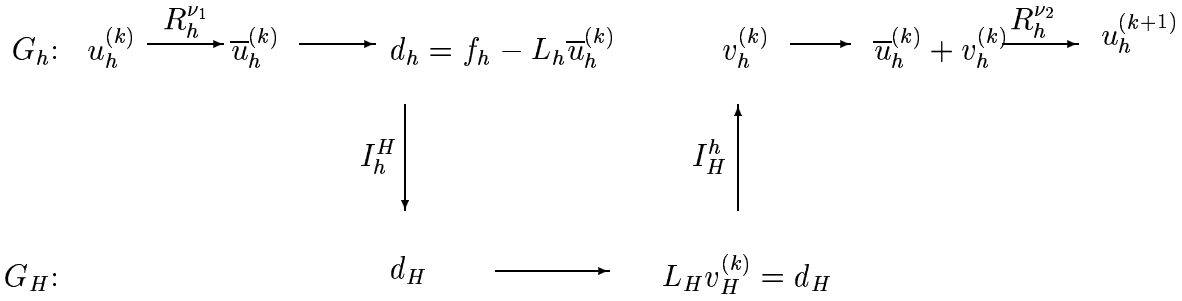


Abbildung 4.4: Illustration des Zweigitter Correction Scheme.

Das Mehrgitterverfahren

Ersetzt man im Zweigitterverfahren den Teilschritt (ZGV 4) rekursiv durch γ Iterationen eines Zweigitterverfahrens mit neuer gröberer Schrittweite h_{2H} bis zur gröbsten Schrittweite, so erhält man einen Mehrgitteralgorithmus. Abhängig von der Größe des gröbsten Gitters wird hier die Grobgittergleichung direkt oder iterativ gelöst. Für $\gamma = 1$ erhält man den sogenannten V-Zyklus, für $\gamma = 2$ den in vielen Fällen robusteren W-Zyklus (vgl. Abb.4.5).

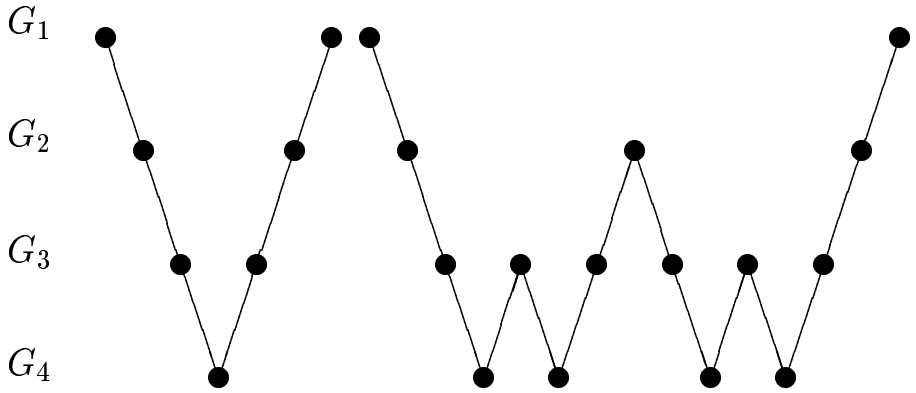


Abbildung 4.5: V- und W-Zyklus bei $L = 4$ Gitterebenen.

4.3.3 Gittertransferoperatoren

Ein Kernstück der Mehrgitteriteration ist die Transformation von Informationen zwischen den groben und feinen Gitterebenen. Hierfür wird die Standardvergröberung der Diskretisierungsgitter

$$2 \cdot h_l = h_{l-1}$$

für eine Folge von Schrittweiten

$$h_1 > \cdots > h_l \cdots \quad \text{mit} \quad \lim_{l \to 0} h_l = 0$$

verwendet. Bei der cell centered Diskretisierung entspricht das einer Zerlegung des Gebiets Ω in $2^l \times 2^l$ achsenparallele Teilgebiete. Das zweidimensionale Diskretisierungsgitter mit der Schrittweite $h_l = 1/2^l$

$$G_{h_l} = \left\{ x \in I\!R^2 : x = ((i - 1/2) \cdot h_l, (j - 1/2) \cdot h_l),\ i, j = 1, 2, 3, .., 2^l \right\} \tag{4.4}$$

fasst die Mittelpunkte der Gitterzellen zusammen (vgl. Kapitel 4.1). Hierdurch liegen die Grobgitterpunkte bei der cell centered Diskretisierung (vgl. Abbildung 4.6) nicht auf der nächst feineren Gitterebene.
Im Folgenden werden Operatoren für den Transfer zwischen zwei verschiedenen Gitterebenen vorgestellt. Diese sind so skaliert, dass eine konstante Gitterfunktion auf die gleiche konstante Gitterfunktion auf einer feineren bzw. gröberen Gitterebene abgebildet wird.

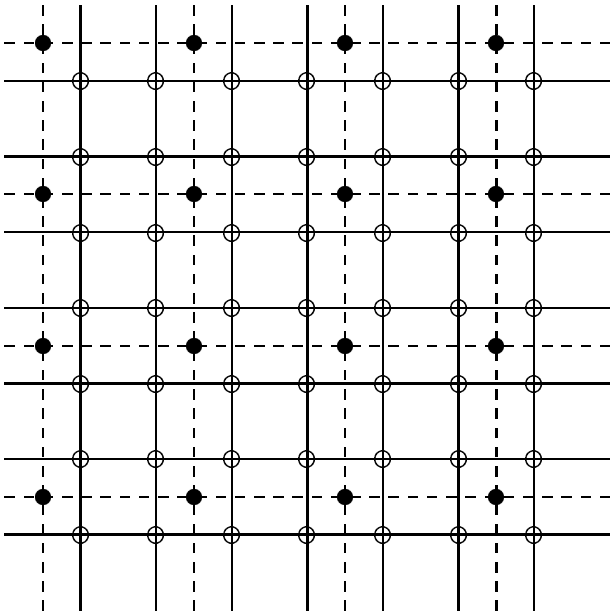


Abbildung 4.6: Lage des durch die Standardvergröberung erzeugten groben Gitters bei der Cell Center Diskretisierung relativ zum feinen Gitter.

Cell centered Interpolation

Stückweise konstante Interpolation

Die stückweise konstante Interpolation wird auch als nearest neighbour Interpolation bezeichnet, weil sie jedem feinen Gitterpunkt den Wert des am nächsten liegenden groben Gitterpunkt zuordnet.
Gilt $h = K \cdot H$ mit $1/h$ der Anzahl der feinen und $1/H$ der Anzahl der groben Gitterpunkte, so ergeben sich für die eindimensionale konstante Interpolation die folgenden Werte für die feinen Gitterpunkte:

$$\begin{aligned} (I_H^h u_H)_{K\cdot i-k} &= (u_H)_i \quad \text{für} \quad 1 \le k \le \lceil \frac{K}{2} \rceil \\ (I_H^h u_H)_{K\cdot i+k} &= (u_H)_i \quad \text{für} \quad 0 \le k \le \lfloor \frac{K}{2} \rfloor. \end{aligned}$$

Für die Standardverfeinerung mit $2h = H$ vgl. Abb. 4.7 gilt:

$$\underbrace{(I_H^h u_H)_{2\cdot i-1}}_{=(u_h)_{2\cdot i-1}} = \underbrace{(I_H^h u_H)_{2\cdot i}}_{=(u_h)_{2\cdot i}} = (u_H)_i.$$

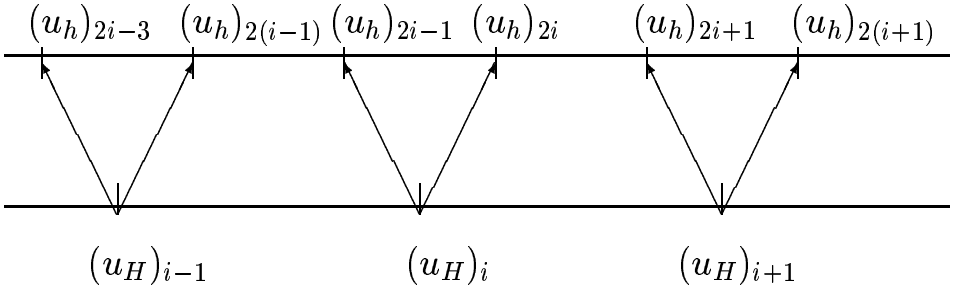


Abbildung 4.7: Illustration der eindimensionalen konstanten Interpolation für $h = 2H$.

Lineare Interpolation

Bei der linearen Interpolation werden die direkt benachbarten Grobgitterpunkte eines Feingitterpunktes anteilmäßig zusammengefasst. Gilt wie oben $h = K \cdot H$, so ergeben sich für die eindimensionale lineare Interpolation die folgenden Werte für die feinen Gitterpunkte:

$$(I_H^h u_H)_{K\cdot i+j} = \frac{2(K-j)-1}{2K}(u_H)_i + \frac{2j+1}{2K}(u_H)_{i+1}, \quad \text{für} \quad 0 \le j \le L-1$$

$$(I_H^h u_H)_{K\cdot i-j} = \frac{2(K-j)-1}{2K}(u_H)_{i-1} + \frac{2j+1}{2K}(u_H)_i, \quad \text{für} \quad 1 \le j \le L.$$

Für die Standardverfeinerung ergeben sich die folgenden Werte (vgl. Abb. 4.8):

$$(I_H^h u_H)_{2\cdot i-1} = \frac{3}{4}(u_H)_i + \frac{1}{4}(u_H)_{i-1}$$

$$(I_H^h u_H)_{2\cdot i} = \frac{3}{4}(u_H)_i + \frac{1}{4}(u_H)_{i+1}.$$

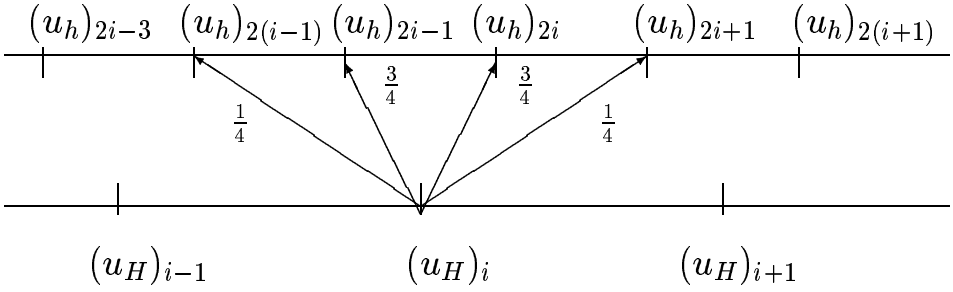


Abbildung 4.8: Lineare Interpolation für $K = 2$.

Bilineare Interpolation

Die im obigen Text beschriebenen Interpolationen lassen sich direkt auf zwei- oder dreidimensionale Probleme übertragen. In der Praxis wird häufig die bilineare Interpolation verwendet. Bei der Standardvergröberung der Gitterpunkte berechnen sich die Werte des feinen Gitters für die bilineare Interpolation (gemäß Abbildung 4.9) wie folgt:

$$\begin{aligned} u_h^a &= \frac{1}{16}(9u_H^A + 3u_H^B + 3u_H^C + u_H^D) \\ u_h^b &= \frac{1}{16}(9u_H^B + 3u_H^A + 3u_H^D + u_H^C) \\ u_h^c &= \frac{1}{16}(9u_H^C + 3u_H^A + 3u_H^D + u_H^B) \\ u_h^d &= \frac{1}{16}(9u_H^D + 3u_H^B + 3u_H^C + u_H^A). \end{aligned}$$

Cell centered Restriktion

Nearest neighbour Restriktion

Bei der nearest neighbour Restriktion der feinen Gitterpunkte auf gröbere Gitterebenen wird den groben Gitterpunkten der Wert des am nächsten liegenden feinen Gitterpunktes zugeordnet. Diese Art der Zuordnung ist nicht eindeutig, da mehrere feine Gitterpunkte den gleichen Abstand zu einem Grobgitterpunkt haben (vgl. Abbildung 4.6), so dass für die praktische Umsetzung ein weiteres Kriterium eingeführt werden muss. In dieser Arbeit wurde der am nächsten zum Ursprung liegende feine Gitterpunkt gewählt. Ein Beispiel hierfür ist die in der Abbildung 4.10 illustrierte eindimensionale nearest neighbour Restriktion.

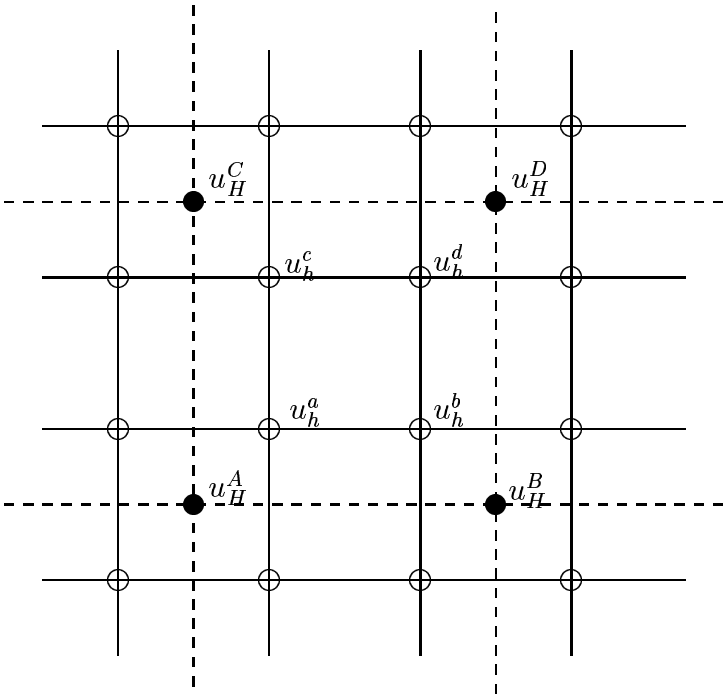


Abbildung 4.9: Bezeichnung der Gitterpunkte bei der bilinearen Interpolation.

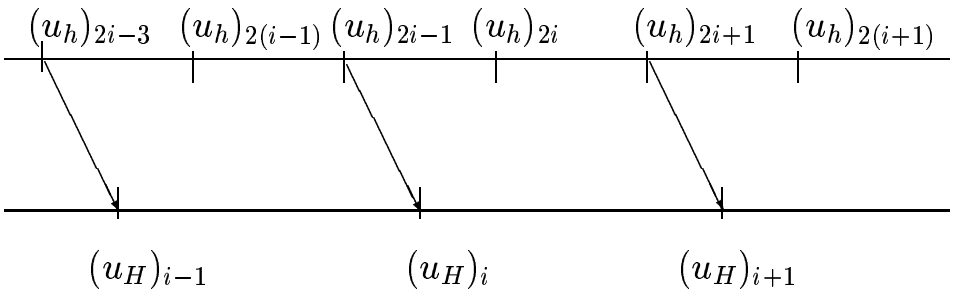


Abbildung 4.10: Illustration der eindimensionalen nearest neighbour Restriktion.

Gewichtete Restriktion

Bei der nearest neighbour Restriktion verliert man die Information vieler feiner Gitterpunkte. Bei der gewichteten Restriktion werden alle Werte der Nachbarpunkte auf dem feinen Gitter zur Berechnung des Wertes im Grobgitterpunkt herangezogen.

Numerische Ergebnisse

In Tabelle 4.1 sind Näherungen für den Spektralradius für das MG-CS mit verschiedenen Mehrgitterkomponenten aufgeführt. Zum Transfer der Daten zwischen den Gitterebenen wird die bilineare Interpolation bzw. die gewichtete Restriktion verwendet.

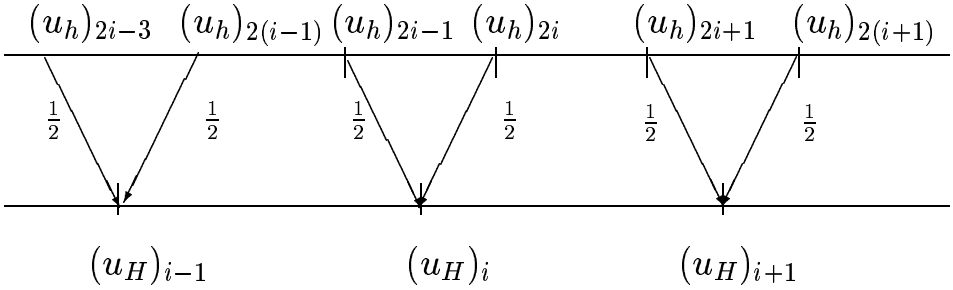


Abbildung 4.11: Illustration der eindimensionalen gewichteten Restriktion.

Zur Berechnung der Lösung wird ein V-Zyklus mit maximaler Anzahl von Grobgitter-

Dimension	2D		3D	
pre/post	$(0,2)$	$(2,2)$	$(0,2)$	$(2,2)$
$GS_{lex}(1.00)$	0.26	0.14	0.35	0.16
$GS_{lex}(1.10)$	0.20	0.10	0.27	0.15
$GS_{lex}(1.20)$	0.15	0.09	0.21	0.12
$GS_{lex}(1.30)$	0.28	0.12	0.36	0.12
$GS_{rb}(1.00)$	0.27	0.11	0.34	0.15
$GS_{rb}(1.10)$	0.23	0.09	0.28	0.13
$GS_{rb}(1.20)$	0.19	0.07	0.23	0.09
$GS_{rb}(1.30)$	0.16	0.04	0.17	0.05

Tabelle 4.1: Konvergenzradien der Mehrgitterverfahren für einen V-Zyklus.

ebenen verwendet. Als besonders günstige Glättungsverfahren haben sich die Gauß-Seidel Einzelschrittverfahren mit lexikographischer (GS_{lex}) oder schachbrettartiger (GS_{sb}) Nummerierung der Gitterpunkte und leichter Überrelaxation erwiesen. Die optimalen Werte für den Relaxationsparameter ω liegen bei der lexikographischen Nummerierung bei etwa $\omega \approx 1.2$ und bei der schachbrettartigen Nummerierung bei etwa $\omega \approx 1.3$. Die Berechnung der Ergebnisse erfolgte anhand der rechten Seite

$$\begin{aligned} f_1(x,y) &= \pi^2 \cdot \mu \cdot (2 \cdot \sin \pi x \sin \pi y - (\cos \pi x \cos \pi y - \sin \pi x \sin \pi y)) \\ f_2(x,y) &= \pi^2 \cdot \mu \cdot (2 \cdot \sin \pi x \sin \pi y - (\cos \pi x \cos \pi y - \sin \pi x \sin \pi y)) \end{aligned}$$

mit der ihr zugehörigen exakten Lösung

$$\begin{aligned} u_1(x,y) &= \sin \pi x \sin \pi y \\ u_2(x,y) &= \sin \pi x \sin \pi y, \end{aligned}$$

für die zweidimensionale Problemstellung und den entsprechenden Funktionen in drei Dimensionen für die dreidimensionale Problemstellung.

4.3.4 Mehrgitterverfahren zur Lösung nichtlinearer partieller Differentialgleichungen

Zur numerischen Lösung nichtlinearer partieller Differentialgleichungen, wie etwa Gleichung (3.22) mit nichtlinearem Operator $N = L \cdot -f(\cdot)$ und dessen Diskretisierung N_h, sind zwei grundsätzlich verschiedene Vorgehensweisen bekannt.
Eine Möglichkeit ist das Mehrgitter-Newton Verfahren, bei dem ein lineares Mehrgitterverfahren mit einem Newton Verfahren kombiniert wird. Die zweite Möglichkeit ist, die Mehrgitteridee durch ein Full Approximation Scheme (FAS) direkt auf das nichtlineare Problem anzuwenden.

Das Mehrgitter-Newton Verfahren

Die globale Linearisierung (etwa durch das Newton Verfahren) zur Lösung der diskretisierten nichtlinearen Randwertaufgabe

$$\begin{array}{rcl} N_h u_h & = & g_h \quad \text{auf} \quad \Omega_h \\ u_h & = & 0 \quad \text{auf} \quad \Gamma_h \end{array} \tag{4.5}$$

führt auf die folgende Iterationsvorschrift

$$u_h^{(n+1)} = u_h^{(n)} - (J_h^{(n)})^{-1} \underbrace{(N_h u_h^m - g_h)}_{=-d_h^n} \quad \text{für} \quad n = 0, 1, ...$$

mit geeignetem Startwert $u_h^{(0)}$, dem Defekt $d_h^{(n)}$ und der Jacobi-Matrix

$$J_h^{(n)} = N_h'[u_h^{(n)}]$$

der n-ten Iteration. In jedem Iterationsschritt wird hierbei das lineare Problem

$$J_h^{(n)} v_h^{(n)} = d_h^{(n)}$$

mit der Defektkorrektur $v_h^{(n)} = u_h^{(n+1)} - u_h^{(n)}$ gelöst.
Beim Mehrgitter-Newton Verfahren wird durch Lösen des linearisierten Problems durch ein Correction Scheme aus Kapitel 4.3.2 die Richtung der nächsten Verbesserung bestimmt. Anschließend wird durch Lösen eines eindimensionalen Unterproblems die Länge des Schrittes ermittelt. Hierfür wird ein Dämpfungsparameter λ eingeführt, durch den das Verfahren stabilisiert wird. Für eine aktuelle Näherungslösung $u^{(n)}$ ergibt sich der folgende Algorithmus (vgl. [6]):

(MG-NV 1) (Bestimmung der Abstiegsrichtung)

$d_h^{(n)} = g_h - N_h(u_h^{(n)})$

löse $J_h(u_h^{(n)})v_h^{(n)} = d_h^{(n)}$ mit Startwert $v_h^{(n)} = 0$ mit dem Correction Scheme.

(MG-NV 2) (1D-Unterproblem, lineare Suche)

Wähle $\lambda \leq 1$ maximal, so dass

$$||N_h(u_h^{(n)} + \lambda v_h^{(n)}) - g_h^{(n)}|| \leq (1 - \frac{\lambda}{2})||N(u_h^{(n)}) - g_h^{(n)}||.$$

(MG-NV 3) Korrektur

setze $u_h^{(n+1)} = u_h^{(n)} + \lambda v_h^{(n)}$

Das Full Approximation Scheme (FAS)

Neben einer globalen Linearisierung der nichtlinearen partiellen Differentialgleichung gibt es die Möglichkeit einer lokalen Linearisierung, die auf nichtlineare Iterationsverfahren zur Lösung der Gleichung führt. Durch die Approximationen zweiter Ordnung mit der Finiten Differenzen Methode für den Operator L erhält man zum Beispiel das nichtlineare Gesamtschrittverfahren:

$$\frac{6}{h^2}u_{1,h}^{(n+1)} + f_{1,h}(u^{(n+1)}) = g_{1,h} + \frac{1}{h^2}\begin{bmatrix} & 1 & \\ 2 & & 2 \\ & 1 & \end{bmatrix} u_{1,h}^{(n)} + \frac{1}{4h^2}\begin{bmatrix} -1 & & 1 \\ & & \\ 1 & & -1 \end{bmatrix} u_{2,h}^{(n)}$$

$$\frac{6}{h^2}u_{2,h}^{(n+1)} + f_{2,h}(u^{(n+1)}) = g_{2,h} + \frac{1}{h^2}\begin{bmatrix} & 2 & \\ 1 & & 1 \\ & 2 & \end{bmatrix} u_{2,h}^{(n)} + \frac{1}{4h^2}\begin{bmatrix} -1 & & 1 \\ & & \\ 1 & & -1 \end{bmatrix} u_{1,h}^{(n)}$$

zur Lösung des Randwertproblems (4.5).

Wie bei der Lösung linearer Probleme, erkennt man hierbei, dass durch das nichtlineare Relaxationsverfahren nur die hochfrequenten Fehleranteile vernünftig gedämpft werden. Durch das nichtlineare FAS Mehrgitterverfahren (vgl. [7], [30]) wird die Lösung der Gleichung auf mehrere Gitterebenen übertragen. Die groben Gitterebenen dienen hierbei nicht nur zur Korrektur der auf den feinen Gittern

berechneten Näherungslösungen, sondern auch zur Approximation der Lösung des Problems.

Hierzu benötigt man, neben den für das Correction Scheme definierten Operatoren, einen weiteren

$$\text{Restriktionsoperator } \hat{I}_h^H : (\mathcal{F}(G_h))^d \to (\mathcal{F}(G_H))^d$$

zur Überführung der aktuellen Näherungslösung $\overline{u}_h^{(n)}$ auf die nächst gröbere Gitterebene. Zur Lösung der Defektgleichung

$$N_H(\hat{I}_h^H(u_h^{(n)}) + v_H^{(n)}) = d_H + N_H(u_H^{(n)}) \tag{4.6}$$

für das FAS Mehrgitterverfahren auf den gröberen Gittern, wird das lineare Relaxationsverfahren $\mathcal{R}_h$ durch den

$$\text{nichtlinearen Relaxationsoperator } \hat{\mathcal{R}}_h : (\mathcal{F}(G_h))^d \to (\mathcal{F}(G_h))^d$$

ersetzt.

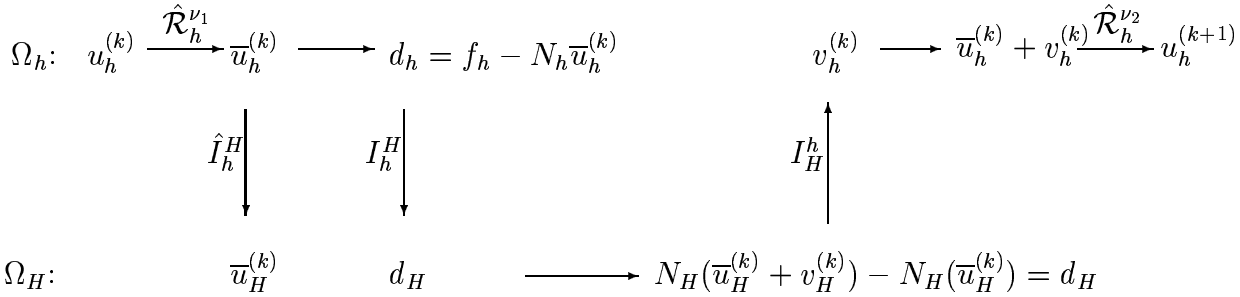


Abbildung 4.12: Illustration des Zweigitter Full Approximation Scheme.

Zur Glättung der hochfrequenten Fehleranteile auf den verschiedenen Gitterebenen hat sich hierbei die Jacobi-Picard bzw. Gauß-Seidel-Picard Iteration als wirkungsvolle Alternative zum Jacobi Gesamtschritt- bzw. Gauß-Seidel Einzelschrittverfahren bewährt (vgl. z.B. [36]). Hierbei wird der nichtlineare Anteil des Operators N nicht während der Iteration ausgewertet, sondern einmalig

$$\hat{g}_h = g_h - f_h(u_h^{(n)})$$

vor der $(n+1)$-ten Iteration berechnet. Man erhält hierdurch etwa die Jacobi-Picard Relaxation

$$\frac{6}{h^2}u_{1,h}^{(n+1)} = \hat{g}_{1,h} + \frac{1}{h^2}\begin{bmatrix} & 1 & \\ 2 & & 2 \\ & 1 & \end{bmatrix} u_{1,h}^{(n)} + \frac{1}{4h^2}\begin{bmatrix} -1 & & 1 \\ & & \\ 1 & & -1 \end{bmatrix} u_{2,h}^{(n)}$$

$$\frac{6}{h^2}u_{2,h}^{(n+1)} = \hat{g}_{2,h} + \frac{1}{h^2}\begin{bmatrix} & 2 & \\ 1 & & 1 \\ & 2 & \end{bmatrix} u_{2,h}^{(n)} + \frac{1}{4h^2}\begin{bmatrix} -1 & & 1 \\ & & \\ 1 & & -1 \end{bmatrix} u_{1,h}^{(n)}$$

für das nichtlineare Randwertproblem (4.5). Das nichtlineare FAS Zweigitterverfahren kann durch das Schema in Abbildung 4.12 illustriert werden. Die Erweiterung auf ein Mehrgitterverfahren erfolgt analog zum Correction Scheme rekursiv.

Kapitel 5

Minimierung des Defektfunktionals mit Hilfe von Mehrgitterverfahren

In diesem Abschnitt werden die in Kapitel 3 beschriebenen Minimierungs- und Regularisierungsverfahren für das Defektfunktional

$$D(u) = \langle h(u), h(u) \rangle \quad \text{mit} \quad h(u) = T(x - u(x)) - R(x)$$

mit den in Kapitel 4 beschriebenen Mehrgitterverfahren zur Berechnung der Abstiegsrichtungen kombiniert. Obwohl durch die Einführung eines Regularisierungsterms das Problem "gutartiger" wird, müssen bei der Wahl des Minimierungsverfahrens mehrere Aspekte berücksichtigt werden. Die iterativ berechneten Näherungslösungen sollen einerseits gegen eine geeignete Lösung des in Kapitel 2 beschriebenen inversen Problems konvergieren. Andererseits spielen praktische Belange eine erhebliche Rolle. Die Anzahl der Daten ist in praktischen Anwendungen immens. Hierdurch wird der Einsatz schneller Algorithmen erforderlich.
Im Folgenden werden zwei verschiedene Zugänge zur Lösung des Problems unterschieden. Einerseits wird ein linearisierter Ansatz vorgestellt, bei dem die in Kapitel 3.2.2 vorgestellte Landweber-Iteration durch ein Mehrgitterverfahren zur Berechnung der Abstiegsrichtung ergänzt wird. Untersuchungen zu diesem Verfahren wurden bereits in früheren Arbeiten vorgestellt. In [21] wurde für die auftretenden linearen partiellen Differentialgleichungen geeignete Lösungsverfahren diskutiert. In [22] wurden die Strategien zur Wahl geeigneter Regularisierungsparameter vorgestellt. Die Erweiterung des Verfahrens auf mehrere Auflösungsstufen (vgl. Kapitel 5.5) wurde in [23] beschrieben.

Andererseits werden nichtlineare Verfahren hergeleitet, die eine geeignete Minimallösung des Defektfunktionals durch die Minimierung des in Kapitel 3.3 untersuchten Tikhonov-Funktionals bestimmen.
Bei allen vorgestellten Verfahren ist, neben der effektiven numerischen Lösung der resultierenden partiellen Differentialgleichungen durch ein Mehrgitterverfahren, die geeignete Wahl des Regularisierungsparameters in jedem Iterationsschritt notwendig.

5.1 Linearisierte Abstiegsverfahren

Bei der in Kapitel 3.2.2 vorgestellten regularisierten Landweber-Iteration zur Minimierung des Defektfunktionals stellte sich die Abstiegsrichtung als Lösung eines partiellen Differentialgleichungssystems heraus. Die Berechnung der Abstiegsrichtung in jedem Iterationsschritt erfolgt, wie in Kapitel 4.3 beschrieben, iterativ durch ein Mehrgitterverfahren. Die Mehrgitteriteration wird in diesem Zusammenhang als innere Iteration bezeichnet, im Gegensatz zur äußeren Landweber-Iteration.

5.1.1 Wahl des Regularisierungsparameters

Wahl des Regularisierungsparameters als Lösung eines eindimensionalen Minimierungsproblems

Für Lösungen v_α des Minimierungsproblems (3.15) gilt $v_\alpha = v_1/\alpha$, d.h. eine Skalierung der Abstiegsrichtung erfolgt direkt durch den Regularisierungsparameter α. Deshalb kann hier der Regularisierungsparameter als Lösung des eindimensionalen Minimierungsproblems

$$\text{finde} \quad \alpha, \qquad \text{so dass} \quad \alpha = \arg \min_{\omega \in [0,\delta]} D(u^{(k)} + \omega d), \tag{5.1}$$

mit $d = v_1/||v_1||_\infty = v_\alpha/||v_\alpha||_\infty$ berechnet werden. Als nächste Näherungslösung $u^{(k+1)}$ berechnet man

$$u^{(k+1)} = u^{(k)} + \alpha d.$$

Mit dem sogenannten trust region Parameter δ in (5.1) bestimmt man eine Umgebung um die aktuelle Näherung $u^{(k)}$, in der das Defektfunktional minimiert werden soll.
Strategien zur Wahl des trust region Parameters werden etwa in [12] beschrieben. In

Standard-Verfahren wird der trust region Parameter so gewählt, dass für alle ω mit $0 < \omega \leq \delta$

- der Wert des Defektfunktionals in jedem Iterationsschritt reduziert wird

$$D(u^{(k)} + \omega d) < D(u^{(k)})$$

- und

$$D(u^{(k)} + \omega d) \approx D(u^{(k)}) + \omega \langle J_D(u^{(k)}), d \rangle$$

gilt. Als guter Kompromiss für die praktische Anwendung des Verfahrens bei der Anpassung digitaler Bilddaten, hat sich die konstante Wahl des trust region Parameter von $\delta \approx 2$ herausgestellt. Das bedeutet, dass der Template Bilddatensatz $T(x)$ in einem Iterationsschritt nicht mehr als zwei Bildelemente verschoben werden kann.

5.1.2 Algorithmus

Die iterative Anwendung der oben beschriebenen Schritte führt auf ein Abstiegsverfahren. Als Startnäherung wird $u^{(0)} = 0$ gewählt, das bedeutet, dass die Iterationen von den Ausgangsbilddaten $T(x)$ und $R(x)$ starten. Die Iterationen erzeugen eine Folge von Verschiebungen $\left\{u^{(k)}(x)\right\}_{k \in I\!N}$, die sukzessive den Wert des Defektfunktionals $D(u)$ reduzieren. Die Iteration wird beendet, falls $\nabla D(u_{k+1}) \approx 0$ oder eine festgelegte Anzahl von Iterationsschritten durchgeführt wurden. Hiermit erhält man den folgenden Algorithmus:

Algorithmus zur Minimierung des Defektfunktionals $D(u)$ durch linearisierte und regularisierte Abstiegsverfahren:

$k = 0$; $u_0 = 0$; $\delta_0 = 2 \cdot h$;

repeat

berechne v durch eine innere Iteration (MG-CS)

$d := v / ||v||_{\mathcal{X}}$;

bestimme α_k durch ein Kriterium aus Abschnitt 5.1.1

$u_{k+1} = u^{(k)} + \alpha_k \cdot d;\ k = k + 1;$

modifiziere (wenn nicht konstant) trust region δ_{k+1}

until $((\nabla D(u_{k+1}) \approx 0)\ ||\ (k > k_{max}))$

5.2 Das L-Kurven Verfahren

In Kapitel 3.3 wurde mit der nichtlinearen partiellen Differentialgleichung (3.22) eine notwendige Bedingung für eine Minimallösung des Tikhonov-Funktionals

$$T_\alpha(u) = D(u) + \alpha a(u,u)$$

angegeben. Mit Satz 2.3.1 sollte sich theoretisch der Wert des Defektfunktionals allein dadurch reduzieren lassen, dass man für immer kleiner werdende Regularisierungsparameter α eine Minimallösung des Tikhonov-Funktionals berechnet. Unglücklicherweise wird die numerische Lösung von Gleichung (3.22) mit einem MG-FAS für alle α mit $\alpha < \overline{\alpha}$ (für unbekanntes $\overline{\alpha}$) instabil. In [14] werden Strategien für eine geeignete Wahl des Regularisierungsparameters angegeben, welche auf dem Diskrepanzprinzip oder auf heuristischen Verfahren beruhen. Bei vielen nichtlinearen Problemen bleibt nur "trial and error", um einen geeigneten Regularisierungsparameter zu finden.

In diesem Abschnitt wird das L-Kurven Kriterium zur Bestimmung geeigneter Regularisierungsparameter vorgestellt. Es wurde Anfang der neunziger Jahre in den Arbeiten von HANSEN [18] und [19] eingeführt, und man erhält hiermit in vielen praktischen Problemfällen akzeptable Regularisierungsparameter.

Das L-Kurven Kriterium basiert auf der graphischen Darstellung einer Norm der regularisierten Lösung

$$x(\alpha) = ||u_\alpha||_X^2 = a(u_\alpha, u_\alpha)$$

und der korrespondierenden Defektnorm

$$y(\alpha) = ||T(x - u_\alpha) - R(x)||_2^2 = D(u_\alpha)$$

mit einer Näherung u_α für eine Minimallösung des Tikhonov-Funktionals $T_\alpha(u)$. Der Name "L-Kurve" entstand wegen der in vielen Fällen charakteristischen und wie der Buchstabe "L" verlaufenden Kurve

$$L = \{x(\alpha), y(\alpha)\}_{\alpha \in (0,\infty)}. \tag{5.2}$$

Die Idee bei der Bestimmung des Regularisierungsparameters α durch das L-Kurven Kriterium ist, die Normen des parameterabhängigen Glättungsterms $x(\alpha)$ und der ebenso parameterabhängigen Defektnorm $y(\alpha)$ in einer ausgewogenen Art klein zu

halten. Die L-Kurve (5.2) besitzt in vielen Anwendungen eine deutlich ausgeprägte "L-förmige" Ecke, in der für die Lösung u_α die Dominanz des Glättungsterms übergeht in Lösungen, bei denen die Defektnorm dominiert. Die Ecke der L-Kurve, also der Teil der Kurve mit maximaler Krümmung, deutet auf einen geeigneten Regularisierungsparameter hin.
In den Arbeiten von Hansen wird vorgeschlagen, den Regularisierungsparameter α so zu wählen, dass der korrespondierende Punkt auf der L-Kurve die stärkste Krümmung aufweist. Diese Wahl wird als L-Kurven Kriterium zur Bestimmung des Regularisierungsparameters bezeichnet.
Auch für die in dieser Arbeit betrachtete Problemstellung ist das L-Kurven Kriterium sinnvoll, wenn die zur Berechnung der Kurve zugrundeliegenden Punkte für einen sinnvollen Größenbereich von α berechnet werden.

5.3 Minimierung des Zielfunktionals durch ein modifiziertes MG-FAS

Im vorherigen Kapitel wurde die Diskrepanz bei der Wahl des Regularisierungsparameters α beschrieben und das L-Kurven Verfahren als Kompromisslösung für eine geeignete Wahl von α eingeführt. In diesem Abschnitt wird ein Verfahren vorgestellt, welches die Aufgabe einer geeigneten Parameterwahl während der Berechnung einer Lösung bewältigt. Hierbei wird nicht eine partielle Differentialgleichung mit festem α durch ein MG-FAS gelöst, wodurch die Näherung einer Minimallösung des entsprechenden Tikhonov-Funktionals berechnet wird, sondern es wird das MG-FAS zur Lösung von Gleichung (3.22) so modifiziert, dass innerhalb des Mehrgitterprozesses das Funktional $D(u)$ geeignet minimiert wird.
Die wichtigsten Komponenten des MG-FAS sind das Glättungsverfahren für die hochfrequenten Fehleranteile und die Grobgitterkorrektur. Die Berechnung einer geeigneten Näherungslösung $\overline{u}_h^{(k)}$ auf dem feinen Gitter erfolgt wie beim Standard MG-FAS durch ein nichtlineares Relaxationsverfahren. Das Relaxationsverfahren wird hierbei mit beliebigem α und mit einem geeigneten Startvektor $u_h^{(k)}$ durchgeführt.
Bei ungünstiger Wahl des Regularisierungsparameters $\alpha \leq \overline{\alpha}$ können die hochfrequenten Fehlerterme nicht mehr vernünftig geglättet werden. Das MG-FAS wird instabil. Um dies zu korrigieren, wird die mit den Relaxationsverfahren ermittelte Näherungslösung wie folgt modifiziert

$$\overline{\overline{u}}_h^{(k)} = u_h^{(k)} + \omega \cdot (\overline{u}_h^{(k)} - u_h^{(k)}).$$

Der Parameter ω ist hierbei die Lösung des eindimensionalen Minimierungsproblems

$$\text{finde } \omega \in I\!R, \quad \text{so dass} \quad \omega = arg \min_{\overline{\omega} \in I\!R^{>0}} D(u_h^{(k)} + \omega \cdot (\overline{u}_h^{(k)} - u_h^{(k)})). \tag{5.3}$$

Der Defekt der Gleichung ergibt sich durch

$$d_h = g_h - N_h(\overline{\overline{u}}_h^{(k)}) = -N_h(\overline{\overline{u}}_h^{(k)})$$

mit dem diskreten nichtlinearen Operator N_h und der rechten Seite $g_h = 0$ der diskreten nichtlinearen Randwertaufgabe (4.5).
Bei der Berechnung der Grobgitterkorrektur wird, genau wie beim Standard Full Approximation Scheme, zunächst der Defekt d_h und die aktuelle Näherungslösung

$\overline{\overline{u}}_h^{(k+1)}$ auf ein grobes Gitter restringiert. Anschließend wird die Lösung des Grobgitterproblems auf das feinere Gitter interpoliert und letztendlich zu der aktuellen Approximation auf dem feinen Gitter addiert.
Wegen der Abhängigkeit der Lösung v_H des Grobgitterproblems

$$N_H(u_H + v_H) = d_H + N_H(u_H) \tag{5.4}$$

vom Regularisierungsparameter α, wird die Berechnung von v_H wie folgt modifiziert. Zunächst wird die Lösung v_H mit einem nichtlinearen Relaxationsverfahren

$$\overline{u}_H = u_H + v_H = \hat{\mathcal{R}}_H(u_H, v_H, d_H) \tag{5.5}$$

berechnet. Hierbei ist

$$u_H \;=\; (u_1^{(1,1)}, \cdots, u_1^{(n-1,n-1)}, u_2^{(1,1)}, \cdots, u_2^{(n-1,n-1)}) \in \mathcal{F}(G_H) \times \mathcal{F}(G_H)$$

die Näherungslösung auf dem groben Gitter und $d_H \in \mathcal{F}(G_H) \times \mathcal{F}(G_H)$ der Defekt auf dem groben Gitter.

Im nächsten Schritt wird die Grobgitterkorrektur durch einen Parameter ω wie folgt skaliert

$$\overline{v}_H = \omega \cdot v_H.$$

Der Parameter ω ist Lösung des eindimensionalen Unterproblems

$$\text{finde } \omega \in I\!R, \quad \text{so dass} \quad \omega = arg \min_{\overline{\omega} \in I\!R^{>0}} D(u_H + \overline{\omega} \cdot v_H). \tag{5.6}$$

Die so skalierte Grobgitterkorrektur wird nach der Interpolation auf das feine Gitter zu der dort existenten Approximation addiert

$$u_h^{k+1} = \overline{\overline{u}}_h^{(k)} + I_H^h(\overline{v}_H).$$

Bei der Berechnung des Parameters ω durch die Minimierung eines eindimensionalen Unterproblems in (5.3) bzw. (5.6) wird der Glättungsterm des Tikhonov-Funktionals nicht mehr berücksichtigt. Diese Vorgehensweise reduziert den Einfluss des Regularisierungsparameters auf die Näherungslösung. Die Iteration kann hierdurch durchgeführt werden, ohne den Regularisierungsparameter vorweg durch ein Kriterium (wie z.B. das L-Kurven Kriterium) zu bestimmen. Außerdem wird hiermit der Einfluss fehlerhafter Bilddaten unterdrückt. Zusammengefasst ergeben sich

die folgenden Schritte für das modifizierte Full Approximation Scheme auf zwei Gitterebenen:

(1) Vorglättung durch ν_1 Iterationen eines nichtlinearen Relaxationsverfahrens

$$\overline{u}_h^{(k)} = \hat{\mathcal{R}}_h^{\nu_1}(u_h^{(k)}, N_h, f_h)$$

(2) Minimierung des eindimensionalen Unterproblems (5.3) mit Lösung ω_f

(3) Modifikation der Näherungslösung durch

$$\overline{\overline{u}}_h^{(k)} = u_h^{(k)} + \omega_f \cdot (\overline{u}_h^{(k)} - u_h^{(k)})$$

(4) Berechnung des Defekts

$$d_h = f_h - N_h(\overline{\overline{u}}_h^{(k)})$$

(5) Restriktion des Defekts auf das gröbere Gitter

$$d_H = I_h^H(d_h)$$

(6) Restriktion der Näherungslösung auf das gröbere Gitter

$$u_H = \hat{I}_h^H(\overline{\overline{u}}_h^{(k)})$$

(7) Berechnung einer Näherungslösung v_H der Grobgitterkorrekturgleichung

$$N_H(u_H + v_H) = d_H + N_H(u_H)$$

(8) Minimierung des eindimensionalen Unterproblems (5.6) mit Lösung ω_c

(9) Modifikation der Näherungslösung durch

$$\overline{v}_H = \omega_c v_H$$

(10) Interpolation der modifizierten Grobgitterkorrektur auf das feine Gitter

$$v_h = I_H^h(\overline{v}_H)$$

(11) Berechnung der korrigierten Näherung

$$\overline{\overline{\overline{u}}}_h^{(k)} = \overline{\overline{u}}_h^{(k)} + v_h$$

(12) Nachglättung durch ν_2 Iterationen eines Relaxationsverfahrens

$$\hat{u}_h^{(k)} = \hat{\mathcal{R}}_h^{\nu_2}(\overline{\overline{\overline{u}}}_h^{(k)}, N_h, f_h)$$

(13) Minimierung des eindimensionalen Unterproblems

$$\text{finde } \omega \in I\!R, \quad \text{so dass} \quad \omega = arg \min_{\overline{\omega} \in I\!R^{>0}} D(\overline{\overline{\overline{u}}}_h^{(k)} + \omega \cdot (\hat{u}_h^{(k)} - \overline{\overline{\overline{u}}}_h^{(k)}))$$

mit Lösung ω_f

(14) Modifikation der Näherungslösung durch

$$u_h^{(k+1)} = \overline{\overline{\overline{u}}}_h^{(k)} + \omega_f \cdot (\hat{u}_h^{(k)} - \overline{\overline{\overline{u}}}_h^{(k)}).$$

Dieser auf zwei Gittern beschriebene Algorithmus kann, genau wie das Standard Full Approximation Scheme in Kapitel 4.3.4, durch rekursives Lösen des Grobgitterproblems durch ein Zweigitterverfahren zu einem Mehrgitterverfahren erweitert werden.

5.4 Fortsetzungsverfahren

In Kapitel 5.3 wurde eine Iteration vorgestellt, die den Einfluss des Regularisierungsparameters auf die Lösung durch eine Korrektur reduziert. In diesem Abschnitt wird eine Idee verfolgt, die auf der folgenden Beobachtung basiert:
Für große Regularisierungsparameter α können Minimallösungen numerisch stabil mit einem FAS-Mehrgitterverfahren berechnet werden. Die berechnete Deformation bewirkt jedoch kaum Verschiebung im Templatebild, und der Wert des Defektfunktionals $D(u)$ wird nur wenig reduziert. Andererseits wird für kleine α die numerische Lösung mit einem FAS-Mehrgitterverfahren instabil, weil der Einfluss des Glättungsanteils im Tikhonov-Funktional abgeschwächt wird und somit auf den groben Gittern die verbleibenden hochfrequenten Fehleranteile nicht mehr vernünftig geglättet werden können.

Der beschriebenen Diskrepanz bei der Wahl des Regularisierungsparameters wird im

Folgenden mit der sukzessiven Annäherung an einen geeigneten Regularisierungsparameter begegnet. Entscheidend hierbei ist, dass Minimallösungen des Tikhonov-Funktionals T_{α_1} mit relativ großem α_1 zwar keine Minimallösungen des Defektfunktionals $D(u)$ sind, jedoch gute Startnäherungen bei der Minimierung des Tikhonov-Funktionals T_{α_2} mit einem Regularisierungsparameter $\alpha_2 < \alpha_1$. Um eine geeignete Minimallösung für das Defektfunktional $D(u)$ zu berechnen, wird dieser Prozess iteriert. Hierzu wird die Folge von Minimallösungen

$$\left\{ \min_{u \in \mathcal{X}} \{ \underbrace{||h(u)||^2_{L_2(\Omega)} + \alpha_k a(u - u^{(k)}, u - u^{(k)})}_{=:T_k(u)} \} \right\}_{k=0,1,2,\ldots}$$

der Tikhonov-Funktionale $T_k(u)$ mit $\alpha_0 > \alpha_1 > \alpha_2 > ...$ berechnet.

Das Hinzunehmen einer Startnäherung vermindert den Einfluss des Regularisierungsterms, weil die von $T_k(u)$ berechnete Minimallösung schon eine gute Näherung für eine Minimallösung des Tikhonov-Funktionals $T_{k+1}(u)$ ist. Deshalb kann der Regularisierungsparameter α groß gewählt werden, um Lösungen zu berechnen, für deren Berechnung ohne die Startnäherung (d.h. $u^{(k)} = 0$) der Parameter α entsprechend reduziert werden müsste. Für zu kleine Parameter α wird jedoch, wie bereits beschrieben, die Berechnung einer Lösung mit dem MG-FAS instabil. Durch die Wahl einer geeigneten Startnäherung können also Lösungen bei gleicher Glättung stabil berechnet werden, die sonst wegen der Instabilität des MG-FAS nicht erreichbar sind. Außerdem kann der Startwert $u^{(k)}$ hierbei auch als Auswahlkriterium für die Menge der Minimalstellen des Tikhonov-Funktionals $T_{k+1}(u)$ angesehen werden.

Dieser iterative Prozess kann als iterative Tikhonov-Regularisierung betrachtet werden, bei der die Lösung der k-ten Iteration durch die Lösung der $(k+1)$-ten Iteration fortgesetzt wird. Deshalb wird die Iteration auch als Fortsetzungsverfahren bezeichnet.

Für die numerische Realisierung wird für die Startnäherung $u^{(0)} \in \mathcal{X}$ (etwa $u^{(0)} = 0$ oder wie in Kapitel 5.5 beschrieben) und für $\alpha_0 \gg 0$ eine Minimallösung $u^{(1)}$ von $T_1(u)$ durch ein FAS-Mehrgitterverfahren näherungsweise bestimmt. Für den nächsten Iterationsschritt wird der Wert des Parameters α_0 durch einen Faktor $0 < \kappa < 1$ (zum Beispiel $\kappa = \frac{1}{2}$) reduziert und $u^{(1)}$ als Startnäherung für das Minimierungsproblem der nächsten Iteration verwendet.

Durch iterative Anwendung entsteht eine Folge von Verschiebungsvektoren

$\{u^{(k)}\}_{k=0,1,2,\ldots}$, die den Wert des Defektfunktionals $D(u)$ sukzessive reduzieren. Die Iteration wird beendet, falls der Wert einer Iterierten $u^{(k+1)}$ den aktuellen Wert des Defektfunktionals $D(u)$ nicht weiter verkleinert, also wenn $D(u^{(k)}) \leq D(u^{(k+1)})$ gilt. Zur weiteren Verbesserung der Lösung wird der Wert von α_{k+1} wieder erhöht (zum Beispiel $\alpha_{k+1} = \alpha_0$) und mit der Startnäherung $u^{(k)}$ die Iteration fortgesetzt. Dieser Vorgang wird solange wiederholt, bis entweder das Funktional einen festen Wert $D(u^{(k)}) \approx \eta$ erreicht hat oder eine feste Anzahl von Iterationen durchgeführt wurden.

Die beiden beschriebenen Iterationen werden in dem folgenden Algorithmus als innere und äußere Iteration bezeichnet und werden jeweils innerhalb einer repeat-until Schleife abgearbeitet.

Das Fortsetzungsverfahren in C-Notation:

$k = 0$; $u^{(k)} = u^*$; $\alpha_k = M \gg 0$;

// äußere Iteration

repeat

  // innere Iteration

  repeat

    $u^{(k+1)} = \min_{u \in \mathcal{X}} \{||h(u)||^2_{L_2(\Omega)} + \alpha_k ||u - u^{(k)}||^2_{\mathcal{X}}\}$

    $\alpha_{k+1} = \alpha_k \cdot \kappa$;

    $k = k + 1$;

  until $(D(u^{(k+1)}) > D(u^{(k)}))$

  // Die letzte (innere) Iteration war zuviel!!

  $k = k - 1$;

  $\alpha_k = \alpha_0$;

until ($(k > k_{max})$ || ($D(u^{(k)}) \approx \eta$))

5.5 Bestimmung geeigneter Startnäherungen

Die in den vorherigen Abschnitten beschriebenen Minimierungsverfahren für das Defektfunktional $D(u)$ bestimmen eine Lösung $u^* \in \mathcal{X}_h \subset \mathcal{X}$. Die Dimension des Raumes $\mathcal{X}_h$ ist bestimmt durch die Abtastschrittweite h der digitalen Bilder T^h und R^h, mit der in Kapitel 3.2 auch die partielle Differentialgleichung diskretisiert wurde.

Bei genauer Analyse erkennt man, dass hierbei zwei Probleme auftreten. Zum einen werden bei einer Schrittweite h die kleinen Strukturen (hohe Bildfrequenzen) bevorzugt angepasst, jedoch größere vernachlässigt. Das führt dazu, dass die Minimierungsverfahren in vielen Fällen unbrauchbare lokale Minimalstellen berechnen. Zum andern sind zum Erreichen dieser Minimalstelle viele Iterationschritte der Minimierungsverfahren notwendig, so dass der Minimierungsprozess gerade auf feinen Auflösungsstufen sehr lange dauert.

Zur Verbesserung wird die Full Multigrid (FMG) Idee (Brandt [7] und Stüben & Trottenberg [30]) herangezogen. Ziel des FMG Verfahren ist, eine Näherung für die diskrete Lösung eines Randwertproblems zu berechnen, deren Approximationsfehler kleiner ist als der globale Diskretisierungsfehler. Die Idee hierbei ist die Berechnung geeigneter Näherungen mit einem Mehrgitterverfahren auf gröberen Diskretisierungsebenen, die als Startnäherung auf die nächst feinere Gitterebene interpoliert werden.

Zur Minimierung des Defektfunktionals $D(u)$ wird die FMG Idee zur Berechnung geeigneter Startnäherungen für die Minimierungsverfahren auf die Problemstellung angepasst. Hierbei wird ausgenutzt, dass durch die Minimierungsverfahren die hochfrequenten Bildstrukturen bevorzugt angepasst werden.
Das Abtasttheorem der digitalen Bildverarbeitung [25] besagt, dass eine periodische Bildstruktur aus den Abtastwerten (Bildelementen) nur dann richtig rekonstruiert werden kann, wenn die kleinste Wellenlänge mindestens zweimal abgetastet wird. Das bedeutet, dass bei den Bilddatensätzen mit einer Abtastschrittweite h nur Bildstrukturen mit einer Größe von mindestens $2h$ präsent sind. Um größere Bildstrukturen anzupassen, werden diese auf gröbere Auflösungsstufen transformiert, auf denen sie gerade noch präsentiert werden können, also hochfrequent sind. Dieser Zusammen-

hang wird bei der Minimierung des Defektfunktionals ausgenutzt, indem die Bilder $T, R \in \mathcal{Y}$ auf eine Folge von endlich dimensionalen Räumen

$$\mathcal{Y}_{Lh} \subset \mathcal{Y}_{(L-1)h} \subset ... \subset \mathcal{Y}_h \subset ... \subset \mathcal{Y}$$

transformiert werden. Der Informationsgehalt der Bilddaten wird hierbei immer geringer. Auf gröberen Auflösungsstufen wird der Aufwand zur Bestimmung einer Lösung drastisch reduziert, und die Größe der anzupassenden Strukturen kann durch die Abtastschrittweite gesteuert werden. Zur Darstellung der Deformationen auf den gröberen Auflösungsstufen wird der Raum $\mathcal{X}$ durch eine Folge endlich dimensionaler Unterräume mit der Eigenschaft

$$\mathcal{X}_{Lh} \subset \mathcal{X}_{(L-1)h} \subset ... \subset \mathcal{X}_h \subset ... \subset \mathcal{X}$$

analog zu $\mathcal{Y}$ zerlegt. Hierdurch erhält man, je nach Minimierungsansatz, eine Folge von Minimierungproblemen für das Tikhonov-Funktional

$$\left\{ \min_{u_l \in X_l} \{||h(u_l)||_2^2 + \alpha a(u_l, u_l)\} \right\}_{l=1,2,...,L}$$

oder eine Folge von Minimierungsproblemen für das Defektfunktional

$$\left\{ \min_{u_l \in X_l} D(u_l) \right\}_{l=1,2,...,L,}$$

die jeweils durch die oben vorgestellten Verfahren gelöst werden. Die Restriktion des Defektfunktionals $D(u)$ erfolgt durch die Restriktion der digitalen Bilder auf eine gröbere Bildauflösung durch den Operator $I_{lh}^{(l-1)h} : \mathcal{Y}_{lh} \to \mathcal{Y}_{(l-1)h}$. Die Lösung $u_l \in X_l$ wird durch einen Interpolationsoperator $I_{(l-1)h}^{l} : \mathcal{X}_{lh} \to \mathcal{X}_{(l-1)h}$ auf die nächst feinere Auflösungsstufe projiziert. $I_{l-1}^{l}(u_l) \in X_{l-1}$ ist hierbei eine geeignete Startnäherung zur Minimierung des Defektfunktionals.

Minimierungsprozess von $D(u)$
auf mehreren Auflösungsstufen in C-Notation:

$u_L = 0; l = L;$

repeat

if $(l = L)$

berechne u_L^* mit der Startnäherung u_L;

$l = l - 1;$

else

$u_l = I_{i-1}^i(u_{l+1}^*);$

berechne u_l^* mit der Startnäherung u_l;

endif

$l = l - 1;$

until l = 0

5.6 Ergebnisse

Zum Abschluss des Kapitels sollen die Algorithmen zur Minimierung des Defektfunktionals $D(u)$ an einigen synthetischen und realen Bilddaten getestet werden. Zunächst soll das auf der Berechnung von Abstiegsrichtungen basierende Verfahren aus Kapitel 5.1 anhand von synthetischen Bilddaten getestet werden. Hierfür sind links in Abbildung 5.1 das Templatebild $T(x)$ und in Abbildung 5.3 rechts das Referenzbild $R(x)$ gegeben. Die Abbildungen 5.1, 5.2 und 5.3 zeigen den Verlauf der Iteration anhand der Bilddaten auf. In jedem Iterationsschritt wird das Templatebild durch die aktuelle Approximation der Lösung verändert. Hierdurch wird eine Bildfolge erzeugt, die sich im Laufe der Iteration immer weiter an die gewünschte Position des Referenzbildes anpasst. Um die Wirkungsweise der Verschiebungsvektoren besser einsehen zu können, werden sie auf ein äquidistantes Gitter angewendet (vgl. Abbildung 5.4, 5.5 und 5.6).
Zur Berechnung der Verschiebungsvektoren wurde in jedem Iterationsschritt ein $V(1,1)$-MG-CS Zyklus mit maximaler Gitteranzahl verwendet. Der Graph in Abbildung 5.7 zeigt den Verlauf des Defektfunktionals $D(u)$ für den Anpassungsprozess der synthetischen Bilddaten. Hierbei wird deutlich, dass mit jeder Iteration der Wert von $D(u)$ reduziert wird. Der Graph ist durch den Aufwand eines Iterationschrittes (engl. work unit (WU)) skaliert, so dass gleiche Intervalle auf der x-Achse etwa den gleichen Rechenzeiten entspricht. In den ersten Iterationen erkennt man, dass die Iterationen den Wert von $D(u)$ deutlich verringern, während im weiteren Verlauf der Iteration kaum noch Veränderungen erkennbar sind.

Zur Beschleunigung der Bildanpassung werden geeignete Startnäherungen für den Minimierungsprozess auf der feinsten Auflösungsstufe (vgl. Kapitel 5.5) bestimmt. Hierzu werden die Bilder T und R auf gröbere Auflösungsstufen transformiert und hier das Defektfunktional $D(u)$ minimiert. Die Originalbilder haben 128×128 Bildpunkte. Der Anpassungsprozess wird mit der Reduktion der Bilddaten auf 32×32 Bildpunkte gestartet. Auf dieser Auflösungsstufe wird ein Verschiebungsvektor $u_{1/32} \in \mathcal{X}_{1/32}$ berechnet, der auf die nächst feinere Auflösungsstufe mit 64×64 Bildpunkten interpoliert wird. Mit dieser Startnäherung wird der Anpassungsprozess erneut gestartet und die resultierende Lösung auf die Auflösungsstufe der Orginalbilder

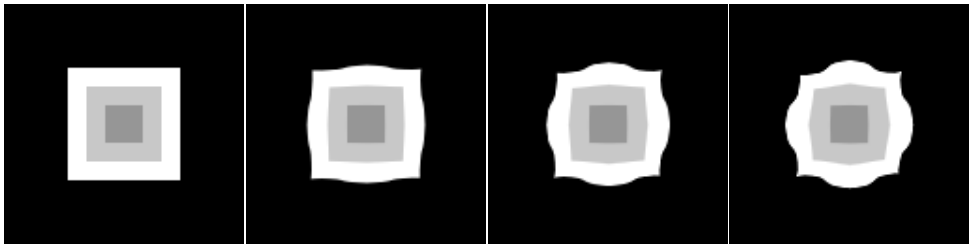


Abbildung 5.1: Von links nach rechts: Template $T(x)$ und die Zwischenergebnisse nach 1, 2 und 3 Iterationen.

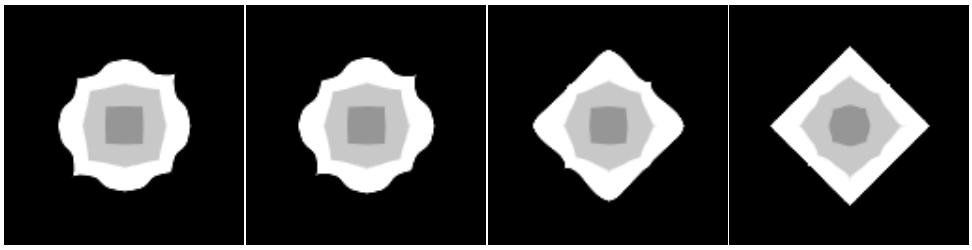


Abbildung 5.2: Zwischenergebnisse nach 4, 5, 10 und 20 Iterationen.

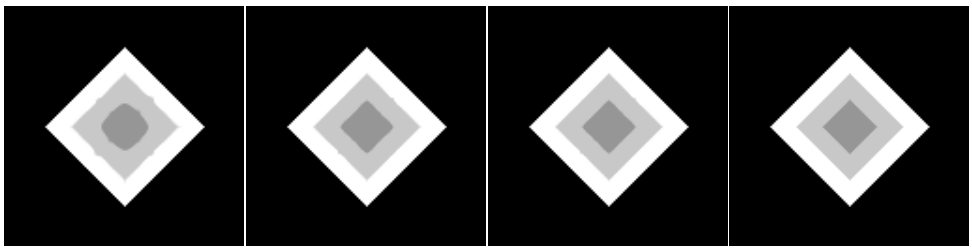


Abbildung 5.3: Zwischenergebnisse nach 30, 40 und 50 Iterationen. Rechts das Referenzbild $R(x)$.

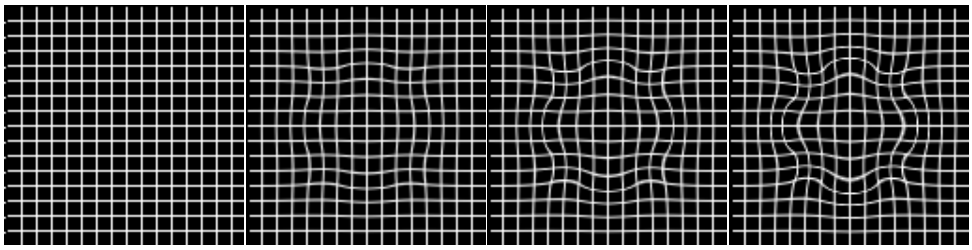


Abbildung 5.4: Verschiebungsvektoren angewandt auf das äquidistante Gitter (ganz links) nach der ersten, zweiten und dritten Iteration.

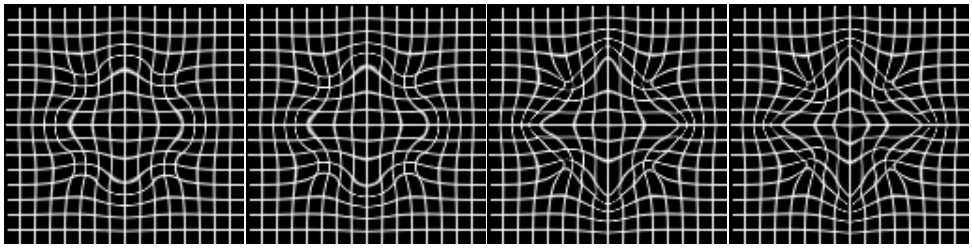


Abbildung 5.5: Das äquidistante Gitter nach 4, 5, 10 und 20 Iterationen.

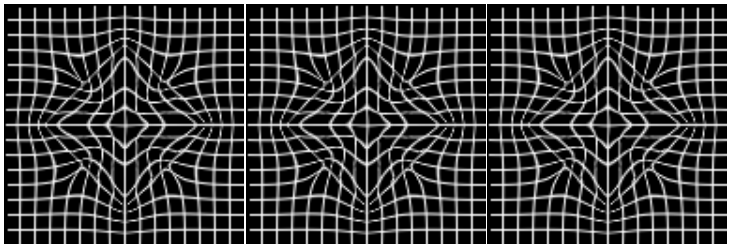


Abbildung 5.6: Das äquidistante Gitter nach 30, 40 und 50 Iterationen.

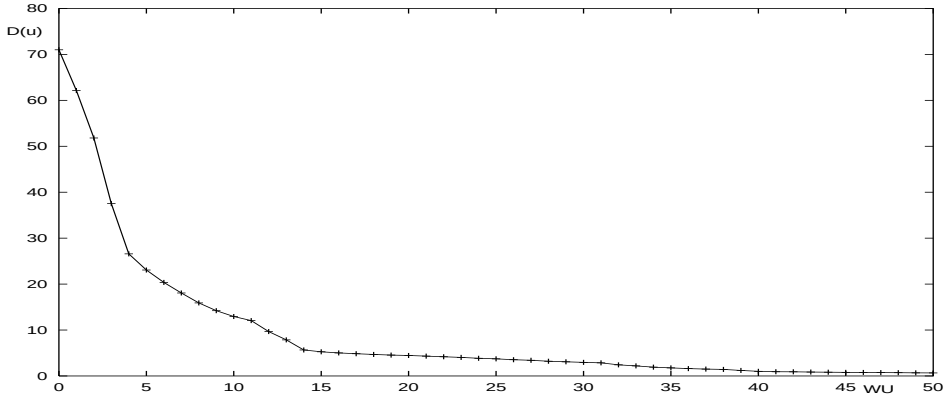


Abbildung 5.7: Verlauf des Defektfunktionals $D(u)$ während der Iteration.

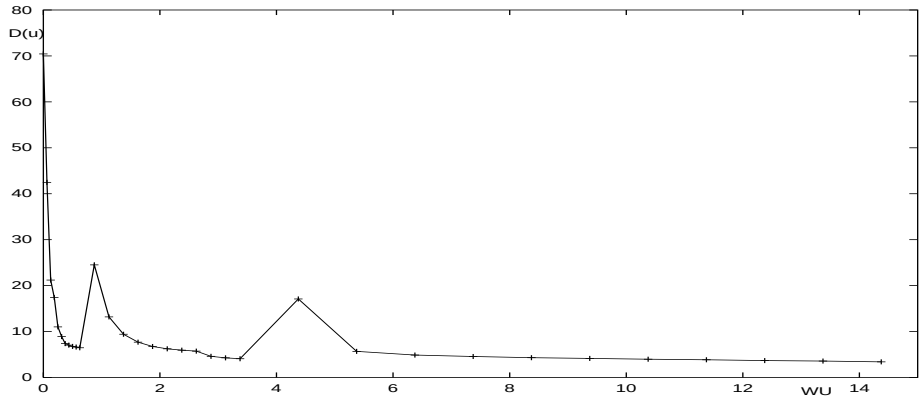


Abbildung 5.8: Verlauf des Defektfunktionals $D(u)$ während des Minimierungsprozess auf mehreren Bildauflösungsstufen.

transformiert. Hier wird der Minimierungsprozess nach wenigen Iterationsschritten beendet. In Abbildung 5.8 ist der entsprechende Graph des Minimierungsprozesses skizziert. Die maximale Anzahl von Iterationen wurde hierbei auf $k_{max} = 10$ begrenzt.

Das nächste Beispiel (vgl. Abb. 5.9) demonstriert das Verfahren anhand zweier Schichtdatensätze, die mit Hilfe der Magnet-Resonanz-Tomographie (MRT) erstellt wurden. Die Berechnung der Verschiebungsvektoren erfolgt hierbei wie im vorherigen Beispiel auf drei Bildauflösungsstufen mit maximaler Anzahl von Iterationen $k_{max} = 10$. Die Verschiebungsvektoren wurden mit einem $V(1,1)$-MG-CS Zyklus mit maximaler Gitteranzahl berechnet.

Die Abbildungen 5.11, 5.12 und 5.13 zeigen ein MRT-Referenz- bzw. MRT-Templatebild und das Ergebnis der dreidimensionalen Bildanpassung. Durch die perspektivische Darstellung der Bilder mit Hilfe einer künstlichen Lichtquelle ist gut zu erkennen, dass sowohl die Oberfläche als auch die inneren Strukturen des Templatebildes nahezu komplett auf das Referenzgehirn transformiert wurden.
Die Berechnung der Verschiebungen erfolgt wie im vorherigen Beispiel auf drei verschiedenen Auflösungsstufen. Die maximale Anzahl von Iterationen auf den verschiedenen Auflösungsstufen wurde hierbei auf $k_{max} = 50$ begrenzt.

Die in den Abbildungen 5.8 und 5.10 dargestellten Graphen für die Werte des Defektfunktionals $D(u)$ während des Anpassungsprozesses auf verschiedenen Bildauf-

lösungsstufen, weisen beim Übergang zwischen zwei Bildauflösungsstufen einen deutlichen Anstieg der Werte des Defektfunktionals auf. Dies ist ein typisches Phänomen bei der Berechnung von Näherungslösungen auf gröberen Gitterebenen. Die gleiche Erscheinung tritt auch beim Full Multigrid (FMG) Verfahren, zur Berechnung von Näherungslösungen bei partiellen Differentialgleichungen auf gröberen Gitterebenen auf.

Der Grund hierfür ist, dass die berechnete Näherungslösung das Problem mit den auf groben Gittern präsenten Informationen löst. Die feinen Bildstrukturen bleiben hierbei unberücksichtigt und können erst auf den feinen Gittern angepasst werden. Die gleiche Beobachtung kann im Folgenden, nämlich bei der Minimierung des Defektfunktionals auf mehreren Auflösungsstufen durch die nichtlinearen Verfahren, gemacht werden. Um das Prinzip und die Stabilität der in Kapitel 5.3 und 5.4 beschriebenen nichtlinearen Verfahren zur Anpassung der Bilddaten zu verdeutlichen, werden zunächst die synthetischen Bilddaten aus dem vorherigen Abschnitt verwendet. In Abbildung 5.14 ist jeweils das Resultat des Minimierungsprozesses durch das L-Kurven Verfahren, das modifizierte MG-FAS und für das Fortsetzungsverfahren sowie die entsprechenden Verschiebungen skizziert. Während die Lösung mit dem L-Kurven Verfahren (mit der L-Kurve in Abb. 5.15) nur auf der feinsten Auflösungsstufe berechnet wurde, wurden die übrigen Verfahren nacheinander auf einer Folge von Räumen

$$X_{1/32} \subset X_{1/64} \subset X_{1/128}$$

ausgeführt. Auf jeder Auflösungsstufe erhält man eine Folge von Lösungen $\{u^{(k)}\}_{0,1,..}$, die sukzessive den Wert des Defektfunktionals reduzieren. Dies wird durch die Graphen in den Abbildungen 5.16 und 5.17 deutlich. Allein nach der Transformation der Lösungsvektoren auf eine feinere Auflösungsstufe wächst der Wert von $D(u)$ an, was, wie beschrieben, an dem erhöhten Informationsgehalt der Bilddaten liegt. Die Graphen sind durch den Aufwand FAS-Mehrgitteriteration (WU) auf den entsprechenden Auflösungsstufen skaliert. Eine WU des Fortsetzungsverfahrens 5.17 entspricht hierbei etwa $\frac{1}{3}$ WU des modifizierten FAS.

Zur Abschätzung der Anpassungsqualität bei MRT Schnittbildern werden die Anpassungsalgorithmen auf das in Abbildung 5.18 dargestellte Templatebild und das in Abbildung 5.19 dargestellte Referenzbild angewendet. Abbildung 5.20 zeigt das mit

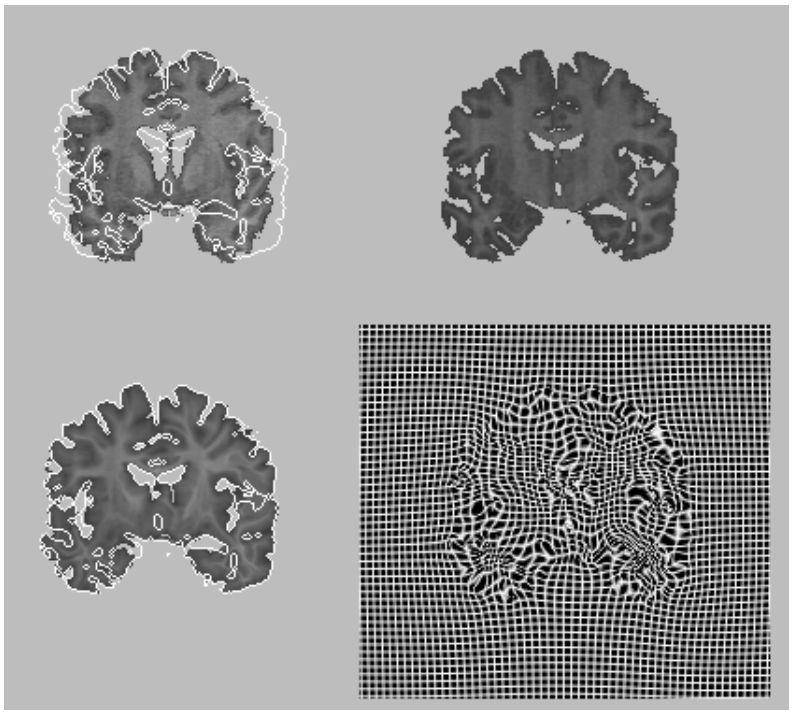


Abbildung 5.9: Oben links: Templatebild mit überlagerter Kontur des Referenzbildes. Oben rechts: Das Referenzbild $R(x)$. Unten links: Templatebild nach der Anpassung mit überlagerter Kontur des Referenzbildes. Unten rechts: Das Verschiebungsfeld angewendet auf ein äquidistantes Gitter.

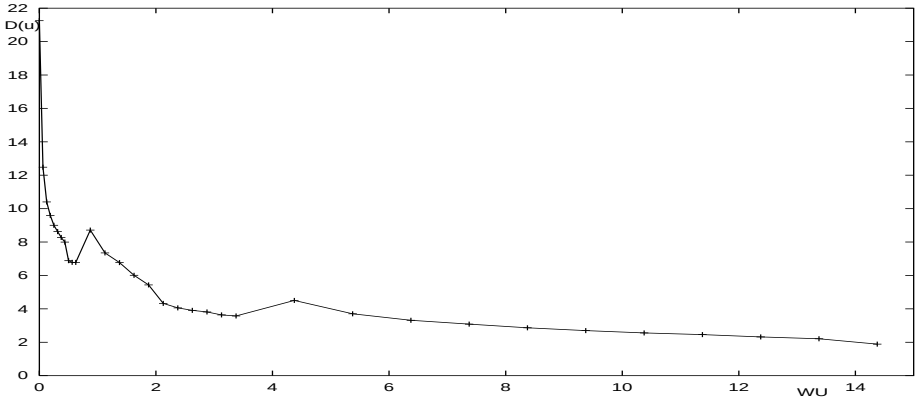


Abbildung 5.10: Verlauf des Defektfunktionals $D(u)$ während des Minimierungsprozesses der Bilddaten aus Abbildung 5.9 auf mehreren Bildauflösungsstufen.

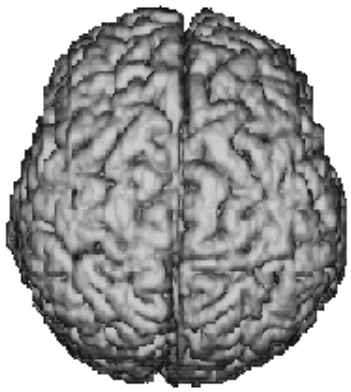


Abbildung 5.11: Perspektivische Darstellung des dreidimensionalen MRT-Referenzbildes.

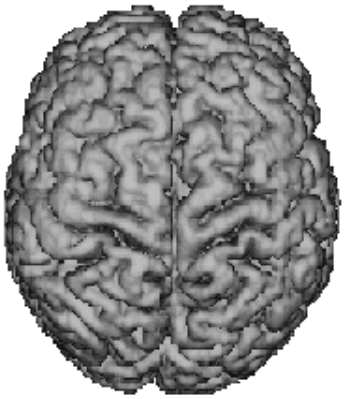


Abbildung 5.12: Perspektivische Darstellung des dreidimensionalen MRT-Templatebildes.

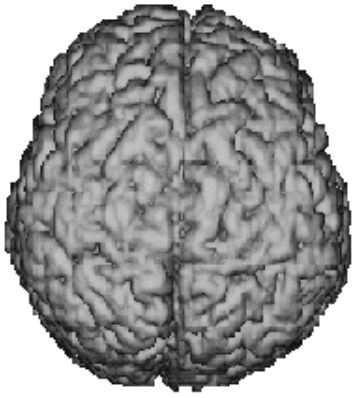


Abbildung 5.13: Perspektivische Darstellung des durch die berechnete Verschiebung transformierten Templatebildes aus Abbildung 5.12.

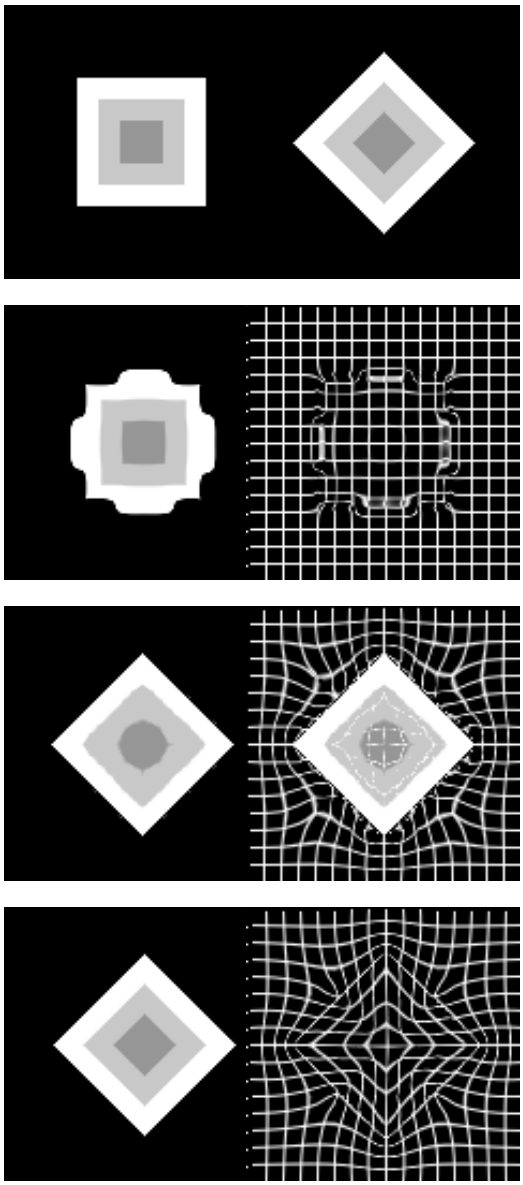


Abbildung 5.14: Von oben nach unten und von links nach rechts: 1. Templatebild $T(x)$. 2. Referenzbild $R(x)$. 3. Templatebild, deformiert durch die Minimallösung des Tikhonov-Funktionals mit dem durch das L-Kurven Kriterium (vgl. Kapitel 5.2) vorgeschlagenen Regularisierungsparameter. 4. Durch die Lösung deformiertes Gitter. 5. Templatebild, deformiert durch die mit dem modifizierten FAS-MG (vgl. Kapitel 5.3) berechnete Lösung. 6. Deformiertes Templatebild, überlagert mit dem deformierten Gitter. 7. Templatebild deformiert durch die mit dem Fortsetzungsverfahren (vgl. Kapitel 5.4) berechnete Lösung. 8. Durch die Lösung deformiertes Gitter.

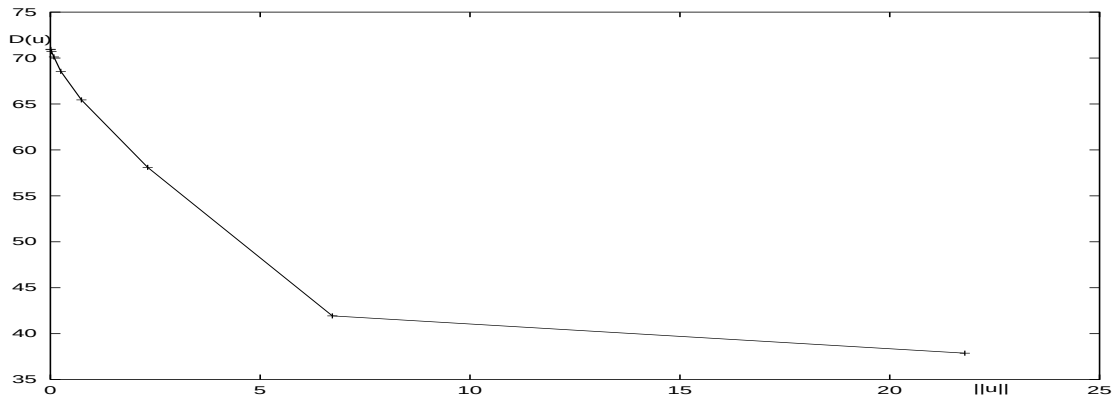


Abbildung 5.15: Verlauf der L-Kurve für das Beispiel aus Abb. 5.14.

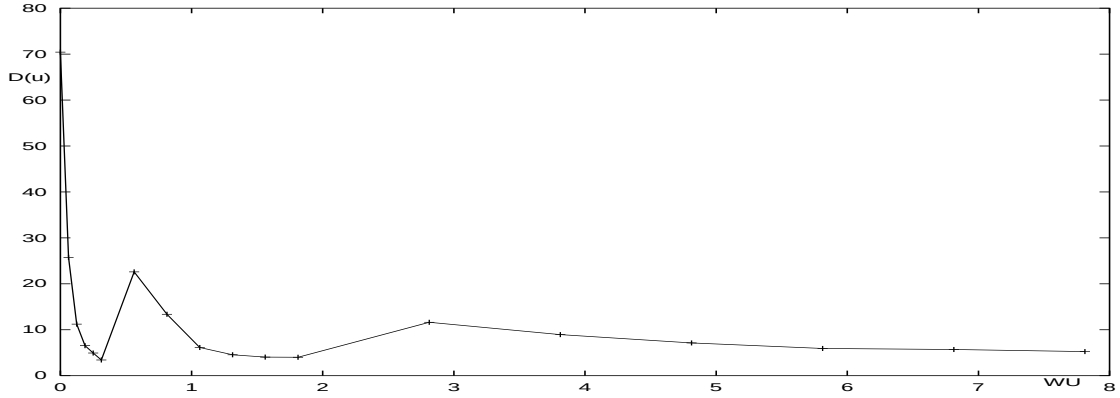


Abbildung 5.16: Verlauf des Defektfunktionals $D(u)$ für das Beispiel in Abb. 5.14 (mitte), berechnet mit dem modifizierten FAS.

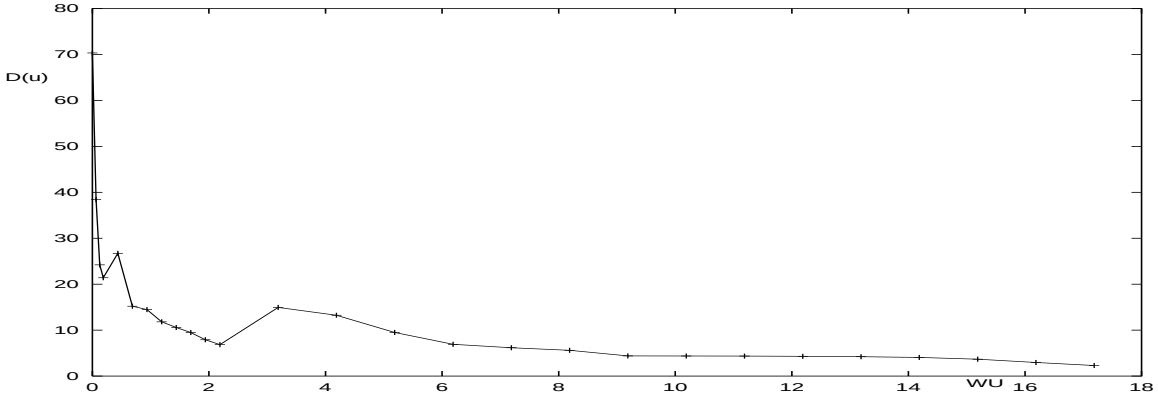


Abbildung 5.17: Verlauf des Defektfunktionals $D(u)$ für das Beispiel in Abb. 5.14 (unten), berechnet mit dem Fortsetzungsverfahren.

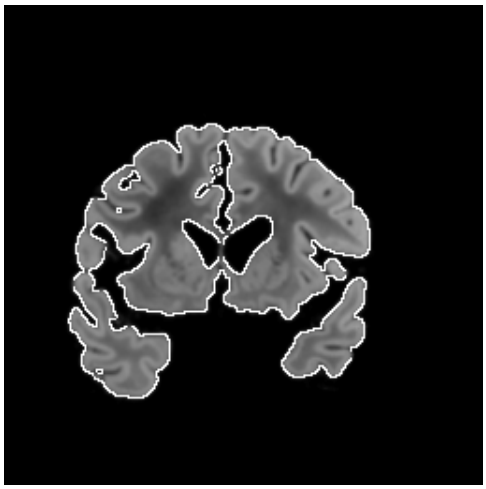


Abbildung 5.18: MRT Referenzbild mit überlagerter Bildkontur.

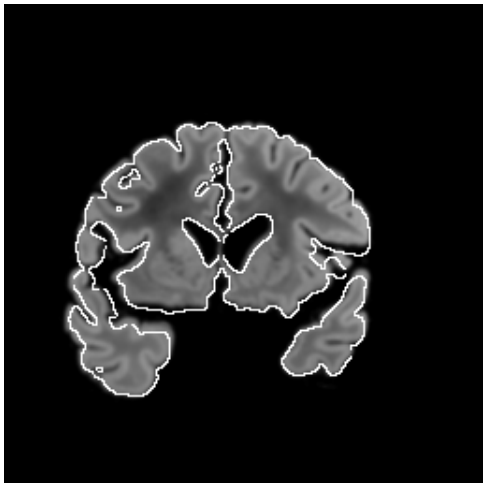


Abbildung 5.19: MRT Templatebild mit überlagerter Kontur des Referenzbildes in Abbildung 5.18.

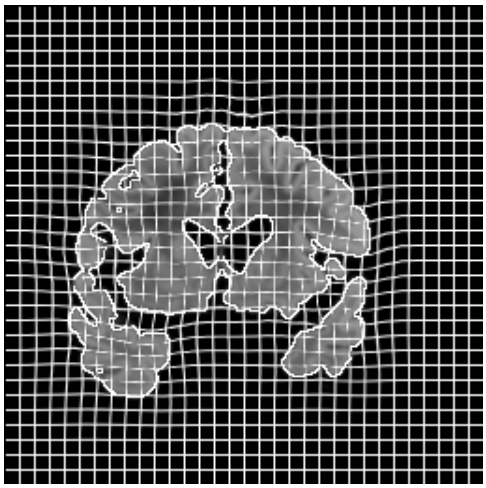


Abbildung 5.20: Deformiertes Templatebild mit überlagerter Kontur des Referenzbildes und überlagertem deformiertem Gitter.

dem Fortsetzungsverfahren berechnete Ergebnisbild. Die Berechnung der Ergebnisse erfolgte wieder auf drei Auflösungsstufen $X_{1/64} \subset X_{1/128} \subset X_{1/256}$. Die durch die WU skalierten Graphen in den Abbildungen 5.21 und 5.22 zeigen das im Laufe der Iteration abfallende Defektfunktional $D(u)$.
Das Fortsetzungsverfahren wurde in den beiden vorgestellten Beispielen mit einem Regularisierungsparameter von $\alpha = 100$ gestartet. Am Ende eines inneren Iterationsschritts wurde der Wert von α halbiert ($\kappa = \frac{1}{2}$). Dass dies eine gute Wahl für den Reduzierungsfaktor κ ist, zeigen die Graphen in Abbildung 5.23 und Abbildung 5.24. In Abbildung 5.23 wird die Entwicklung der Werte des Defektfunktionals für unterschiedliche Strategien bei der Wahl von κ für die erste innere Iteration im Fortsetzungsverfahren aufgezeigt, solange das Abbruchkriterium nicht erfüllt ist, also $D(u^{(k)} > D(u^{(k+1)})$ gilt. Die Werte entstanden bei der Anpassung der synthetischen Bilddaten aus Abbildung 5.14. Hierbei wird deutlich, dass für zu kleine κ und damit auch sehr stark fallendem α das Verfahren schnell in Bereiche kommt, in denen die Lösungen nicht mehr stabil berechnet werden können. Für größere κ nähert sich das Verfahren langsamer an den kritischen Bereich und das Verfahren wird zudem durch die Lösung der vorherigen Iteration regularisiert, so dass auch für kleine α eine geeignete Lösung berechnet werden kann. Wählt man κ hingegen zu groß, so nähert man sich nur sehr langsam an eine geeignete Lösung.
In Abbildung 5.24 sind die Graphen für verschiedene Strategien zur Reduzierung von α innerhalb des Fortsetzungsverfahrens mit mehreren inneren Iterationen angegeben. Wählt man κ nahe bei eins, so reduziert sich der Wert des Defektfunktionals nur sehr langsam. Andererseits reduziert sich der Wert des Defektfunktionals bei kleinen κ am Anfang sehr schnell. Das Verfahren verläuft sich jedoch nach einigen Iterationsschritten in eine lokale Minimalstelle. Bei der Wahl von $\kappa \approx \frac{1}{2}$ (in Abb. 5.24 $\kappa = \frac{1}{3}$ bzw. $\kappa = \frac{3}{4}$) wird der Wert des Defektfunktionals nach etwa 50 inneren Iterationsschritten auf $D(u) < 1.00$ reduziert.

Die Stabilität des Fortsetzungsverfahrens wird bei der Anpassung von gestörten Bilddaten deutlich. Die Abbildungen 5.25-5.29 zeigen jeweils das durch Impulsrauschen (Salt & Pepper Noise) gestörte Referenzbild, sowie das Ergebnis nach der Anpassung an das Templatebild aus Abbildung 5.14 und die auf ein äquidistantes Gitter angewandte Verschiebung. Zusätzlich ist für alle Beispiele der Graph mit den Werten des

Defektfunktionals während der inneren Iteration skizziert. Hierbei ist zu beobachten, dass im Laufe der Iteration der Wert des Defektfunktionals für alle Beispiele nach ca. 60 inneren Iterationen bis auf das Niveau des Datenfehlers $||R(x) - R^{\delta}(x)|| = \delta$ abfällt. Das Datenfehlerniveau ist durch eine Gerade parallel zur x-Achse aufgetragen. Es sei hierbei betont, dass die künstlich erzeugten Störungen in der Praxis durch Bildverarbeitungsalgorithmen weitgehend eliminiert werden können und in dieser Arbeit ausschließlich zur Illustration der Stabilität der Verfahren dienen.
In den Abbildungen 5.30, 5.31 und 5.32 ist das Referenzbild durch einen linearen Glättungsfilter der Breite (3,5 und 7) unterschiedlich stark verschmiert worden. Bei der Minimierung der resultierenden Bilddaten sinkt der Wert des Defektfunktionals deutlich unter das Niveau des Datenfehlers.
In Abbildung 5.33 ist das verschmierte Referenzbild aus Abbildung 5.30 mit zusätzlichem Impulsrauschen überlagert worden. Auch hier sinkt bei der Minimierung der resultierenden Bilddaten der Wert des Defektfunktionals unter das Niveau des Datenfehlers.
Die Anpassung von MRT Schnittbildern mit gestörtem Referenzbild ist in den Abbildungen 5.38 bis 5.40 demonstriert. Trotz der 2 % Impulsrauschen innerhalb des Referenzbildes kommen, die Bilddaten vollständig zur Deckung. Allein bei der Anpassung des rechten Temporallappens kommt es zu kleineren Störungen der Verschiebungen. Insgesamt ist zu erkennen, dass das Fortsetzungsverfahren die gesuchten Systemparameter (die Verschiebungen), trotz der teilweise erheblichen Störungen des Referenzbildes, gut rekonstruiert und damit die Bilddaten beinahe vollständig aufeinander transformiert werden. Zu beobachten ist hierbei, dass das hochfrequente Impulsrauschen dem Verfahren größere Probleme bereitet als Verschmierungen innerhalb der Bilddaten.

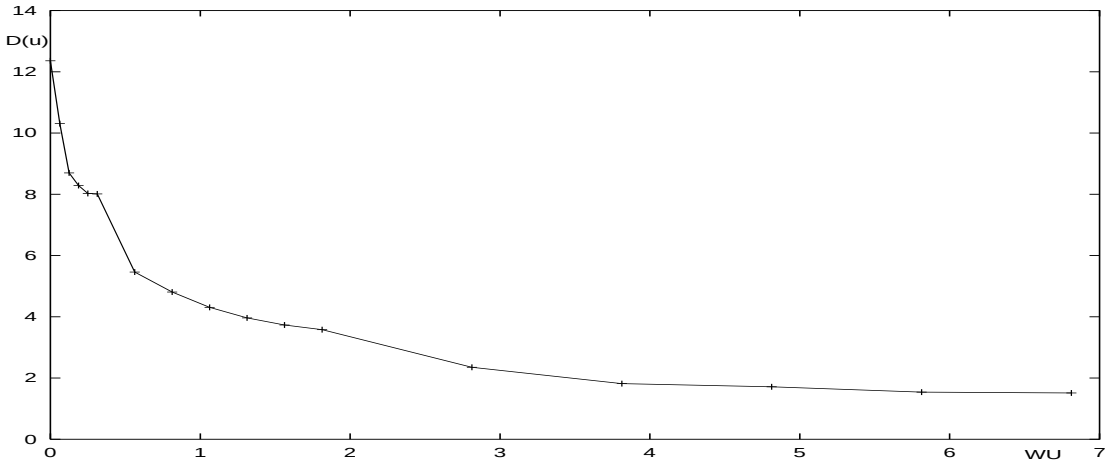


Abbildung 5.21: Verlauf des Defektfunktionals $D(u)$ für das Beispiel aus Abb. 5.18-5.20, berechnet mit dem modifizierten FAS.

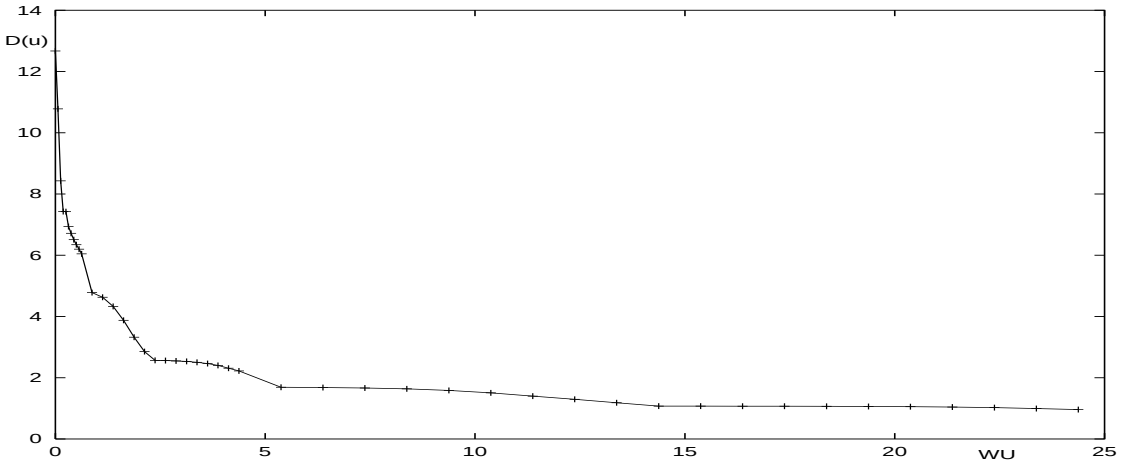


Abbildung 5.22: Verlauf des Defektfunktionals $D(u)$ für das Beispiel aus Abb. 5.18-5.20, berechnet mit dem Fortsetzungsverfahren.

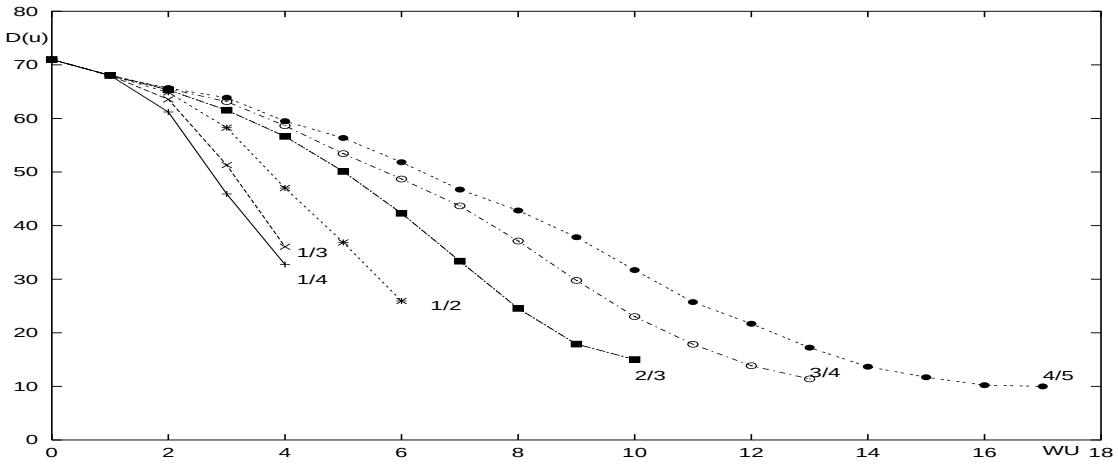


Abbildung 5.23: Verlauf des Defektfunktionals während der ersten inneren Iteration des Fortsetzungsverfahren, mit unterschiedlichen Strategien ($\kappa = \frac{1}{4}, \frac{1}{3}, \frac{1}{2}, \frac{2}{3}, \frac{3}{4}, \frac{4}{5}$) zur Reduzierung des Regularisierungsparameters α.

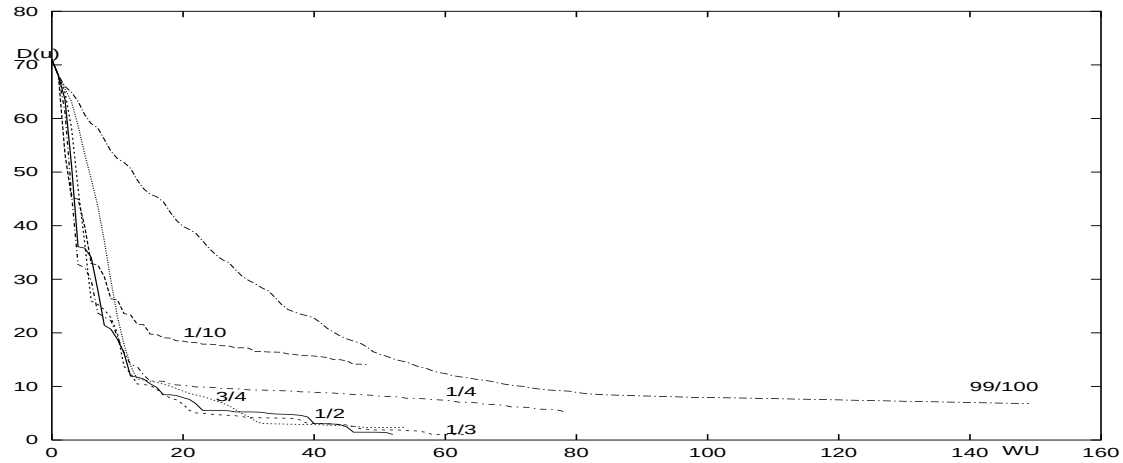
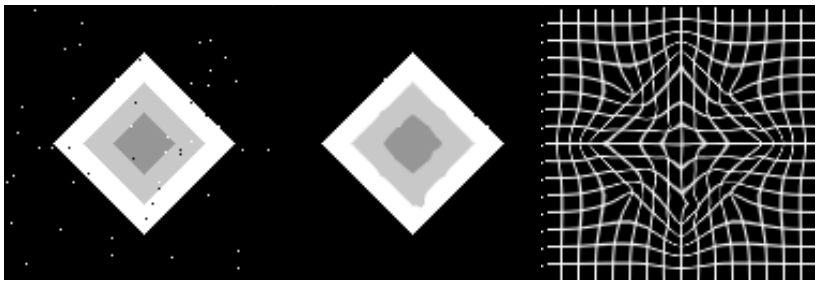


Abbildung 5.24: Verlauf des Defektfunktionals $D(u)$ für das Beispiel aus Abb. 5.14 für verschiedene Reduzierungsfaktoren κ innerhalb des Fortsetzungsverfahrens.



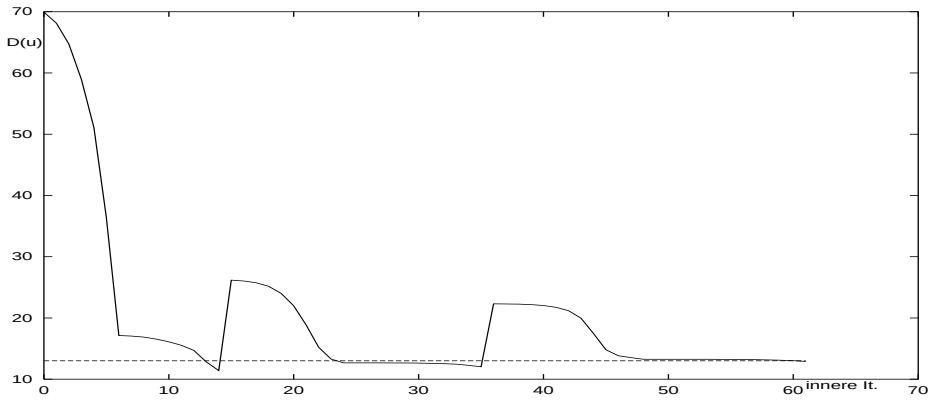
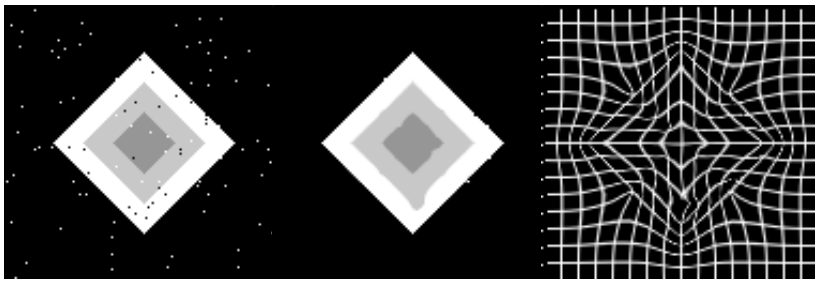


Abbildung 5.25: Fortsetzungsverfahren für Referenzdatensatz mit etwa 0.5 % Impulsrauschen.



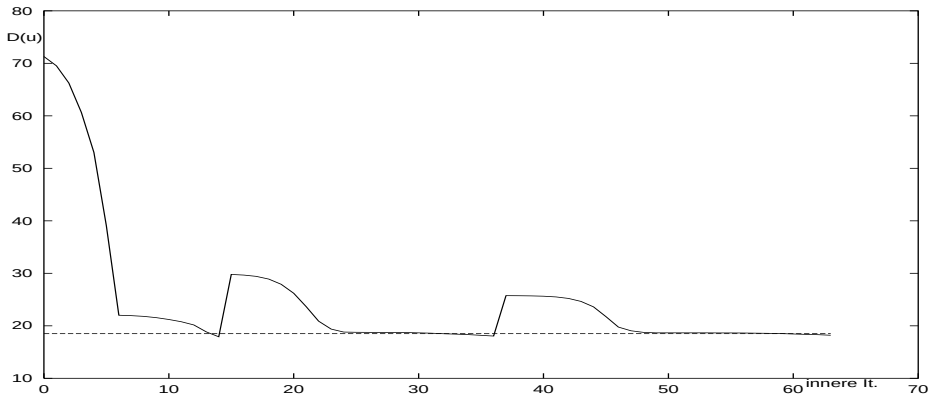


Abbildung 5.26: Fortsetzungsverfahren für Referenzdatensatz mit etwa 1.0 % Impulsrauschen.

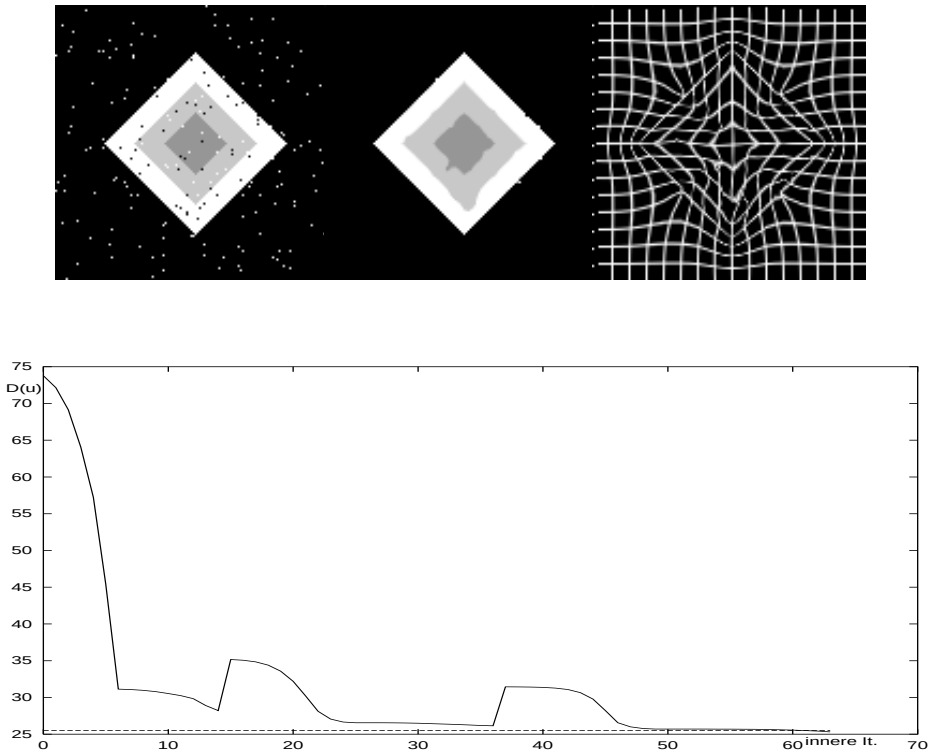


Abbildung 5.27: Fortsetzungsverfahren für Referenzdatensatz mit etwa 2.0 % Impulsrauschen.

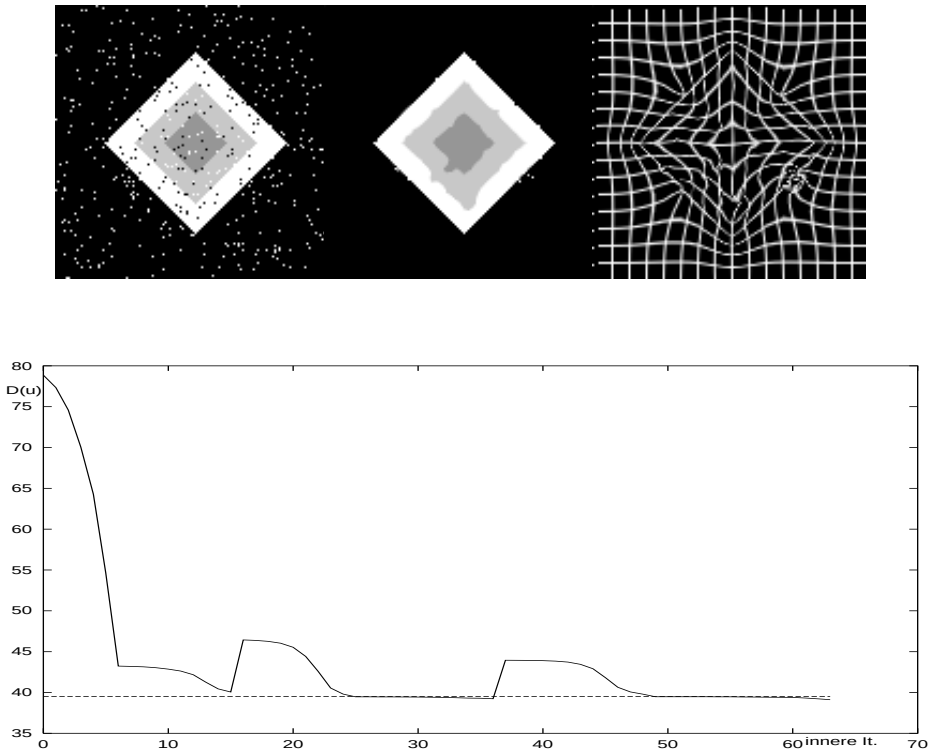
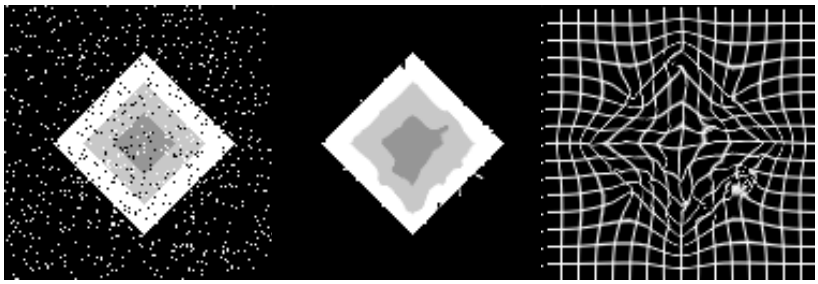


Abbildung 5.28: Fortsetzungsverfahren für Referenzdatensatz mit etwa 5.0 % Impulsrauschen.



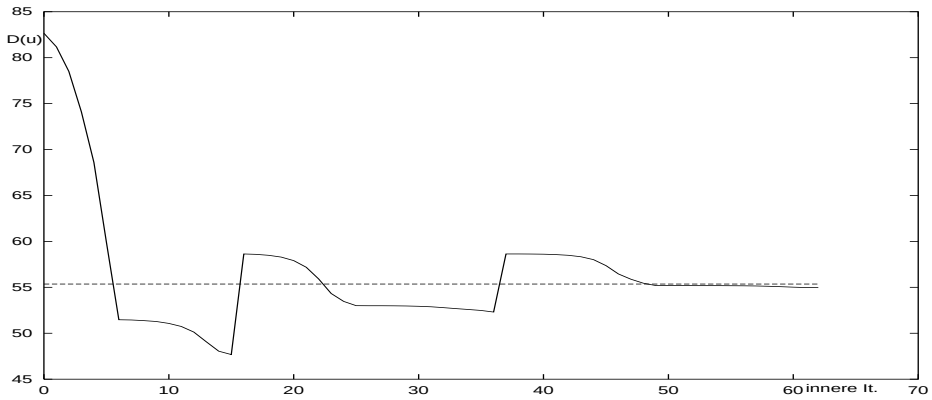
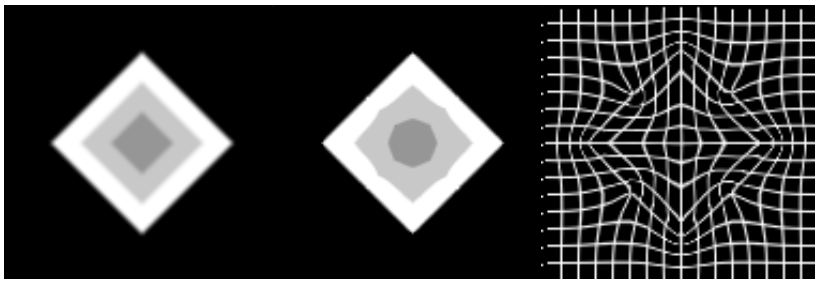


Abbildung 5.29: Fortsetzungsverfahren für Referenzdatensatz mit etwa 10.0 % Impulsrauschen.



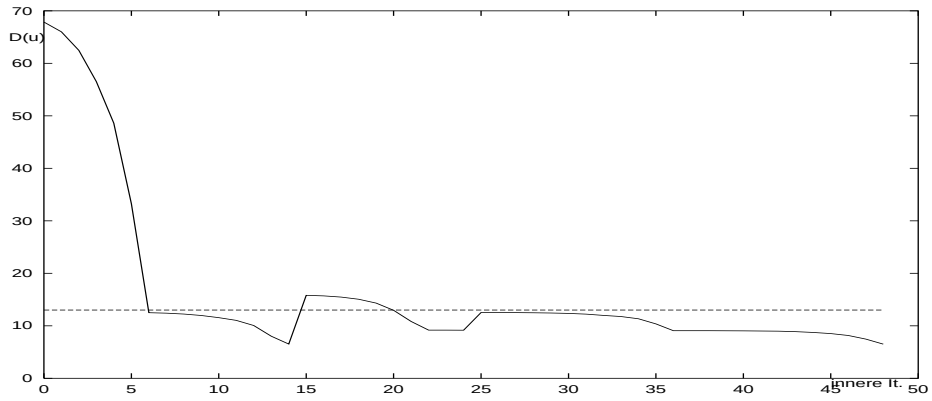


Abbildung 5.30: Fortsetzungsverfahren für leicht verschmierten Referenzdatensatz.

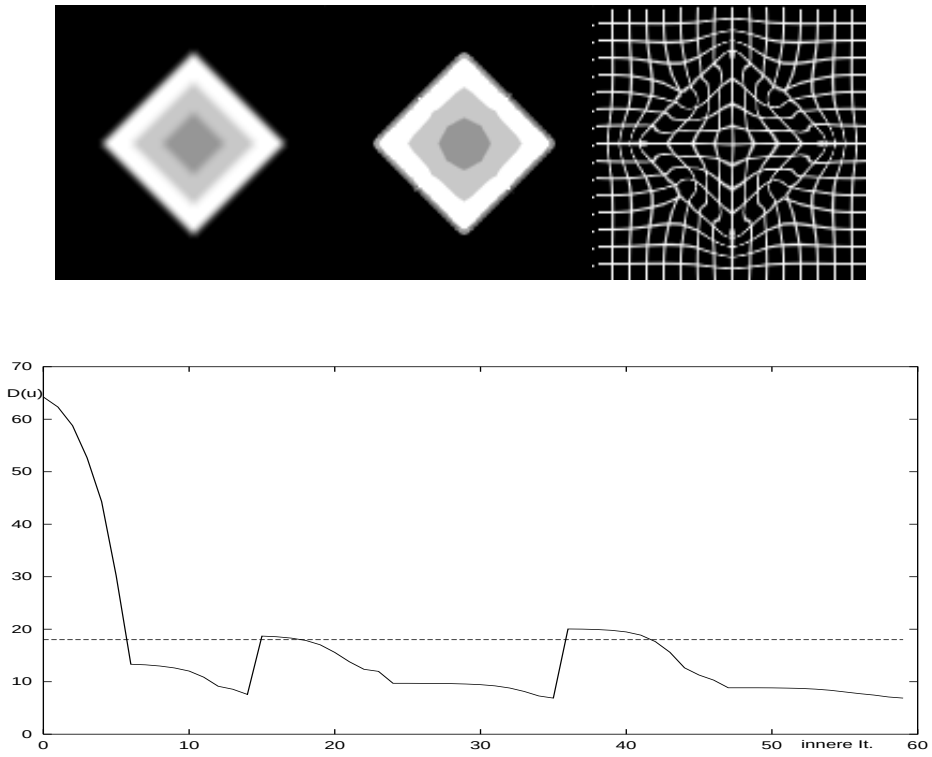


Abbildung 5.31: Fortsetzungsverfahren für verschmierten Referenzdatensatz.

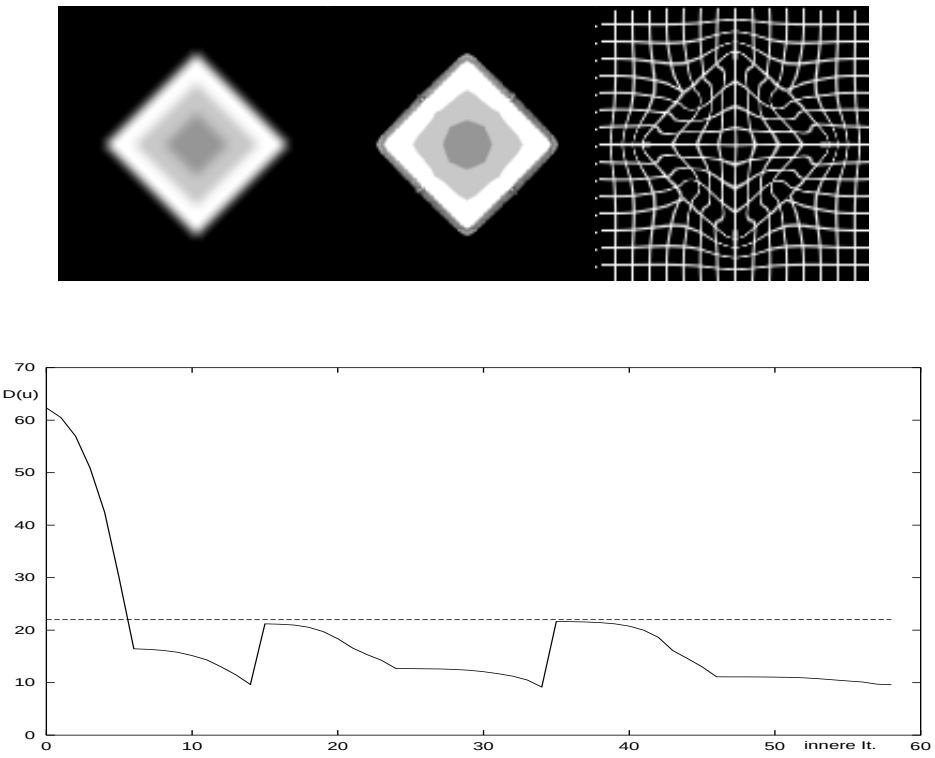
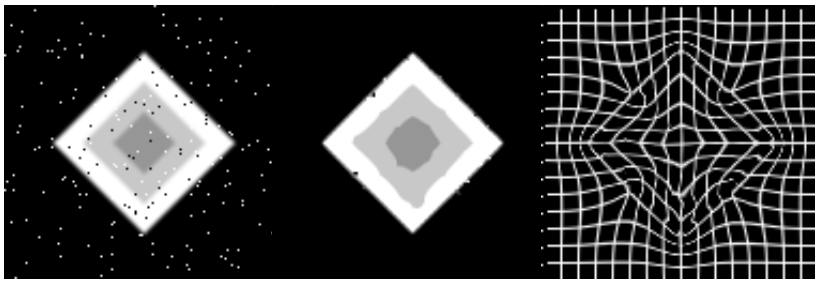


Abbildung 5.32: Fortsetzungsverfahren für stark verschmierten Referenzdatensatz.



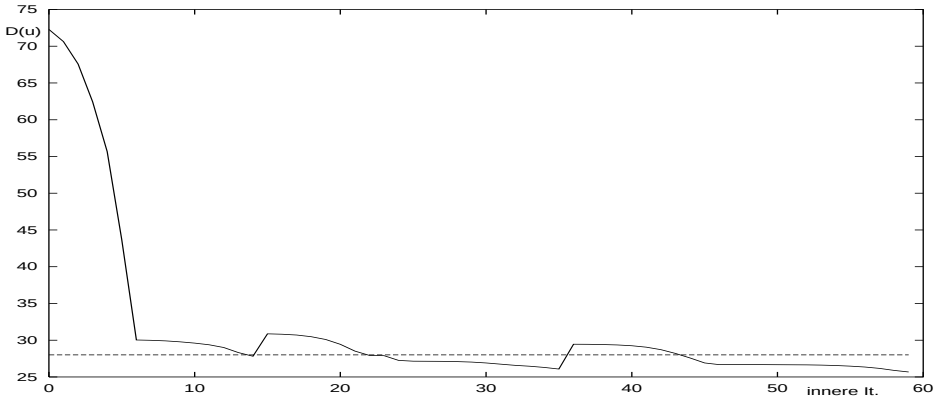


Abbildung 5.33: Fortsetzungsverfahren für leicht verschmierten Referenzdatensatz mit etwa 2.0 % Impulsrauschen.

Abbildung 5.34: Durch etwa 2 % Impulsrauschen gestörtes MRT Referenzbild mit überlagerter Bildkontur.

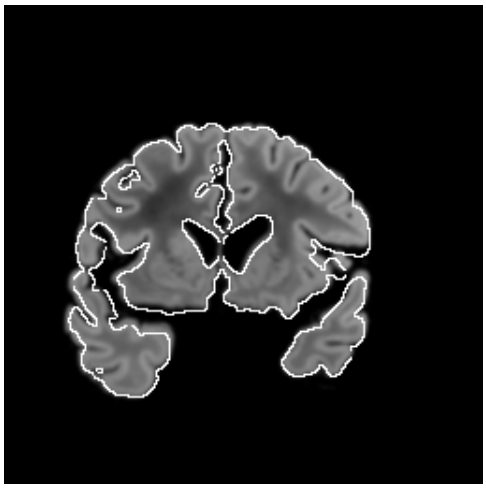


Abbildung 5.35: MRT Templatebild mit überlagerter Kontur des Referenzbildes in Abbildung 5.18.

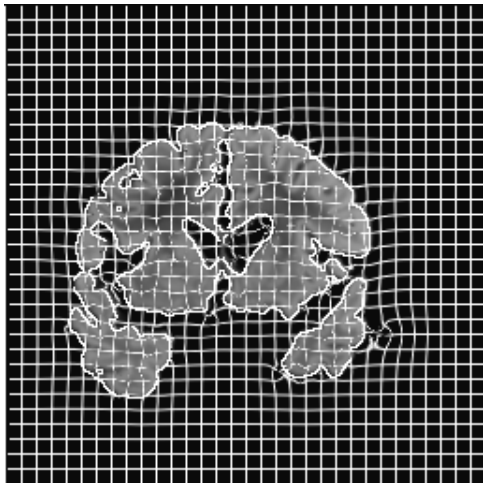


Abbildung 5.36: Deformiertes Templatebild mit überlagerter Kontur des Referenzbildes und überlagertem deformiertem Gitter.

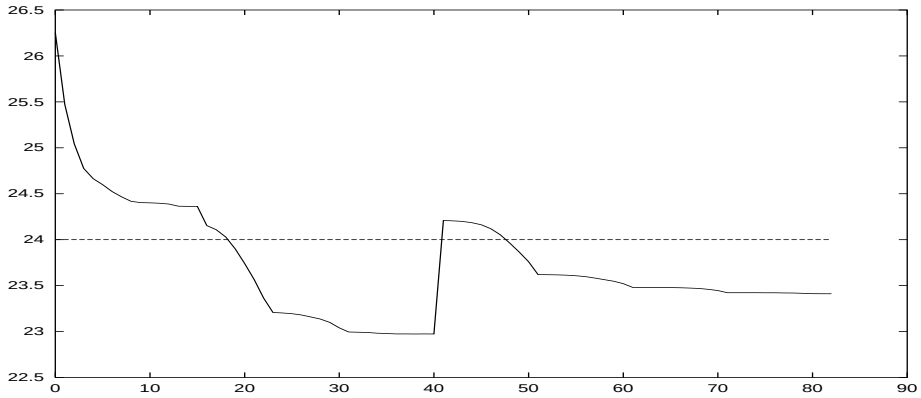


Abbildung 5.37: Verlauf des Defektfunktionals $D(u)$ für das Beispiel aus Abb. 5.34-5.36, berechnet mit dem Fortsetzungsverfahren.

Abschließend sollen Ergebnisse zu der in Kapitel 2.4 beschriebenen Aufgabe präsentiert werden. Hierbei sollen aus zwei von einem Satelliten aufgenommenen Bilddatensätzen auf die Windrichtung bzw. Windgeschwindigkeit geschlossen werden. In Abbildung 5.38 ist links das mit einem äquidistanten Gitter überlagerte Templatebild und rechts das ebenso mit einem äquidistanten Gitter überlagerte Referenzbild gegeben, wobei der Abstand zwischen zwei Gitterzellen in diesem Beispiel etwa 150 km beträgt. Hierbei handelt es sich bei dem Template um eine Aufnahme von Ostafrika um 10:30 Uhr. Das Referenzbild ist eine Aufnahme der selben Region um 11:00 Uhr. Mit bloßem Auge sind kaum Unterschiede zwischen den beiden Bildaufnahmen zu erkennen. Trotzdem sind die Wolken durch den Einfluss des Windes erheblich verschoben worden. Die Verschiebungen der Bilddatensätze sind in Abbildung 5.39 links durch das deformierte äquidistante Gitter angedeutet. In der rechten Abbildung ist das Ergebnis der Anpassung gegeben. Die Berechnung der Ergebnisse erfolgte durch das Fortsetzungsverfahren mit anfänglichem Regularisierungsparameter $\alpha = 100$ und einer Halbierung des Parameters im Laufe der Iteration ($\kappa = \frac{1}{2}$). Der Verlauf der Defektnorm durch die Verschiebungsvektoren ist in Abbildung 5.40 gegeben.

Wie in Abbildung 5.38 zu sehen ist, steckt die Information der Wolken in den Bildelementen mit hohen Grauwerten (weiße pixel), deshalb wurden die Wolken aus den Bildern durch eine sogenannte look-up table Operation heraussegmentiert. Hierbei wurden alle Grauwerte auf null gesetzt, die nicht mindestens den Grauwert 150 angenommen haben. Alle anderen Grauwerte blieben unverändert.

Bei den betrachteten Wolkenbildern hat man in der Regel eine höhere Bildauflösung, als bei den Bildern, die aus medizinischen Anwendungen resultieren. Zur Demonstration der Verfahren wurde die Bildgröße von 1024×1024 auf 256×256 Bildpunkte reduziert. Die Berechnung der Verschiebungen auf der Originalauflösung ist ein Beispiel dafür, dass die in Kapitel 6 vorgestellten parallelen Algorithmen auch bei zweidimensionalen Problemen zur Anwendung kommen.

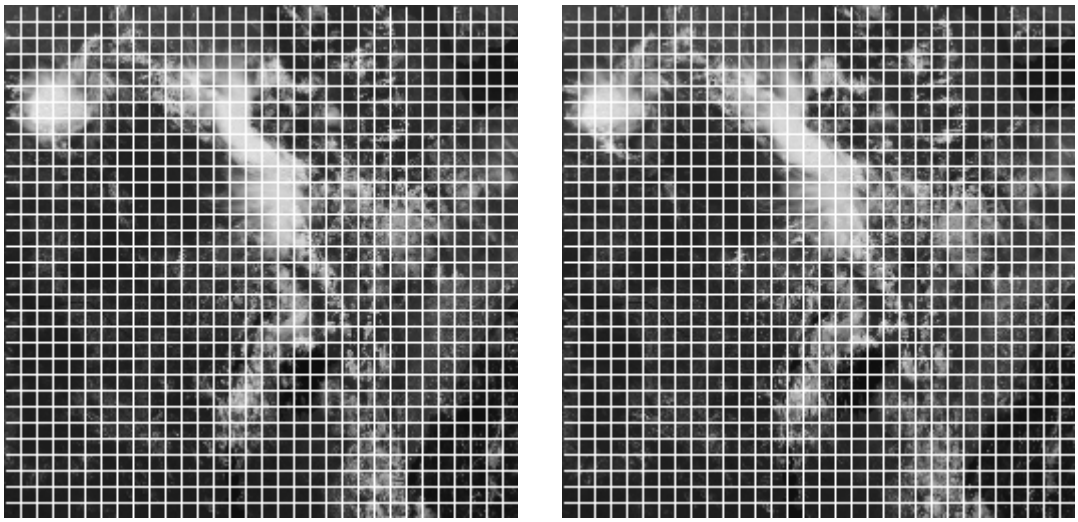


Abbildung 5.38: Template- und Referenzbild (10:30h) bzw. (11:00h).

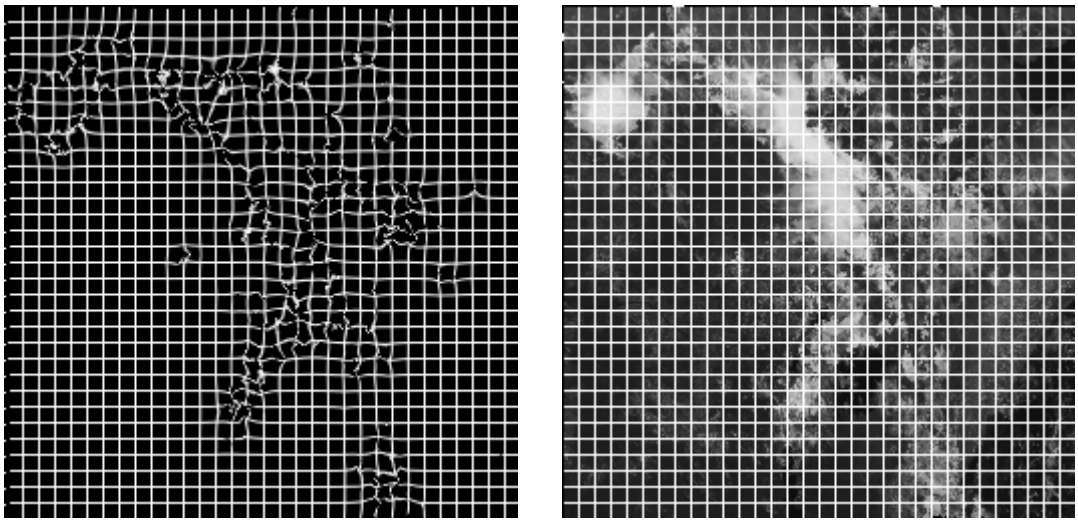


Abbildung 5.39: Links: Die berechneten Verschiebungen angewendet auf ein äquidistantes Gitter. Rechts: Der berechnete Bilddatensatz.

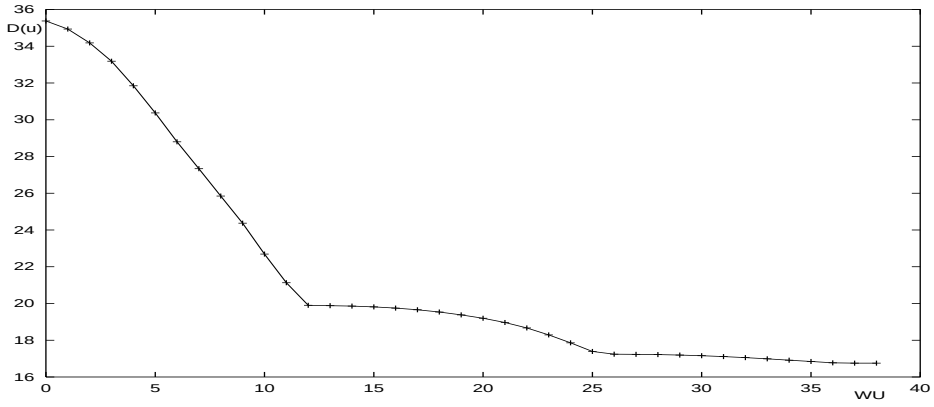


Abbildung 5.40: Verlauf des Defektfunktionals $D(u)$ während der Anpassung der Satellitenbilddaten aus Abbildung 5.38.

Kapitel 6

Parallelisierung der gitterbasierenden Minimierungsverfahren auf Parallelrechnern mit verteiltem Speicher

Der Informationsgehalt der Bilddaten ist in vielen Anwendungen enorm. Typischer Weise enthalten in der Medizin zweidimensionale Schnittbilder bis zu 512×512 Bildelemente und dreidimensionale Volumendatensätze $256 \times 256 \times 128$ Bildelemente. Aufnahmen der Erdoberfläche können bis zu 4096×4096 Bildelemente enthalten. Wegen dieser Datenmengen wird in diesem Kapitel das Parallelisierungspotential der Minimierungsverfahren diskutiert und ein geeigneter paralleler Algorithmus entwickelt. Die parallele Implementierung der Gitteralgorithmen auf Parallelrechnern mit verteiltem Speicher basiert auf der Methode des Grid Partitioning. Hierbei werden die digitalen Bilddaten in verschiedene Teilbilder zerlegt und auf entsprechend viele Prozessoren verteilt.

6.1 Die Methode des Grid Partitioning

Die in Kapitel 4 und 5 betrachteten Algorithmen zur numerischen Lösung des inversen Problems liefern eine Lösung u_h, die auf einem Diskretisierungsgitter G_h definiert ist. Die Diskretisierungsschrittweite h ist hierbei durch die gegebenen Daten vorgegeben.

Haben die Bilder einen hohen Informationsgehalt, wie in den meisten praktischen Anwendungen üblich, so ist h sehr klein und damit die Anzahl der Gitterpunkte sehr groß.
Das für solche Probleme wichtigste Hilfsmittel ist, neben der Verbesserung der numerischen Algorithmen, deren Parallelisierung, wodurch die numerische Berechnung beschleunigt werden soll.

Die Parallelisierung erfordert eine genaue Analyse und teilweise die Modifikation der Lösungsalgorithmen. Das betrifft nicht nur die parallele Abarbeitung der Lösungsschritte, sondern auch deren Speicherverwaltung. Die Zielrechnerklasse dieser Arbeit sind Multiprozessorsysteme mit verteiltem Speicher und Workstationcluster. Zur Parallelisierung der verwendeten Algorithmen wird der gesamte Speicher (globaler Speicher) auf mehrere lokale Speichereinheiten (lokaler Speicher) verteilt. Dabei hat ein Prozessor nur direkten Zugriff auf seinen lokalen Speicher. Die Information der den übrigen Rechnern zugewiesenen lokalen Speichereinheiten muss durch Kommunikation (Message-passing) zugänglich gemacht werden.

Die Kommunikation zwischen den Prozessoren mit zugehörigem lokalen Speicher ist ein kritischer Punkt in den folgenden parallelen Algorithmen und wird deshalb im nächsten Abschnitt genauer untersucht. Anschließend wird eine bekannte Variante zur verteilten Speicherung von Daten auf Mehrprozessorsystemen mit verteiltem Speicher, das Grid Partitioning, an die Lösungsalgorithmen angepasst.

6.1.1 Ein Kommunikationsmodell, Speed up und Effizienz für parallele Algorithmen

Bei der parallelen Bearbeitung von Algorithmen wird eine Beschleunigung dadurch herbeigeführt, dass die Arbeitslast auf mehrere Prozessoren verteilt wird. Das betrifft, neben der Abarbeitung des Algorithmus, auch die Speicherung der benötigten Daten. Bei Multiprozessorsystemen mit verteiltem Speicher hat jeder Prozessor nur auf seinen lokalen Speicher direkten Zugriff. Die Informationen auf anderen lokalen Speichereinheiten, die anderen Prozessoren zugeordneten sind, müssen durch explizite Kommunikation (Message passing) zugänglich gemacht werden. Die gesamte Rechenzeit, um einen parallelen Algorithmus durchzuführen, besteht aus der maximalen Rechenzeit T_{comp} der verschiedenen Prozessoren und aus der Zeit für die Kommuni-

kation T_{comm} zwischen den Prozessoren.

Ein in der Praxis häufig betrachtetes einfaches Kommunikationsmodell ist die Zeit, die zum Versenden einer Nachricht der Länge L benötigt wird. Sie wird durch

$$T_{comm} = \alpha + \beta L$$

mit den Parametern α und β modelliert. Der Parameter α beschreibt hierbei die sogenannte Startupzeit, die für den Aufbau der Verbindung benötigt wird. Der Parameter β ist die nominale Zeit, um jeden einzelnen der L Datensätze zu verschicken. Durch $1/\beta$ ist die Bandbreite der zugehörigen Verbindungsleitung zwischen den Prozessoren gegeben.

Zur Beurteilung paralleler Algorithmen auf Parallelrechnern mit verteiltem Speicher oder Workstationclustern wird die Zeit, die benötigt wird, um das Problem mit P Prozessoren zu lösen mit $T(P)$ bezeichnet. Der Beschleunigungsfaktor (speed up) $S(P)$ des parallelen Algorithmus und die parallele Effizienz $E(P)$ (die Beschleunigung pro Prozessor) werden durch

$$S(P) = \frac{T(1)}{T(P)} \quad \text{bzw.} \quad E(P) = \frac{S(P)}{P}$$

definiert. Hierbei wird angenommen, dass der parallele Algorithmus auch auf einem Prozessor ausgeführt werden kann. Interessant ist das Verhalten von $S(P)$ und $E(P)$ mit wachsender Anzahl der Prozessoren P. Idealerweise gilt

$$S(P) \approx P \quad \text{bzw.} \quad E(P) \approx 1,$$

was bedeutet, dass die Berechnung um einen Faktor P beschleunigt werden kann, wenn P statt eines Prozessors an der Arbeit beteiligt werden. Ein Grund, die Anzahl der Prozessoren P zu erhöhen, ist die wachsende Problemgröße N (wie etwa die Anzahl der Bildpunkte). Die Entwicklung der Funktionen S und E hängt bei praktischen Anwendungen wesentlich von der Problemgröße ab, und sie werden deshalb als Funktionen in Abhängigkeit von P und N betrachtet.

Bei Problemen mit gegebener Problemgröße sinkt bei wachsender Anzahl der Prozessoren die parallele Effizienz $E(P, N)$ des Algorithmus wieder ab. Der Grund hierfür ist, dass die Anzahl der arithmetischen Operationen, die ein Prozessor ausführt, bevor er mit den anderen wieder kommuniziert, immer geringer wird. Hierdurch kommt

es zu einem zu großen Anteil an Kommunikation (Communikation-overhead).
Ein weiterer Grund für unbefriedigende Effizienz eines parallelen Algorithmus kann eine ungleiche Lastenverteilung zwischen den verwendeten Prozessoren sein (load imbalance). Hierbei kommt es dazu, dass, während ein Teil der Prozessoren noch rechnet, andere auf ihre Daten warten.
Die Bewertung bzw. Entwicklung paralleler Algorithmen in der Praxis hängt von vielen verschiedenen Voraussetzungen in der Hardware und der Software ab, die im Einzelfall geprüft werden müssen. Die Realisierung der in dieser Arbeit betrachteten parallelen Lösungsalgorithmen basiert auf der Methode des Grid Partitioning, welche im folgenden Abschnitt vorgestellt wird.

6.1.2 Grid Partitioning

Das Prinzip des Grid Partitioning ist ein bekanntes Verfahren zur Lösung von Gitteralgorithmen auf Parallelrechnern mit verteiltem Speicher [36]. Hierbei wird die Information des Gitters G_h (globales Gitter) auf $P = p_x \times p_y$ Teilgitter verteilt, so dass alle Teilgitter ungefähr die gleiche Anzahl von Gitterpunkten enthalten. Jedes Teilgebiet wird einem Prozessor, der an der Berechnung der Lösung beteiligt werden soll, zugewiesen. Die Informationen der im Lösungsverfahren auftretenden Gitterfunktionen (f, u, T und R sowie diverse Hilfsfunktionen) werden für die jeweiligen Teilgitter im lokalen Speicher der Prozessoren gehalten. Hierdurch entsteht das Problem, dass die lokale Information auf den verschiedenen Prozessoren nicht allgemein verfügbar ist.

Ein Beispiel ist die Berechnung der Funktion f durch Gleichung (4.3) mit dem symmetrischen Differenzenoperator in x- und y-Richtung für ∇T. Hierbei ist es erforderlich, dass in den Randpunkten der Teilgitter die jeweiligen Werte von T in den Nachbarpunkten bereit gestellt werden (vgl. Abbildung 6.1). Statt immer, wenn die Nachbarpunkte benötigt werden, mit dem entsprechenden Prozessor zu kommunizieren, werden die Teilgitter jeweils um einen Überlappungsbereich der Breite $b = 1$ ergänzt. Die Überlappungsbereiche werden jeweils vor der Berechnung von f aufgefrischt.

Zusammengefasst ist die Methode des Grid Partitioning durch die folgenden Eigenschaften charakterisiert:

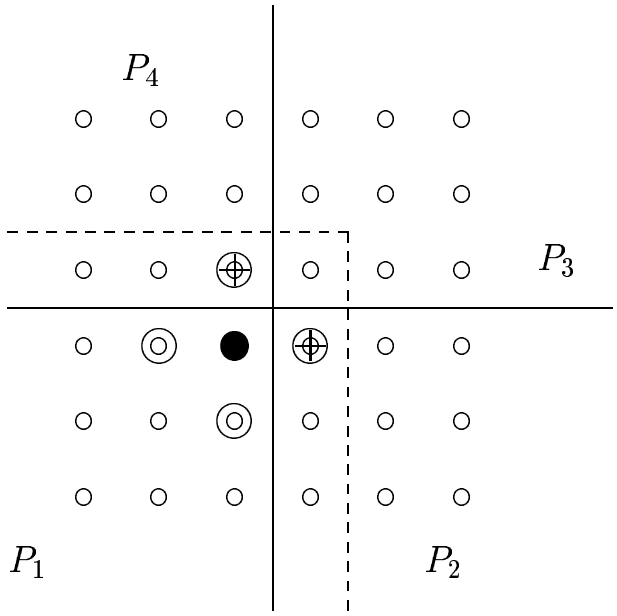


Abbildung 6.1: Zur Berechnung der Funktion f auf einem Randpunkt (dunkel markiert) auf Prozessor P_1 werden auch Bildpunkte benötigt, die auf den Prozessoren P_2 bzw. P_4 liegen (mit Kreuzen markiert).

- Das globale Gitter G_h wird in Teilgitter zerlegt.
- Die Werte der Funktionen u, f, T und R sowie diverser Hilfsfunktionen werden gemäß der Zerlegung von G_h den Prozessoren zugewiesen. Hierdurch wird gesichert, dass die lokale Berechnung der Lösung im Innern der Teilgitter durchgeführt werden kann.
- Zur Durchführung der Lösungsschritte auf den Gitterpunkten am Rand der Teilgitter werden die auf den Teilgittern definierten Funktionen um die nötigen Überlappungsbereiche erweitert.
- Die Kommunikation zwischen den Teilgittern erfolgt im Wesentlichen durch das Auffrischen der nötigen Überlappungsbereiche.

Abhängig von dem Multiprozessorsystem wird die Partitionierungsstrategie des globalen Gitters unterschiedlich gewählt. Bei t_x oder t_y gleich eins spricht man von einer Streifenzerlegung, bei $t_x \approx t_y$ von einer Rechteckzerlegung (vgl. Abbildung 6.2).

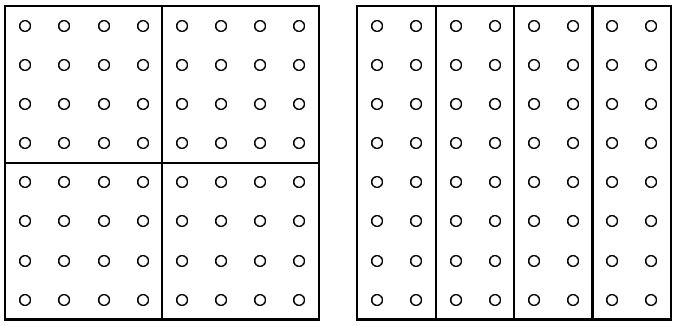


Abbildung 6.2: Rechteck- (links) und Streifenzerlegung (rechts) des globalen Gitters.

Da bei der Streifenzerlegung jeder Prozessor nur mit seinem linken bzw. rechten Nachbarn kommunizieren muss, ist die Anzahl der Nachrichten bei der Streifenzerlegung geringer als bei der Rechteckzerlegung. Dafür ist eine Nachricht bei der Streifenzerlegung etwa $\sqrt{P}$-mal so groß wie bei der Rechteckzerlegung. Bei Architekturen, bei denen die Startupzeit (α) dominant groß ist, ist eine Streifenzerlegung von Vorteil. Andererseits ist bei Architekturen mit hoher Transferzeit eine Rechteckzerlegung des globalen Gitters zu empfehlen.

Grid Partitioning bei Mehrgitteralgorithmen

Bei der Durchführung der Methode des Grid Partitioning bei Algorithmen, die mehrere Auflösungsstufen zur Berechnung einer Lösung einbeziehen, werden die Gitterpunkte der verschiedenen Gitterebenen so auf die Prozessoren verteilt, dass sie jeweils auf einem Prozessor liegen (vgl. Abb. 6.3). Hierdurch wird gewährleistet, dass die Kommunikation zwischen zwei Gitterebenen nur durch das Auffrischen der Überlappungsbereiche stattfindet. Bei der rekursiven Durchführung des Grid Partitioning auf mehreren Gitterebenen kommt es zu den folgenden Problemen:

- Das Verhältniss T_{comm}/T_{comp} ist auf den groben Gittern wesentlich schlechter als auf den feinen Gittern.
- Auf groben Gitterebenen mit wenig Gitterpunkten kommt es dazu, dass mehrere Prozessoren dieselben Gitterpunkte bearbeiten. In Abb. 6.3 sind zum Beispiel

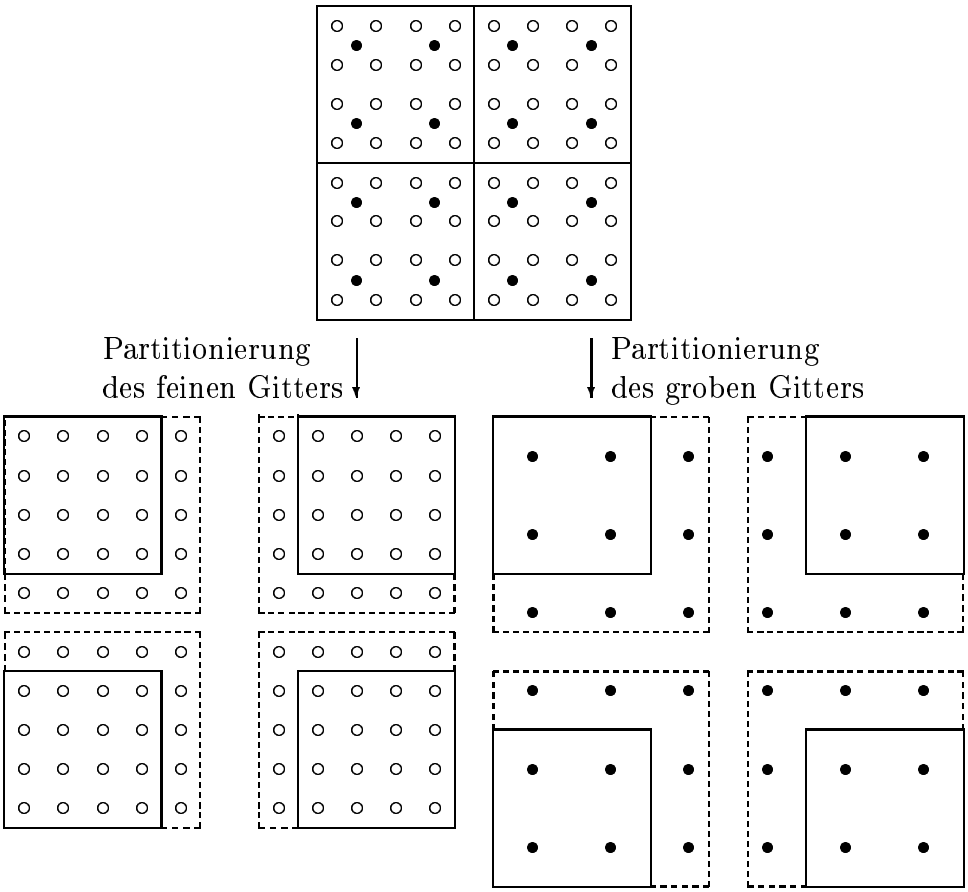


Abbildung 6.3: Partitionierung der Gitterpunkte des groben und feinen Gitters auf vier verschiedene Prozessoren mit Überlappungsbereichen der Breite $b = 1$.

auf dem groben Gitter auf allen vier Prozessoren alle inneren Gitterpunkte präsent.

- Auf groben Gitterebenen kommt es zu einem Communikations-overhead. Das bedeutet, dass die Kommunikationszeit zwischen den verschiedenen Prozessoren im Vergleich zur Rechenzeit der Prozessoren zu lange ist.

Der extreme Communikations-overhead auf den groben Gittern ist weitaus zeitintensiver, als die Arbeit auf weniger Prozessoren zu verteilen und die restlichen Prozessoren in Wartestellung zu lassen. Dies führt auf die Agglomerations-Technik, bei der, bei zu klein werdenden Teilgitterproblemen pro Prozessor, das Problem auf weniger Prozessoren zusammengezogen wird. Wie zum Beispiel in Abbildung 6.4 wird hier beim Übergang auf das grobe Gitter das Problem von vier auf zwei Prozessoren verlagert.

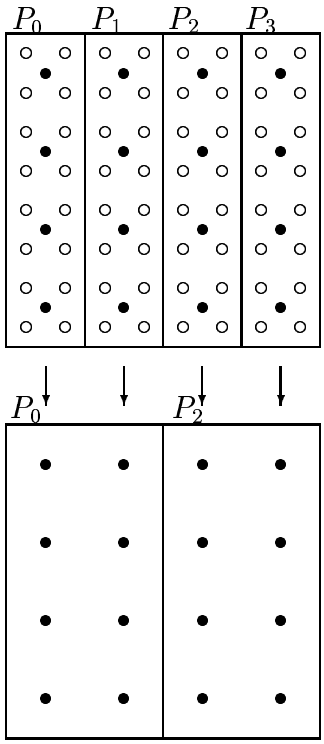


Abbildung 6.4: Agglomeration des groben Gitters auf zwei Prozessoren.

Paralleles Programmiermodell mit Message Passing

Die Implementierung der Lösungsalgorithmen auf einem Parallelrechner mit verteiltem Speicher erfordert eine geeignete Programmierumgebung. Als in den achziger Jahren die ersten Parallelrechner mit verteiltem Speicher entwickelt wurden, hatte jeder Rechner sein eigenes Programmiermodell. In der Folgezeit setzte sich die Parallel Virtual Maschine (PVM) [15] als portable Programmieroberfläche zum Austausch von Daten auf den meisten Systemen durch. Anfang der neunziger Jahre wurde in Zusammenarbeit von Hardwareherstellern und Entwicklern MPI (Message Passing Interface) [27] zum neuen Standard erhoben und bis heute zur Version 2.0 ständig weiter entwickelt.

Dem Entwickler stehen zur Zeit ca. 120 MPI-Unterroutinen in Fortran-77 und C zur Verfügung. Hiermit wird die Kommunikation zwischen den verschiedenen Prozessen realisiert. Die Funktionen `MPI_Init()` bzw. `MPI_Finalize()` müssen von allen Prozessoren vor bzw. nach allen anderen MPI-Routinen aufgerufen werden. MPI setzt voraus, dass alle Programmteile unabhängig von anderen Prozessen (mit Ausnahme von MPI-Aufrufen) ausgeführt werden können.

Die Prozesse einer Anwendung bilden am Anfang eine Gruppe. Die Kommunikation zwischen den Prozessen findet innerhalb eines entsprechenden Kontextes mit einem zugehörigen Kommunikator statt. Nach `MPI_Init()` steht immer `MPI_COMM_WORLD` als Kommunikator für die Gruppe aller Prozesse zur Verfügung. Innerhalb dieser Gruppe sind die Prozesse von null an durchnummeriert. Durch den Aufruf von

```
int myrank, numprocs;
MPI_Comm_rank(MPI_COMM_WORLD, &myrank);
MPI_Comm_size(MPI_COMM_WORLD, &numprocs);
```

bekommt jeder Prozess seine Nummer (`myrank`) und die Anzahl aller Prozesse (`numprocs`) mitgeteilt. Mit Hilfe der Nummern können die Prozesse zu unterschiedlichen Teilgruppen zusammengefasst werden. Dies geschieht beispielsweise mit der Funktion

```
int color;
MPI_Comm_split(MPI_COMM_WORLD, color , myrank , MPI_COARSE_WORLD);
```

durch die der Kommunikator `MPI_COMM_WORLD` in disjunkte Untergruppen partitioniert wird. Es werden so viele Untergruppen erzeugt wie Farben existieren. Jede Untergruppe enthält alle Prozesse mit der gleichen Farbe `color`, die gemäß ihrer alten Nummerierung erneut innerhalb der Untergruppe durchnummeriert werden. In Abbildung 6.5 wird die Gruppe mit den Prozessoren P_0-P_7 in zwei Untergruppen zerlegt. Die Zerlegung der Prozessoren in mehrere Teilgruppen ist ein wichtiges Hilfsmittel für die Agglomeration der Teilgitter bei Mehrgitteralgorithmen und wird im nächsten Abschnitt genauer beschrieben.

Innerhalb der Kommunikatoren kann Kommunikation zwischen den zusammengehörenden Prozessoren stattfinden. Die Kommunikation zwischen zwei Prozessen (Punkt-zu-Punkt-Kommunikation) findet auf der Basis von Nachrichten statt. Der Inhalt einer Nachricht ist ein fortlaufender Speicherbereich mit Elementen gleichen Datentyps. Der Inhalt einer Nachricht ist spezifiziert durch

- einen Zeiger auf den Anfang des Datenbereichs,

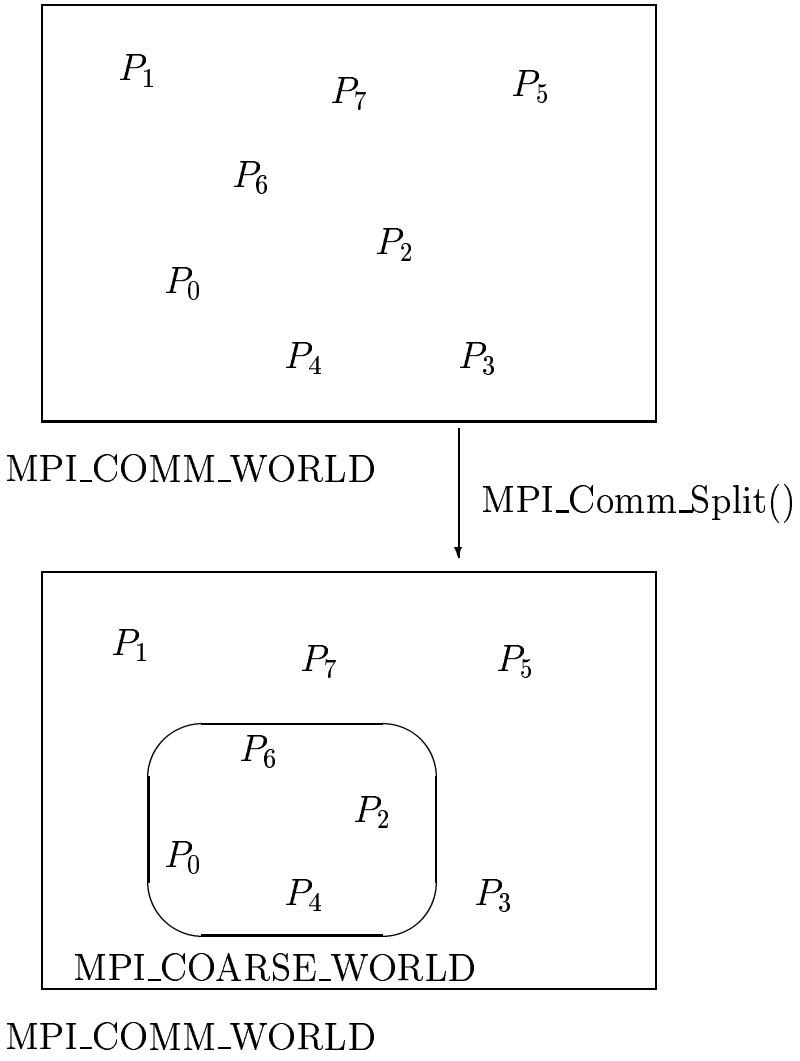


Abbildung 6.5: Zerlegung der Gruppe MPI_COMM_WORLD in zwei Kommunikatoren mit jeweils vier Prozessoren.

- die Anzahl der Elemente und
- den Datentyp.

Neben dem Inhalt enthält eine Nachricht

- den Absender der Nachricht,
- den Empfänger der Nachricht,
- eine Kennung (Tag),
- und den entsprechenden Kommunikator.

Der Empfänger und der Sender einer Nachricht muß hierbei jeweils den Kommunikationspartner, die Kennung und den Kommunikator spezifizieren.
Neben der Punkt-zu-Punkt-Kommunikation bietet MPI die Möglichkeit, mehrere Prozesse an der Kommunikation zu beteiligen. Ein Beispiel ist das Zusammenfassen aller Teilgitter aller Prozesse auf einem ausgewählten Prozessor, auf dem das gesamte Gitter wieder zusammengefügt wird. Hierzu eignet sich die Funktion `MPI_Gather`, mit der eine Nachricht von allen Prozessen einer Gruppe an einen bestimmten Prozess (root) gesendet wird. Das Gegenteil geschieht bei der Funktion `MPI_Scatter`, hiermit werden verschiedene Nachrichtenblöcke von einem Prozessor an alle anderen Prozessoren versendet. Mit der Funktion `MPI_Bcast` wird eine Nachricht mit gleichem Inhalt von dem root-Prozess an alle anderen Prozessoren der Gruppe versendet. Um globale Operationen auf den Teilgittern auszuführen, wie etwa die Berechnung der Norm des Verschiebungsvektors, stellt MPI die Funktion `MPI_Reduce()` bereit. Hiermit werden lokal berechnete Ergebnisse (z.B. die maximale Verschiebung auf den Gitterpunkten eines Teilgitters) auf dem root-Prozessor global weiter verarbeitet.

6.2 Grid Partitioning für Mehrgitteralgorithmen zur Minimierung des Defektfunktionals

Nachdem im vorherigen Abschnitt die Grundlagen und Problematiken für die Parallelisierung von Mehrgitteralgorithmen bereitgestellt wurden, wird im Folgenden im Detail auf die einzelnen Komponenten der Lösungsalgorithmen eingegangen. Hierbei werden zunächst die in Kapitel 4 vorgestellten Mehrgitterverfahren zur Lösung der partiellen Differentialgleichungen analysiert.

6.2.1 Parallelisierung von Mehrgitterverfahren zur Lösung partieller Differentialgleichungen

Die numerische Berechnung der Verschiebungsvektoren durch ein Mehrgitterverfahren (MG-CS oder MG-FAS) mit der Methode des Grid Partitioning ist der zeitaufwendigste Teil der Lösungsalgorithmen aus Kapitel 5 für das Anpassungsproblem. In diesem Abschnitt soll deshalb die Parallelisierung der verwendeten Mehrgitterverfahren genauer untersucht werden. Ein Mehrgitterverfahren ist durch seine Gitterhier-

archie (z.B. V- oder W- Zyklus) und durch die gewählten Komponenten (Glättungsverfahren, Restriktion, Interpolation) definiert.

Analyse der Mehrgitterkomponenten

Bei der Analyse der Mehrgitterkomponenten spielt die Anzahl der Gitterpunkte, die gleichzeitig bearbeitet werden können (Parallelitätsgrad der Mehrgitterkomponente), eine wichtige Rolle.
Die Berechnung der groben Gitterpunkte durch die in Kapitel 4 vorgestellten Restriktionsoperatoren I_h^H kann für alle groben Gitterpunkte parallel durchgeführt werden. Gleiches gilt für die Berechnung der feinen Gitterpunkte durch die Interpolationsoperatoren I_H^h aus Kapitel 4, die ebenfalls unabhängig von allen anderen Feingitterpunkten durchgeführt werden kann.

Einen unterschiedlichen Grad der Parallelisierung erhält man für die Relaxationsverfahren. Die in Kapitel 4.3.1 vorgestellten Relaxationsverfahren für das MG-CS sind zur Berechnung eines Wertes einer Gitterfunktion in einem Punkt, eine Funktion der umliegenden Nachbarpunkte. Damit sind die linearen Relaxationsverfahren ein lokaler Algorithmus. Einen vollständigen Grad der Parallelisierung $|\Omega_h| = N$ erhält man für das Gesamtschrittverfahren, weil hierbei zur Berechnung einer neuen Näherung die umliegenden alten Näherungslösungen verwendet werden.
Beim Block-Gauß-Seidel Verfahren (vgl. Kapitel 4.3.1) werden zunächst die Punkte der ersten Komponente von $u^{(n)} = (u_1^{(n)}, u_2^{(n)})^T$ mit dem Gesamtschrittverfahren iteriert, anschließend die Punkte der zweiten Komponente unter Verwendung der gerade berechneten Werte $u_1^{(n+1)}$ und den alten Werten der zweiten Komponente $u_2^{(n)}$. Hierdurch treten keinerlei Abhängigkeiten zwischen den Punkten auf, und die Iteration hat ebenso einen maximalen Parallelisierungsgrad N.

Der Parallelisierungsgrad der Gauß-Seidel-artigen Verfahren hängt von der Nummerierung der Gitterpunkte ab, in der das Gitter durchlaufen wird.

Die nichtlinearen Relaxationsverfahren aus Kapitel 4.3.4 haben den gleichen Parallelisierungsgrad wie die entsprechenden linearen Relaxationsverfahren. Sie sind jedoch keine lokalen Verfahren, weil die Berechnung einer Näherung in einem Gitterpunkt keine Funktion seiner Nachbarpunkte ist. Das liegt an der globalen Auswertung des

nichtlinearen Funktionals

$$f_h(u_h(x_h)) = \nabla T_h(x_h - u_h(x_h)) \cdot (T_h(x_h - u_h(x_h)) - R_h(x_h))$$

im Gitterpunkt x_h. Bei großen Verschiebungen u kann es sein, dass die Werte von $T(x - u(x))$ gerade einem anderen Prozessor zugeordnet werden (vgl. Abbildung 6.6). Deshalb wird die Funktion T_h nicht auf die verschiedenen Prozessoren verteilt, sondern es wird auf jedem Prozessor die gesamte Funktion T_h gespeichert. Die

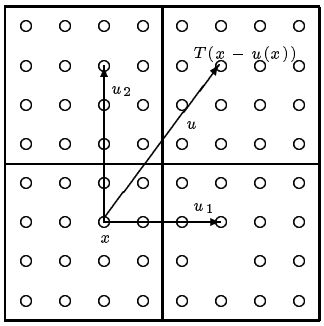


Abbildung 6.6: Die Berechnung von $T(x-u(x))$ ist problematisch, wenn die Gitterfunktion T mit der Methode des Grid Partitioning auf verschiedene Prozessoren verteilt wird.

Berechnung der Defektfunktion d_h kann für das MG-CS und das MG-FAS parallel durchgeführt werden.

Analyse der Mehrgitterstruktur

Die Parallelisierung von Mehrgitterverfahren beschränkt sich auf die Parallelisierung der Mehrgitterkomponenten. Die Gitterebenen werden wie im sequentiellen Algorithmus hierarchisch nacheinander behandelt. Der Parallelitätsgrad auf den gröberen Gittern sinkt hierdurch mit der Anzahl der Gitterpunkte. Auf den groben Gittern ist der Grad der Parallelisierung also wesentlich geringer als auf den feinen Gittern. Das führt dazu, dass der parallele V-Zyklus einen ganz anderen Grad der Parallelisierung hat als z.B. der W-Zyklus. In [36] wird diese Problematik anhand eines Modellproblems analysiert. In der praktischen Anwendung ist wegen der notwendigen Agglomeration und der großen Diskrepanz zwischen Kommunikations- und Rechenaufwand auf den groben Gittern ein V-Zyklus empfehlenswert, zumal bei der Berechnung der Verschiebungsvektoren keine großen Genauigkeitsanforderungen gestellt sind.

6.2.2 Parallelisierung der Minimierungsverfahren für das Defektfunktional

Algorithmische Aspekte bei der Parallelisierung

Die Minimierung des Defektfunktionals verläuft, wie in Kapitel 5.5 beschrieben, auf mehreren Gitterebenen, die durch die Bildauflösung bestimmt sind. Der parallele Algorithmus lässt sich in drei Phasen aufteilen:

- **Partitionierungsphase**

  In der ersten Phase des parallelen Algorithmus wird das durch die Bildauflösungsstufe bestimmte globale Gitter G_h auf die gewünschte Anzahl von Prozessoren verteilt. In dieser Phase werden die Gitter der verschiedenen Auflösungsstufen generiert und die benötigten Überlappungsbereiche hinzugefügt. Die Zuordnung der Gitter zu den Prozessoren findet innerhalb des Algorithmus wohldefiniert statt.

- **Gittertransferphase**

  Während der Gittertransferphase wird die lokale Information zwischen den einzelnen Gitterebenen weitergegeben. Hierbei werden verschiedene Schritte unterschieden:

  - Der Transfer der digitalisierten Bilder T_h bzw. R_h auf die unterschiedlichen Bildauflösungsstufen.
  - Die Überführung der auf den groben Bildauflösungen berechneten Näherungslösungen auf die nächst feinere Bildauflösungsstufe.
  - Die Restriktion des Defekts (bzw. der Näherungslösung im MG-FAS) von einer feinen auf die nächst gröbere Gitterebene und die Interpolation der Korrektur von der groben auf die nächst feinere Gitterebene.

  Die hierbei benutzte Gitterhierarchie wird in der Partitionierungsphase eindeutig festgelegt.

- **Minimierungsphase auf einer Bildauflösungsstufe**

  Zu Beginn der Minimierungsphase durch ein linearisiertes Abstiegsverfahren

oder durch das Fortsetzungsverfahren auf einer Gitterebene ist die Partitionierung festgelegt. Auch die Gitterhierarchie für die verwendeten MG-Verfahren auf den verschiedenen Bildauflösungsstufen ist festgelegt. Die Berechnung der Verschiebungsvektoren, deren Normen und die Transformation der Bilder, wird auf jedem Prozessor lokal durchgeführt.

Gitterpartitionierung

Die digitalen Bilddaten werden möglichst gleichmäßig auf alle Prozessoren verteilt. Dies geschieht im Falle einer Streifenzerlegung entlang der letzten Raumdimension, d.h. die digitalen Bilder werden im Zweidimensionalen in Streifen (Bildspalten) oder im Dreidimensionalen in zweidimensionale Scheiben zerlegt.
Dies geschieht wie folgt: Sei n die Anzahl der Bildelemente der letzten Koordinatenrichtung. Dann wird jedem der P zur Verfügung stehenden Prozessoren n/P Streifen bzw. Scheiben zugewiesen. Zudem erhalten die ersten $mod(n, P)$ Prozessoren einen weiteren Streifen bzw. eine weitere Scheibe. Die Zuordnung geschieht von links nach rechts, so dass der erste Prozessor die Streifen $0, ..., n/P - 1$ falls $mod(n, P) = 0$ und der letzte die Streifen $(P-1)n/P, .., n$ erhält. Die Anzahl der Bildpunkte in den verschiedenen Koordinatenrichtungen ist in der digitalen Bildverarbeitung meistens eine Zweierpotenz, um einen optimalen Lastenausgleich zu bekommen, wird die Anzahl der Prozessoren $P = 2^k$ mit $k \in I\!N$ gewählt.

Genau wie die Bilder, werden die Gitter für die Verschiebungsvektoren und die rechte Seite zerlegt. Für die Gitter der Verschiebungsvektoren wird ein Überlappungsbereich der Breite eins hinzugefügt.

Auf den gröberen Gitterebenen werden die Bildpunkte in gleicher Weise auf die Prozessoren verteilt. Allerdings sollen hier nicht mehr alle Prozessoren an der Arbeit beteiligt werden, deshalb werden die Bildpunkte auf P^1 Prozessoren mit $P^1 = 2^l < 2^k$ verteilt. Prozessoren, die keine Gitterpunkte auf den groben Gittern bearbeiten, werden solange in einen Ruhezustand versetzt, bis die Berechnung auf der jeweiligen Gitterebene weiter geht. Anschaulich werden die Prozessoren abwechselnd mit verschiedenen Farben markiert. Nur die Prozessoren der ersten Farbe haben Bildpunkte auf der nächst gröberen Auflösungsstufe. Bei einer Zwei-Farben-Agglomeration der Prozessoren (vgl. Abb. 6.7) bekommt jeder Prozessor der ersten Farbe die Grobgit-

terpunkte seines mit der zweiten Farbe markierten rechten Nachbarn.

- Initialisierung (Level 1): Alle acht Prozessoren werden mit der Farbe 0 markiert.

	P_0	P_1	P_2	P_3	P_4	P_5	P_6	P_7
myid	0	1	2	3	4	5	6	7
color	0	0	0	0	0	0	0	0

- Agglomerationsphase (Level 1): Die Prozessoren werden in zwei Gruppen aufgeteilt. Die erste Gruppe wird mit der Farbe 0 und die zweite mit der Farbe 1 markiert.

	P_0	P_1	P_2	P_3	P_4	P_5	P_6	P_7
myid	0	0	1	1	2	2	3	3
color	0	1	0	1	0	1	0	1

  Auf der groben Bildauflösungsstufe sind nur noch Prozessoren an der Arbeit beteiligt, die mit der Farbe 0 markiert wurden. Die anderen Prozessoren verweilen in Wartestellung, bis der Algorithmus wieder Level 1 erreicht hat.

- Agglomerationsphase (Level 2): Die aktiven Prozessoren werden erneut in zwei Gruppen aufgeteilt. Die erste Gruppe bleibt mit der Farbe 0 markiert. Die zweite Gruppe wird mit der Farbe 2 markiert.

	P_0	P_2	P_4	P_6
myid	0	0	1	1
color	0	2	0	2

- Agglomerationsphase (Level 3): Die aktiven Prozessoren werden erneut in zwei Gruppen aufgeteilt.

	P_0	P_4
myid	0	1
color	0	3

- Der Algorithmus läuft auf Level 4,5,.. nur noch auf einem Prozessor.

	P_0
myid	0
color	0

Bei der Implementierung durch MPI geschieht dies durch die im vorherigen Abschnitt beschriebene Funktion `MPI_Comm_split`. In der C++-Methode `processorSplit()` wird die Gruppe der Prozessoren `procWorldFine` in `numberOfColors` viele Teilgruppen zerlegt. Die Zerlegung der Prozessoren in Teilgruppen wird in `processorWorldCoarse` gespeichert.

```
MPI_Comm processorSplit(int numberOfColors) {

int myrank_fine ;

MPI_Comm_rank(procWorldFine, &myrank_fine);

color = myrank_fine % numberOfColors;

MPI_Comm  procWorldCoarse;
MPI_Comm_split( procWorldFine , color , myrank_fine , &procWorldCoarse);

return processorWorldCoarse;

}
```

Bei großen Problemen ist es sinnvoll, nicht direkt beim Übergang vom ersten auf das zweite Gitter die Anzahl der Prozessoren zu reduzieren, sondern erst ab einer gewissen Anzahl der Gitterpunkte.

Paralleler Gittertransfer

Beim Transfer der Daten zwischen den einzelnen Gitterebenen ist es durch die Agglomeration der Gitter auf weniger Prozessoren notwendig, die Daten der Prozessoren, die auf den groben Gitterebenen nicht an der Arbeit beteiligt sind, auf die aktiven Prozessoren zu verteilen. Anders herum müssen beim Transfer von den groben Gittern auf die feinen Gitter die Daten auf die in Wartestellung liegenden Prozessoren verteilt werden. Hierzu legen die auf den groben Gittern weiterhin aktiven Prozessoren auf der feinen Gitterebene ein Hilfsgitter an. Auf dem Hilfsgitter speichern die Prozessoren ihre eigenen Feingitterdaten. Die Prozessoren, die auf den groben Gittern nicht aktiv sind, senden ihre Daten an die weiterhin aktiven Prozessoren, so dass diese Prozessoren die gesamte Feingitterinformation in den Hilfsgittern speichern (vgl. Abb. 6.7). Die Restriktion der Daten findet dann auf den aktiven Prozessoren statt. Umgekehrt werden die Daten zunächst vom groben Gitter in das Hilfsgitter

interpoliert und dann auf die in Wartestellung verweilenden Prozessoren versendet.

Minimierungsverfahren auf den verschiedenen Bildauflösungsstufen

Für die Algorithmen zur Minimierung des Defektfunktionals auf einzelnen Bildauflösungsstufen ist die Partitionierung der Gitterfunktionen durch die entsprechende Zerlegung des digitalen Bildes vorgegeben. Die Berechnung der Verschiebungsvektoren durch ein MG-Verfahren kann die Zerlegung der Gitter direkt übernehmen. Die Partitionierung der groben Gitter ist aus der Partitionierungsphase bekannt. Alle weiteren Schritte der Minimierungsalgorithmen, wie die Berechnung der Maximum- bzw. L_2-Norm und der Addition der Verschiebungsvektoren, kann lokal auf den einzelnen Prozessoren erfolgen. Hierbei besteht keinerlei Abhängigkeit zwischen den Daten der Gitterpunkte.
Um beispielsweise die Maximumnorm für das gesamte Gitter zu berechnen, wird sie lokal auf den verschiedenen Teilgittern berechnet, werden auf einem Prozessor die Normen für die Teilgitter gesammelt, dort ausgewertet und zum Schluss auf die anderen Prozessoren verteilt. Hierzu werden die MPI-Funktionen `MPI_Reduce()` und `MPI_Bcast()` aus Kapitel 6.1.2 verwendet. Im folgenden Beispiel wird mit der Funktion `lokal_maxVV()` die maximale Verschiebung von $d = (u, v)$ berechnet.

```
//lokal
float maX = lokal_maxVV( u , v);
float erg = 0.00 ;
// global
MPI_Reduce(&maX, &erg, 1 , MPI_FLOAT, MPI_MAX , 0 ,  CommWorld);
MPI_Bcast(&erg, 1, MPI_FLOAT , 0,  CommWorld);
```

Anschließend wird mit `MPI_Reduce()` das Maximum aller Teilgitter berechnet und auf Prozessor 0 in der Variable `erg` gespeichert. Der anschließende Aufruf von `MPI_Bcast()` verschickt das Ergebnis an alle Prozessoren die zur Gruppe `CommWorld` gehören.

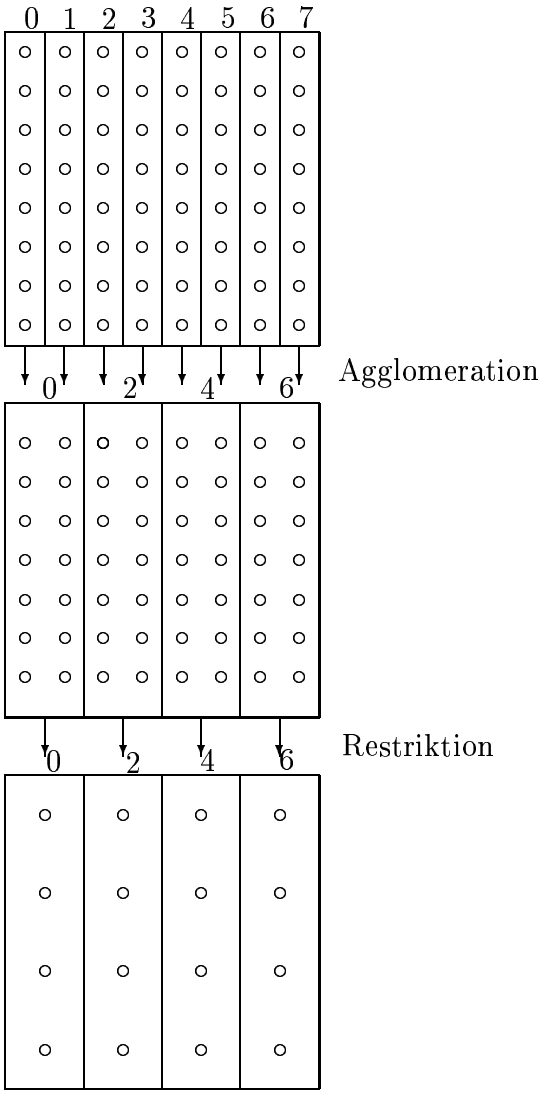


Abbildung 6.7: Illustration des Datentransfers zwischen zwei Gitterebenen anhand einer Zwei-Farben-Agglomeration der Prozessoren. Im ersten Schritt werden die Daten auf die mit der ersten Farbe markierten Prozessoren zusammengezogen. Im zweiten Schritt werden die Daten lokal auf diesen Prozessoren restringiert.

6.2.3 Ergebnisse

In diesem Abschnitt werden die benötigten Rechenzeiten für die Berechnung der Verschiebungsvektoren für Bilddaten mit unterschiedlicher Auflösungsstufe und für verschiedene Agglomerationsstrategien untersucht. Die Tabellen 6.1, 6.2 und 6.3 enthalten die ermittelten Gesamtrechenzeiten in Sekunden für das mit unterschiedlich vielen Prozessoren ausgeführte parallele Programm. Zudem ist die Anzahl der aktiven Prozessoren und die entsprechende parallele Effizienz bzw. parallele Beschleunigung für die Berechnungen aufgeführt.

Die Berechnungen erfolgten auf dem Parallelrechner Origin2000 der Firma Silicon Graphics im Rechenzentrum der Heinrich-Heine-Universität Düsseldorf. In diesem Zusammenhang sei erwähnt, dass sich die theoretischen Komplexitätsbetrachtungen aus Abschnitt 6.1.1 aufgrund moderner Rechnerarchitekturen nicht direkt auf die benötigten Rechenzeiten einer Implementation überführen lassen. So sind die Zeiten für die Berechnung der Verschiebungsvektoren mit einem Prozessor in den Tabellen 6.2 und 6.3 durch die aus Tabelle 6.1 geschätzt. Die tatsächlichen Ausführzeiten sind deutlich länger, was beispielsweise zu einer parallelen Effizienz von über 100% führen kann, wenn die Anzahl der Prozessoren steigt, und damit nicht mit den theoretischen Komplexitätsaussagen zu vereinbaren ist.
Eine Ursache hierfür ist, dass moderne Rechnerarchitekturen zwischen dem Hauptspeicher und dem Prozessor sogennante Caches haben, auf die der Prozessor wesentlich schneller zugreifen kann als auf den Hauptspeicher.
Der Zugriff auf eine Adresse im Cache dauert etwa drei Takte, der auf eine Adresse im Hauptspeicher etwa 10 Takte. Der Cache ist bei dem verwendeten Rechner 4 MB groß. Der Hauptspeicher umfasst insgesamt 13 GB mit jeweils 7 Knoten mit je 2 Prozessoren und jeweils 1 GB bzw. 12 Knoten mit je 2 Prozessoren und jeweils 512 MB. Will ein Prozessor auf eine Adresse im Hauptspeicher zugreifen, so wird zunächst geprüft, ob der Inhalt im Cache gespeichert ist. Ist das der Fall (sogenannter Cache-Hit), so findet ein schneller Cache Zugriff auf den Zwischenspeicher statt. Andernfalls (sogenannter Cache-Miss) wird auf den langsameren Hauptspeicher zugegriffen. Der Zugriff auf den Cache findet hierbei durch spezielle Hash-Algorithmen statt. Bei der Entwicklung paralleler Algorithmen hat man im Allgemeinen keinen Einfluß darauf,

welche Daten im Cache stehen, sondern man verläst sich auf die von den Chipdesignern entworfenen Strategien zur Cachesteuerung.
Bei wachsender Problemgröße kommt es also dazu, dass die Kapazität des Caches ausgeschöpft ist und die Daten aus dem Hauptspeicher geladen werden. Hierdurch entsteht eine maschinenabhängige Verzögerung des Programmablaufs.

In den Abbildungen 6.1 bis 6.3 sind die Ergebnisse für die Berechnung der dreidimensionalen Verschiebungsvektoren mit verschiedenen Bildauflösungsstufen dargestellt. Alle Ergebnisse wurden auf dem oben angesprochenen Parallelrechner Origin2000 ermittelt.
Für die Berechnung wurden unterschiedliche Agglomerationsstrategien untersucht, bei denen die Anzahl der an der Arbeit beteiligten Prozessoren auf den gröberen Gitterebenen variiert wurde. Hierbei ist bei allen betrachteten Problemgrößen zu beobachten, dass der speed-up bei der Berechnung mit zwei Prozessoren stetig anwächst, sobald die Anzahl der aktiven Prozessoren auf den groben Gittern erhöht wird. Die Berechnung mit zwei Prozessoren kommt hierbei für alle Problemgrößen ihrem Idealwert von 200% mit einem Beschleunigungsfaktor von bis zu 183% am nächsten, wenn auf den vier feinsten Gittern alle Prozessoren an der Arbeit beteiligt werden. Andererseits, wenn die beiden Prozessoren nur auf der feinsten Auflösungsstufe gemeinsam an der Arbeit beteiligt werden, sinkt der Beschleunigungsfaktor mit wachsender Problemgröße auf bis zu 128% bei $256 \times 256 \times 128$ Bildpunkten.
Werden zur Berechnung der Verschiebungsvektoren acht Prozessoren eingesetzt, so ist zu erkennen, dass der Beschleunigungsfaktor mit wachsender Problemgröße für alle Agglomerationsstrategien stetig steigt. Wegen der geringen Anzahl von arithmetischen Operationen, die ein Prozessor auf den groben Gittern auszuführen hat bevor er mit seinen Nachbarn kommuniziert (communikation-overhead), ist es hierbei im Allgemeinen kein Vorteil, wenn alle Prozessoren auf den groben Gittern an der Arbeit beteiligt sind. So ist bei allen betrachteten Problemgrößen zu beobachten, dass die Rechenzeiten für eine Agglomerationsstrategie wieder ansteigen, wenn die Daten zu spät (d.h. auf einem zu groben Gitter) auf weniger Prozessoren zusammengezogen werden. Den maximalen Beschleunigungsfaktor von 729% erreicht man somit für $256 \times 256 \times 128$ Gitterpunkte, wenn auf der vierten Grobgitterebene die Berechnung nur noch auf vier Prozessoren durchgeführt wird. Für die kleineren Problemgrößen

wird die maximale Beschleunigung erreicht, wenn die Daten bereits auf der dritten Grobgitterebene auf weniger Prozessoren zusammengezogen werden. Hierdurch wird die Berechnung bei $64 \times 64 \times 64$ Bildpunkten auf 375% und bei $128 \times 128 \times 128$ Bildpunkten auf 594% beschleunigt.

Insgesamt ist also zu beobachten, dass die Berechnung der Verschiebungsvektoren durch die in diesem Kapitel beschriebenen Parallelisierungsstrategien durch hinzunahme weiterer Prozessoren auf etwa 180% bei zwei und bis zu 720 % bei acht Prozessoren beschleunigt werden kann. Das entspricht einer parallelen Effizienz von etwa 90%. Hierbei ist zu betonen, dass die erreichten Rechenzeiten in Abhängigkeit vom Multiprozessorsystem zu verstehen sind und keine feste Eigenschaft des beschriebenen parallelen Algorithmus sind.

P	P aktiv [2]	$T(P)$ [Sek.]	$S(P)$ [%]	$E(P)$ [%]
1	1,1,1,1,1	14.25	100	100
2	2,1,1,1,1	10.33	137	68
4	4,2,1,1,1	6.46	220	55
8	8,4,2,1,1	4.78	298	37
2	2,2,1,1,1	8.65	164	82
4	4,4,2,1,1	5.03	283	70
8	8,8,4,2,1	3.79	375	46
2	2,2,2,1,1	8.26	172	86
4	4,4,4,2,1	4.96	287	71
8	8,8,8,4,1	4.50	316	40
2	2,2,2,2,1	7.75	183	91
4	4,4,4,4,1	4.78	298	74
8	8,8,8,8,1	4.38	325	40

Tabelle 6.1: Rechenzeiten, speed-up und Effizienz des parallelen Programms mit max. Anzahl von Gitterebenen für $N = 64 \times 64 \times 64$ Gitterpunkte.

[1] Geschätzt durch $T_1^N \approx 2 \cdot T_1^{\frac{N}{2}}$.

[2] Anzahl der aktiven Prozessoren auf den Gitterebenen von links (feinstes Gitter) nach rechts (gröbstes Gitter).

P	P aktiv [2]	$T(P)$ [Sek.]	$S(P)$ [%]	$E(P)$ [%]
1	1,1,1,1,1,1	114[1]	100	100
2	2,1,1,1,1,1	82.93	137	68
4	4,2,1,1,1,1	48.34	235	58
8	8,4,2,1,1,1	25.14	453	56
2	2,2,1,1,1,1	64.33	177	88
4	4,4,2,1,1,1	34.70	328	82
8	8,8,4,2,1,1	19.19	594	74
2	2,2,2,1,1,1	67.46	168	84
4	4,4,4,2,1,1	34.05	334	83
8	8,8,8,4,2,1	19.23	592	74
2	2,2,2,2,1,1	63.20	180	90
4	4,4,4,4,2,1	30.64	372	93
8	8,8,8,8,4,1	19.98	570	71

Tabelle 6.2: Rechenzeiten, speed-up und Effizienz des parallelen Programms mit max. Anzahl von Gitterebenen für $N = 128 \times 128 \times 128$ Gitterpunkte.

P	P aktiv [2]	$T(P)$ [Sek.]	$S(P)$ [%]	$E(P)$ [%]
1	1,1,1,1,1,1	456[1]	100	100
2	2,1,1,1,1,1	355.53	128	64
4	4,2,1,1,1,1	195.90	232	58
8	8,4,2,1,1,1	96.11	474	59
2	2,2,1,1,1,1	270.04	168	84
4	4,4,2,1,1,1	136.73	335	83
8	8,8,4,2,1,1	65.42	697	87
2	2,2,2,1,1,1	251.50	181	90
4	4,4,4,2,1,1	130.13	350	87
8	8,8,8,4,2,1	62.55	729	91
2	2,2,2,2,1,1	248.98	183	92
4	4,4,4,4,2,1	138.00	330	82
8	8,8,8,8,4,1	66.07	690	86

Tabelle 6.3: Rechenzeiten, speed-up und Effizienz des parallelen Programms mit max. Anzahl von Gitterebenen für $N = 256 \times 256 \times 128$ Gitterpunkte.

[1] Geschätzt durch $T_1^N \approx 2 \cdot T_1^{\frac{N}{2}}$.

[2] Anzahl der aktiven Prozessoren auf den Gitterebenen von links (feinstes Gitter) nach rechts (gröbstes Gitter).

Kapitel 7

Zusammenfassung

Das Ziel der vorliegenden Arbeit war die Identifikation von Parametern (physikalischer Größen) eines physikalischen Systems mit gegebenem Input und Output in Form von digitalen Bilddaten. Bei den in dieser Arbeit betrachteten Anwendungen aus der Medizin und Meteorologie handelt es sich, bei den Parametern, um Verschiebungsvektoren, die die Objekte (Gehirne bzw. Wolken) innerhalb der Bilddaten assimilieren. Das Problem fällt in die Klasse schlecht gestellter inverser Probleme und zeichnet sich dadurch aus, dass Lösungen, wenn solche existieren, weder eindeutig noch stabil sind.

Ausgangspunkt für die entwickelten Lösungsalgorithmen war die Methode der kleinsten Quadrate, die auf der Minimierung des nichtlinearen Defektnormquadrats zwischen den Datensätzen beruht.
Hierbei wurden zum einen linearisierte Lösungsansätze betrachtet, die durch Linearisierung des Defektfunktionals Abstiegsrichtungen berechnen. Es wurde ein iteratives Verfahren (Landweber-Iteration) für dieses Problem vorgestellt, bei dem die linearisierten Teilprobleme durch eine innere Iteration (Mehrgitter CS) näherungsweise gelöst werden. Ein wesentlicher Bestandteil zur stabilen Berechnung von Näherungslösungen ist hierbei die Skalierung der Abstiegsrichtung, durch die Bestimmung des Regularisierungsparameters innerhalb eines eindimensionalen Unterproblems. In anderen auf einer Linearisierung basierenden Arbeiten zur Anpassung medizinischer Bilddaten (zum Beispiel in [1], [4] und [8]) wird dieses Problem durch die "trial and error" Methode gelöst.

Zum andern wurden zwei verschiedene Lösungsverfahren entwickelt, welche in der

aktuellen Forschung bisher unbeachtet blieben. Die Verfahren ermitteln durch einen Variationsansatz direkt Minimallösungen des nichtlinearen Defektfunktionals. Für beide Ansätze wurde das Prinzip der Regularisierung verwendet um Instabilitäten zu begegnen. Als Regularisierungsterm wurde eine aus der linearen Elastizitätstheorie bekannte Bilinearform gewählt, die die Datensätze wie einen elastischen Körper modelliert. Eine wesentliche Schwierigkeit war die Bestimmung der durch die Regularisierung eingeführten Regularisierungsparameter innerhalb der Lösungsalgorithmen. Hierfür wurden für die verschiedenen Verfahren unterschiedliche Strategien entwickelt, die einen Kompromiss zwischen der Stabilisierung der Verfahren und der Approximation des inversen Problems bilden.

Wegen der enormen Datenmengen in den praktischen Anwendungen, die in der Medizin bei zweidimensionale Schnittbilder typischer Weise 512×512 Bildelemente, bei dreidimensionalen Volumendatensätzen $256 \times 256 \times 128$ Bildelemente und bei Aufnahmen der Erdoberfläche bis zu 4096×4096 Bildelemente enthalten können, erfolgt die numerische Berechnung der Verschiebungsvektoren als Lösung der aus einem Variationsansatz resultierenden partiellen Differentialgleichungen durch Mehrgitterverfahren. Sie zeichnen sich gegenüber anderen Verfahren zur iterativen Berechnung der Verschiebungsvektoren als Lösung einer partiellen Differentialgleichung (wie etwa [4] mit den Jacobi Gesamtschrittverfahren oder [8] mit dem sukzessiven Überrelaxationsverfahren) dadurch aus, dass die Konvergenzraten im Allgemeinen unabhängig von der gewählten Schrittweite h ist. Somit ist der asymptotische Aufwand ($\mathcal{O}(N)$) zur Berechnung der Verschiebungsvektoren für zwei Bilddatensätze mit jeweils N Bildelementen. Zur Diskretisierung der Differentialgleichungen wird das Diskretisierungsgitter für die Verschiebungsvektoren so ausgerichtet, dass die Gitterpunkte gerade in der Mitte der Bildelemente postiert sind (cell centered Diskretisierung). Die Approximation der Gleichung erfolgt durch die Finite Differenzen Methode (FDM). Hierbei werden die auftretenden Ableitungen durch Differenzenquotienten zweiter Ordnung ersetzt.
Zur weiteren Beschleunigung der Verfahren wird in Kapitel 6 das Parallelisierungspotential der Minimierungsverfahren diskutiert und ein geeigneter paralleler Algorithmus entwickelt. Die parallele Implementierung der Gitteralgorithmen auf Parallelrechnern mit verteiltem Speicher basiert auf der Methode des Grid Partitio-

ning. Hierbei werden die digitalen Bilddaten in verschiedene Teilbilder zerlegt und auf entsprechend viele Prozessoren verteilt. Hiermit kann durch die beschriebenen Parallelisierungsstrategien und durch Hinzunahme weiterer Prozessoren die Berechnung der Verschiebungsvektoren auf etwa 180% bei zwei und bis zu 720 % bei acht Prozessoren beschleunigt werden. Das entspricht einer parallelen Effizienz von etwa 90%.

Das Prinzip und die Stabilität der Verfahren wird anhand synthetischer und realer Bilddaten aufgezeigt. Die Verfahren zeichnen sich dadurch aus, dass sie innerhalb weniger Iterationsschritte die Datensätze zur Deckung bringen und bei gestörten Referenzbildern (Output Datensätze) Lösungen berechnen, die sich in Abhängigkeit von den Störungen, nur geringfügig von der Lösung des nicht gestörten Problems entfernt.

An dieser Stelle möchte ich mich bei allen bedanken, die mir bei der Entstehung der Arbeit zur Seite gestanden haben. Besonders zu Dank verpflichtet bin ich Herrn Prof. Dr. K. Witsch für sein großes Interesse an meiner Arbeit, die ständige Gesprächsbereitschaft und die konstruktive Kritik. Frau Prof. Dr. M. Hochbruck und Herrn Prof. Dr. F. Jarre danke ich für die Übernahme der Korreferate. Bei Herrn Dipl.-Math. Volker Grimm bedanke ich mich für die Korrekturhinweise. Herrn Prof. Dr. K. Zilles danke ich für die Möglichkeit am C. und O. Vogt Institut für Hirnforschung bei der Lösung aktueller medizinischer Probleme mitzuarbeiten. Der dortigen Bildverarbeitungsgruppe gilt mein Dank für die gute Zusammenarbeit. Bei Herrn Holmlund der Firma Eumetsat bedanke ich mich für das Bildmaterial und für sein Interesse an meiner Arbeit.

Anhang A

Objektorientierte Implementierung

Die Umsetzung der in den vorherigen Kapiteln beschriebenen abstrakten Gebilde wie Gitterpunkte, Gitter und Gitterfunktionen und den auf ihnen basierenden seriellen und parallelen Algorithmen auf einen oder mehrere Computer, soll im Folgenden mit Hilfe von C++ beschrieben werden.
Die Programmiersprache C++ wurde gewählt, weil ihre objektorientierten Sprachbestandteile eine einfache Möglichkeit zur Modellierung komplexer Algorithmen und Datenstrukturen bieten, mit der auch schwierige Problemstellungen fachfremden Wissenschaftlern nahe gebracht werden können.
Eine in C++ implementierte Klasse ist eine Struktur wie in C, die zusätzlich zu den Daten (Attribute des Objekts) noch Funktionen (Methoden) zu deren Manipulation enthält. Die Interaktion der Klassen mit anderen Programmteilen verläuft über eindeutig definierte Schnittstellen und ist deshalb gut zu überblicken.

Die Klasse soll das entsprechende Objekt der realen Welt möglichst gut abbilden. Grundlegend hierbei ist die Klassen so zu konstruieren, dass sie gut zu überschauen sind und alle relevanten Bestandteile logisch und effizient zusammenfassen. Schnittstellen zu anderen Klassen sollen programmtechnisch möglichst effektiv und leicht zu handhaben sein.

Die in dieser Arbeit entwickelten Klassen können in drei unterschiedliche Kategorien unterteilt werden:

- Die Basis-Klassen sollen Gitter und deren Gitterpunkte beschreiben. Sie dienen ausschließlich zur Beschreibung der Struktur und Anordnung von Gittern und

dienen in den weiteren Klassen zur Indexierung der Vektoren.

- Die Vektor-Klassen sollen die auf Gittern definierten Funktionen abbilden. Jedes Objekt einer Vektor-Klasse ist durch das Gitter, auf dem es erklärt ist, und durch seine Funktionswerte (Vektoren) definiert.

- Die Verfahrens-Klassen implementieren die verschiedenen Algorithmen, die in dieser Arbeit entwickelt wurden. Neben den Gitterfunktionen zeichnet ein Verfahren außerdem aus, wie viele und welche Prozessoren an der Lösung des Problems beteiligt sind.

A.1 Die Basis-Klassen

A.1.1 Die Klasse CellXD

Die Klasse CellXD realisiert einen X-dimensionalen ganzzahligen Vektor. Im Kontext der cell centered Diskretisierung kann ein Element vom Typ CellXD die Koordinate eines Bildpunktes oder eine Stelle in einem Verschiebungsfeld repräsentieren. Hierbei sei darauf hingewiesen, dass ein Element vom Typ CellXD nicht den Wert eines Bildelementes enthält, sondern ausschließlich dessen Koordinaten.

A.1.2 Die Klasse GridXD

Durch die Klasse GridXD wird eine Menge von Elementen des Typs CellXD definiert. Ein Element vom Typ GridXD ist durch zwei Elemente P,Q vom Typ CellXD vollständig definiert, so wird durch

```
Grid2D Gitter(P,Q);
```

eine Variable vom Typ Grid2D angelegt, die einen rechteckigen Bereich eines Bildes oder eines Verschiebungsfeldes definiert; dies entspricht einer Menge von Pixelkoordinaten. Auch hier sei darauf hingewiesen, dass durch den Typ GridXD keinerlei Werte eines Verschiebungsfeldes oder eines Bildes definiert sind.

A.2 Vektor-Klassen

Eine Vektor-Klasse realisiert eine Gitterfunktion $\mathcal{F} : G_h \to I\!R^N$. Ein Objekt einer Vektorklasse enthält

- die Gitterstruktur
- und die zugehörigen Daten

einer Gitterfunktion, deren Funktion und Eigenschaften im Folgenden kurz zusammengefasst werden.

A.2.1 Die Klasse ImageXD

Die Klasse ImageXD beschreibt ein X-dimensionales (X=2,3) Grauwertbild. Ein Objekt der Klasse ImageXD wird durch

```
GridXD Gitter(P,Q);
ImageXD Image(Gitter);
```

angelegt. Hierdurch wird ein Ausschnitt einer digitalen Bildmaske mit unterem linken Pixel P und oberem rechten Pixel Q erzeugt. Die Klasse enthält Methoden für den Transfer der Bilddaten auf andere Bildauflösungsstufen sowie Methoden zur Ein- und Ausgabe der Bilddaten in Dateien.

A.2.2 Die Klasse MoverXD

Die Klasse MoverXD beschreibt eine X-dimensionale Gitterfunktion. Ein Objekt der Klasse MoverXD wird durch

```
Grid3D Gitter(P,Q);
MoverXD Gitterfunktion(Gitter);
```

angelegt. Genau wie bei der Klasse ImageXD wird hierdurch ein X-dimensionaler reeller Vektor mit unterem linken Gitterpunkt P und oberem rechten Gitterpunkt Q angelegt.

A.2.3 Die Klasse DeformationXD

Ein Objekt der Klasse DeformationXD entspricht einer vektoriellen Gitterfunktion. Sie enthält X Komponenten vom Typ MoverXD. Durch

```
Grid3D Gitter(P,Q);
Deformation3D Verschiebungsfeld(Gitter);
```

wird ein Objekt `Verschiebungsfeld` der Klasse DeformationXD erzeugt. Die Klasse enthält Methoden für die Transformation der Gitterfunktionen auf eine gröbere oder feinere Auflösungsstufe sowie zur Berechnung der Normen für die Verschiebungen.

A.3 Klassen für die verwendeten seriellen Algorithmen

A.3.1 Die Klasse MultigridSolver

Ein Objekt vom Typ MultigridSolver wird durch Informationen des Randwertproblems (rechte Seite f und dem Regularisierungsparameter α) sowie durch die Mehrgitterparameter (Zyklus-Typ, Glättungsschritte und die Anzahl der Gitter) erzeugt. Durch

```
float alpha;
int relaxPre, relaxCoarse, relaxPost, level, anzahlLevel;
Deformation3D rhs;
...
MultiGridSolver mehrGitterVerfahren(rhs, alpha ,
                                    relaxPre, relaxCoarse, relaxPost,
                                    level, anzahlLevel);
```

wird etwa das Objekt `mehrGitterVerfahren` erzeugt.
Die einzige Schnittstelle zu dem erzeugten Objekt sind die Methoden

```
void multiGridIteration(int it);
Deformation getDeformation();
```

durch

```
int it = 5;
mehrGitterVerfahren.multiGridIteration( it );
```

wird etwa das Mehrgitterverfahren mit 5 Mehrgitteriterationen gestartet. Die berechnete Verschiebung erhält man durch

```
Deformation u;
u = mehrGitterVerfahren.getDeformation();
```

von dem verwendeten Objekt. Alle übrigen Methoden zur Implementierung des Mehrgitterverfahrens wie die Berechnung des Defekts, die Relaxation und die Gittertransferoperatoren sind vom Rest des Programms abgekapselt und werden nur innerhalb der Klasse MultigridSolver verwendet. Die Methode `multiGridIteration` ist wie folgt implementiert:

```
public:
...
void multiGridIteration(int it){
     for ( int i = 1 ; i <= it ; i++) {
          cout << i << " -te Mehrgitteriteration" << endl;
          multiGridCycle();
          cout << "max. VV " << deformation.maxVV() << endl;
          }
}
```

sie verwendet die versteckte Methode `multiGridCycle()`:

```
private:
...
void multiGridCycle(){
     if (level == anzLevel){
         relax(relaxCoarse);
         return;
         }
      else {
```

```
            relax(relaxPre);
            defekt();
            Deformation3D coarseRhs = defect.zoom(contract(region));

            MultiGridSolver mgs(coarseRhs, alpha,
                                level+1 , anzLevel ,
                                relaxPre , relaxCoarse , relaxPost);

             mgs.multiGridCycle();

             deformation += (mgs.getDeformation()).zoom(region);

             relax(relaxPost);

        }
}
```

die eine Iteration des MG-CS mit einem V-Zyklus durchführt. Die vom Rest des Programms abgekapselten Methoden `relax()`, `defect()` und `zoom()` implementieren die benötigten Mehrgitterkomponenten.

A.3.2 Die Klasse ElasticMatcher

Die Klasse ElasticMatcher implementiert die Minimierung des Defektfunktionals auf einer Auflösungsstufe. Ein Objekt der Klasse ElasticMatcher wird durch die Angabe des Template- sowie des Referenzbildes wie folgt

```
Image3D Template , Reference;
...
ElasticMatcher elasticMatcher( Template , Reference);
```

erzeugt. Die Anweisung zur Berechnung der Verschiebungsvektoren wird dem Objekt durch

```
elasticMatcher.match();
```

mitgeteilt.

A.3.3 Die Klasse MultiElasticMatcher

Durch ein Objekt der Klasse MultiElasticMatcher werden die Verschiebungsvektoren auf mehreren Auflösungsstufen berechnet. Ein Objekt der Klasse MultiElasticMatcher wird durch die Angabe des Templatebildes und des Referenzbildes wie folgt

```
Image Template , Reference;
...
MultiElasticMatcher multiElasticMatcher( Template , Reference ,
                                         level , anzahlLevel);
```

erzeugt. Die Klasse MultiElasticMatcher besitzt drei Instanzvariablen,

```
ElasticMatcher em ;
int          level;
int          anzLevel;
```

die ein Objekt charakterisieren. In `em` wird das Minimierungsverfahren für die Gitterebene `level` festgelegt. Die Gesamtanzahl der Gitter ist in `anzLevel` angegeben. Das Minimierungsverfahren auf mehreren Auflösungsstufen wird durch

```
multiElastMatcher.multiElastCycle();
```

gestartet. Hierdurch wird rekursiv auf allen Auflösungsstufen

```
void multiElastCycle(){

  if (level == anzLevel){
      em.match();
      return;
  } else{
      Image3D T_coarse = (em.getTemplate()).zoom(contract(grid));
      Image3D R_coarse = (em.getReference()).zoom(contract(grid));
```

```
      MultiElastMatcher mem( T_coarse , R_coarse , level+1 , anzLevel);
      mem.multiElastCycle();

      em.setDeformation((mem.getDeformation()).zoom(grid));
      em.match();
  }
}
```

gestartet.

A.4 Klassen für die verwendeten parallelen Algorithmen

Die folgenden Klassen setzen voraus, dass ihre Objekte innerhalb eines MPI-Programms angelegt werden. Das MPI-Programm besteht aus einer Menge von autonomen Prozessen, von denen jeder sein eigenes Objekt der entsprechenden Klassen anlegt. Das angelegte Objekt enthält die für den Prozessor relevanten Daten. Die Kommunikation zwischen den Prozessoren durch MPI findet innerhalb der Methoden für die verschiedenen Objekte statt.

A.4.1 Die Klasse MultiLevelGridPartitioning

Die Klasse MultiLevelGridPartitioning übernimmt die Partitionierung des Gitters in mehrere Teilgitter nach der Methode des Grid Partitioning (vgl. Kapitel 6.1.2). Am Anfang jedes parallelen Mehrgitterprogramms wird ein Objekt der Klasse MultiLevelGridPartitioning durch den Aufruf des Konstruktors

```
MultiLevelGridPartitioning *hierarchy =
new  MultiLevelGridPartitioning( regionOfInterest , 5 , numprocs , 2 );
```

erzeugt. Hierdurch wird die Gitterstruktur für alle Gitterebenen erzeugt und in den Attributen (vgl. Tabelle A.1) gespeichert.

A.4.2 Klassen zur Realisierung der Teilgitterstruktur

Zur Konstruktion der Teilgitter auf den verschiedenen Auflösungsstufen werden die vier Vektor-Klassen aus Abschnitt A.2 erweitert. Neben den lokalen Algorithmen,

Typ	Name	Beschreibung
int	levels	Anzahl der Gitterebenen.
int	*nblocks;	Anzahl der Teilgitter pro Gitterebene.
Grid3D	*fullGrid	Gesamtes Gitter auf jeder Gitterebene.
Grid3D	**internal	Teilgitter ohne Überlappung
Grid3D	**ghosts	Teilgitter mit Überlappung
Grid3D	**f2c	Grobgitterpartitionierung des feinen Gitters.

Tabelle A.1: Attribute der Klasse MultiLevelGridPartitioning.

wie etwa die Berechnung der maximalen Verschiebung auf einem Teilgitter, stellen die Klassen Methoden für den Transfer der Daten zwischen den Gitterebenen bereit.

A.4.3 Die Klasse MultiGridSolverParallel

Ein Objekt der Klasse MultiGridSolverParallel wird auf den verschiedenen Bildauflösungsstufen zur Berechnung der Verschiebungsvektoren eingesetzt. Dem Konstruktor wird neben den Mehrgitterparametern ein Objekt vom Typ MultiLevelGridPartitioning übergeben. Hierdurch kann auf jeder Gitterebene die Partitionierung der Gitterfunktionen vorgenommen werden. Durch die Attribute `node` und `level` wird

Typ	Name	Beschreibung
int	anzLevel	Anzahl der Gitter
int	level	Aktuelle Gitterebene
int	relaxPre	Anzahl der Vorglättungen
int	relaxCoarse	Lösungsschritte auf gr. Gitter
int	relaxPost	Anzahl der Nachglättungen
int	membershipKey	Zugehörigkeitsschlüssel
int	node	Prozessor Id
MPI_Comm	processorWorld	Zugeh. Gruppe von Proz.
MultiLevelGridPartitioning	*hierarchy	Gitterstruktur
CompOfDeformation3D	deformation	Korrekturen
CompOfDeformation3D	defect	Defekt
CompOfDeformation3D	forces	Rechte Seite
float	alpha	Regularisierungsparameter

Tabelle A.2: Attribute der Klasse MultiGridSolverParallel.

dem Objekt die Prozessor Id und die aktuelle Gitterebene übergeben, so dass die benötigten Gitterfunktionen angelegt werden können. Wie bei der seriellen Klasse sind die Methoden, die die Mehrgitterkomponenten implementieren, vom Rest des Programms abgekapselt. In der Methode `multiGridCycle()` wird der MG-CS Algo-

rithmus parallel implementiert.

```
void multiGridCycle(){

  if (level == anzLevel){
     relax(relaxCoarse);
     return;
  } else {
     relax(relaxPre);
     computeDefect();
     MPI_Comm  processorWorldCoarse = processorSplit();
     ComponentOfDeformation3D defectCoarse =
     defect.parallelRestrict(processorWorld , processorWorldCoarse ,
     membershipKey , level , hierarchy);
     ComponentOfDeformation3D u_coarse;
     if (membershipKey == 0){
           MPI_Comm_rank(processorWorldCoarse , &myrank_coarse);
           // Neues Objekt fuer das grobe Gitter
           MultiGridSolverParallel mgs(defectCoarse , my ,
                                       level+1 , anzLevel ,
                                       relaxPre , relaxCoarse , relaxPost ,
                                       hierarchy , myrank_coarse ,
                                       membershipKey , processorWorldCoarse);
         mgs.multiGridCycle();
         u_coarse = mgs.getDeformation();
     } // Ende MembershipKey == 0
     // Weiter im Takt mit allen Prozessoren auf der feinen ALS
     MPI_Barrier(processorWorld);
     refine( u_coarse , processorWorldCoarse );
     relax(relaxPost);
  } // Ende else
} // Ende Method
```

Das Attribut `membershipKey` dient zur Unterscheidung, ob der Prozessor auf den gröberen Auflösungsstufen aktiv ist oder nicht.

A.4.4 Die Klasse ElasticMatcherParallel

Ein Objekt der Klasse ElasticMatcherParallel führt die Minimierung des Defektfunktionals auf einer Bildauflösungsstufe parallel durch. Der Aufbau der Klasse ElasticMatcherParallel entspricht hierbei der seriellen Version ElasticMatcher, wobei die zugehörigen Methoden wie die Berechnung einer Norm, die Berechnung der Kräfte oder die Transformation des Templatebildes parallelisiert wurden. Zur Berechnung der Verschiebungen wird ein Objekt der Klasse MultiGridSolverParallel verwendet.

A.4.5 Die Klasse MultiElasticMatcherParallel

Ein Objekt der Klasse MultiElasticMatcherParallel führt die Minimierung des Defektfunktionals auf mehreren Bildauflösungsstufen parallel durch. Ein wesentlicher Unterschied zur seriellen Version MultiElasticMatcher ist die Gittertransferphase. Hierbei werden, wie in der Klasse MultiGridSolverParallel, mehrere Teilgitter auf einem Prozessor zusammengezogen. Die Mehrgitterstruktur erhält die Klasse durch das Objekt der Klasse MultiLevelGridPartitioning.

Literaturverzeichnis

[1] Amit Y., *A nonlinear variational problem for image matching*, SIAM J. Sci. Comput., Vol. 15, pp. 207-224, (1994).

[2] Amit Y., Grenander U. and Picconi M., *Structural image restoration thought deformable templates*, J. Amer. Statist. Assoc., 86, pp. 376-387, (1991).

[3] Aubert G. and Vese L., *A variational method in image recovery*, SIAM J. Numer. Anal. Vol. 34, No. 5, pp. 1948-1979, (1997).

[4] Bajcsy R. and Kovacic S.,: *Multiresolution Elastic Matching*, Computer Vision, Graphics and Image Processing 46, 1-21, (1989).

[5] Baumeister J., *Stable Solutions of Inverse Problems.*, Vieweg, (1987).

[6] Braess D., *Finite Elemente*, Springer Verlag, (1996).

[7] Brandt A., *Multi-level Adaptive Solutions to Boundary Value Problems*, Math.Comp., 31, pp. 333-390, (1977).

[8] Christensen G.E., Rabbitt R.D. and Miller M.I., *Deformable templates using large deformation kinematics*, IEEE Trans. on Image Processing, Oct. 1996, 5(10), 1435-1447, (1996).

[9] Czajkowski M., *Diskretisierung und iterative Lösung regularisierter Anfangskontrollprobleme bei der Wellengleichung und den Shallow water equations*, Dissertation, Heinrich-Heine-Universität Düsseldorf, (1995).

[10] Davis M.H., Khotanzad A., Flaming D. and Harms S., *A physics based coordinate transformation for 3D medical images*, IEEE Trans. on medical imaging, 16(3), 317-328.

[11] Dengler J. and Schmidt M., *The dynamic pyramid - a model for motion analysis with controlled continuity*, Int. J. Patt. Rec. Art. Intell. 2 , 275-286, (1988).

[12] Dennis J.E. and Schnabel R.B., *Numerical Methods for Unconstrained Optimization and Nonlinear Equation*, SIAM, Philadelphia, (1996).

[13] Esseling N., *Parallele Algorithmen und Werkzeuge für die Zuordnung funktionaler und anatimischer Daten auf Basis von PET- und MRT-Volumendaten*, Bericht des Forschungszentrum Jülich, (1994).

[14] Engl H.W., Hanke M., and Neubauer A., *Regularization of Inverse Problems*, Kluwer Academic Publ., Dordrecht, (1996).

[15] Geist A., Beguelin A., Dongarra J., Weicheng J., Manchek R. und Sunderam V., *PVM: Parallel Virtual Maschine A User's Guide and Tutorial for Networked Parallel Computing*, (1994).

[16] Geyer S., Ledberg A., Schormann T., Kinomura S., Schleicher A., Bürgel U., Klingberg T., Larsson J., Zilles K., Roland P., *Two different areas within the primary motor cortex of man*, Nature 382 pp. 805-807, (1996).

[17] Hackbusch W., *Iterative Lösung großer schwachbesetzter Gleichungssysteme*, B. G. Teubner, Stuttgart, (1993).

[18] Hansen P.C. and O'Leary, *The use of the L-curve in the regularization of discrete ill-posed problems,* Technical Report UNIC, UMIACS-TR-91-142, (1991).

[19] Hansen P.C., *Analysis of discrete ill-posed problems by means of the L-curve,* Siam Rev., 34 pp. 561-580, (1992).

[20] Hadamard J., *Lectures on the Cauchy Problem in Linear Partial Differential Equations*, Dover Publishing, New York, (1923).

[21] Henn S., *Schnelle elastische Anpassung in der digitalen Bildverarbeitung mit Hilfe von Mehrgitterverfahren*, Diplomarbeit Heinrich-Heine-Universität Düsseldorf, (1997).

[22] Henn, S., Witsch, K., *A Multigrid Approach for Minimizing a Nonlinear Functional for Digital Image Matching*, Computing, Volume 64, Issue 4 pp. 339-348, (2000).

[23] Henn, S., Schormann, T., Engler, K., Zilles, K., Witsch, K., *Elastische Anpassung in der digitalen Bildverarbeitung auf mehreren Auflösungsstufen mit Hilfe von Mehrgitterverfahren*, Springer Series: Informatik-Aktuell, Springer-Verlag, pp. 392-399, (1997).

[24] Horn B.K.P. and Schnuck B.G., *Determining optical flow,* Artificial Intelligence, 17, pp. 185-203, (1981).

[25] Jähne B., *Digitale Bildverarbeitung*, Springer Verlag, (1993).

[26] Louis A.K., *Inverse und schlecht gestellte Probleme*, Teubner, Stuttgart, (1989).

[27] Message Passing Interface Forum, *MPI: A Message Passing Interface Standard.* Computer Science Department, University Of Tennessee, Knoxville, Tennessee, (1995).

[28] Piccioni M., Scarlatti, Trouve A., *A variational problem arising from speech recognition*, SIAM J. Appl. Math., Vol. 58, No. 3, pp. 753-771, (1998).

[29] Spenzler U., Siehl H. S. und Gilsbach J. M., *Navigated Brain Surgery: Interdisciplinary Views of Neuronavigation from Neurosurgeons and Computer Scientists*, Wissenschaftsverlag Mainz in Aachen, (1999).

[30] Stüben K., Trottenberg U., *Multigrid Methods: Fundamental Algorithms, Model Problem analysis and Applications*, Lecture Notes in Mathematics, 960, Springer Verlag, (1982).

[31] Terzopoulos D., *Image analysis using multigrid relaxation methods,* IEEE Transactions on pattern analysis and machine intelligence, Vol. Pami-8, No. 2, (1998).

[32] Tikhonov, A.N. and Arsenin, *Solutions of Ill-Posed Problems,* John Wiley & Sons, (1977).

[33] TIKHONOV, A.N., *Solution of incorrectly formulated problems and the regularization method,* Soviet Math Dokl. pp. 1035-1038, (1963).

[34] TIKHONOV, A.N., *Regularization of incorrectly posed problems,* Soviet Math Dokl. pp. 1624-1627, (1963).

[35] THOMPSON P. AND TOGA A., *Anatomically driven strategies for high-dimensional brain image registration and pathology,* Brain Warping, Academic Press 311-336, (1998).

[36] TROTTENBERG U., OOSTERLEE C., SCHÜLLER A., *Multigrid,* Academic Press, (2000).

[37] VELTE W., *Direkte Methoden der Variationsrechnung,* B.G. Teubner, Stuttgart, (1976).

[38] WESSELING P., *An inroduction to multigrid methods*, John Wiley & Sons Ltd., (1992).