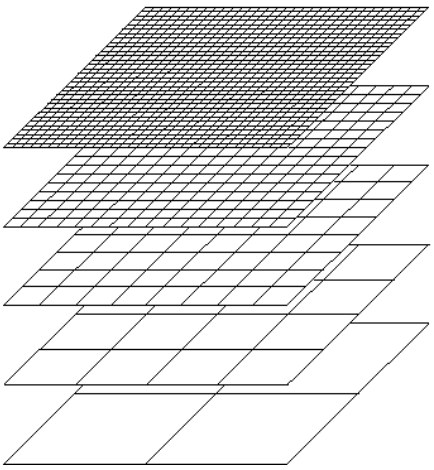
technische universität dortmund

fakultät für mathematik

Bachelorarbeit

Numerischer Vergleich nichtlinearer und linearer Mehrgitterverfahren zum Lösen von partiellen Differentialgleichungen



am Lehrstuhl III: Angewandte Mathematik und Numerik
der Fakultät für Mathematik, TU Dortmund

vorgelegt von

Mirco Altenbernd

betreut durch

Dr. Matthias Möller

September 2013

Inhaltsverzeichnis

Kapitel 1 Einleitung

Diese Arbeit befasst sich mit dem numerischen Vergleich von unterschiedlichen Mehrgitteransätzen zur Lösung nichtlinearer partieller Differentialgleichungen am Beispiel eines Diffusions-Reaktions Problems mittels eines konformen Finite Elemente Ansatzes.

Bei der Lösung von partiellen Differentialgleichungen mit Hilfe der Finite Elemente Methode (FEM) müssen oftmals große dünnbesetzte Gleichungssysteme gelöst werden. Aufgrund der Besetzungsstruktur dieser Matrizen ist dabei eine Berechnung der Lösung mit iterativen Lösungstechniken üblich. Jedoch sind viele gängige iterativen Löser in ihrer Konvergenzrate und -geschwindigkeit von der Gitterweite abhängig, so dass bei Problemen größerer Dimension nicht nur mehr Rechenoperationen, aufgrund der gestiegenen Anzahl an Unbekannten, notwendig sind, sondern auch mehr Iterationen zur Berechnung der Lösung benötigt werden. Bei einem einfachen Konjugierte Gradienten-Verfahren (CG) verdoppeln sich zum Beispiel die benötigten Iterationen bei einer regulären Verfeinerung des Gitters und zusätzlich verursacht eine Matrix-Vektor Multiplikation in etwa viermal so viele mathematische Operationen (2D, bilineare Finite Elemente).
Mit dem Mehrgitterverfahren liegt insbesondere für lineare Probleme ein alternatives iteratives Lösungsverfahren vor, welches optimale (d.h. gitterweitenunabhängige) Konvergenzraten ermöglichen kann.

Betrachtet man nun nichtlineare Probleme, so kann dieses dem Mehrgitterverfahren für lineare Probleme zugrundeliegende Prinzip nicht direkt übertragen werden, weshalb man Modifikationen an dem Verfahren durchführen muss, um ein ähnliches Konvergenzverhalten erzielen zu können. Wir wollen in dieser Arbeit auf zwei unterschiedliche Varianten dieser Umsetzung eingehen.

Das erste vorgestellte Verfahren ist ein in der Praxis häufig verwendeter Ansatz, um die guten Konvergenzeigenschaften des klassischen linearen Mehrgitterverfahrens bei der Lösung nichtlinearer Probleme auszunutzen. Dabei wird das nichtlineare Problem auf dem Feingitter in geeigneter Weise linearisiert und die entstehenden linearen Teilprobleme mit Hilfe des klassischen Mehrgitterverfahrens gelöst. Dieses Verfahren benutzt jedoch nicht den grundlegenden Mehrgitteransatz, der versucht über die Informationen auf den gröberen Gittern die Lösung auf dem Feingitter zu verbessern, da bei der Aktualisierung der Lösung in einem Schritt keine Updates der Systeme auf den Gittern geschehen.
Das sogenannte **Full Approximation Scheme** (FAS) versucht daher den Grundgedanken des Mehrgitteransatzes direkter auf das nichtlineare Problem zu übertragen. Dabei wird jedoch nicht nur eine einfache Defektkorrektur auf den Grobgittern durchgeführt, sondern eben, wie der Name erwarten lässt, eine volle Approximation der Lösung auf den Gitterleveln berechnet und zur Aktualisierung der Feingitterlösung genutzt.

Die Grundlage dieser Arbeit stellen dabei insbesondere die Veröffentlichungen von Borzì [2] und Henson [3] dar. Vor allem die Herleitungen der einzelnen Verfahren orientieren sich an diesen.

Bei der Vorstellung und Demonstration der Algorithmen wollen wir so vorgehen, dass möglichst die Zusammenhänge und Unterschiede zwischen linearem und nichtlinearem Mehrgitterverfahren deutlich werden. Demzufolge beginnen wir einleitend mit der Grundidee des Mehrgitteransatzes, welche zwei eigentlich langsame Lösungskomponenten zu einer mächtigen kombiniert. Anschließend wollen wir in **Kapitel 3** die Funktionsweisen und Voraussetzungen an die einzelnen Komponenten des Mehrgitteralgorithmus vorstellen und zum Teil an Beispielen erläutern. Jedoch werden wir keine

Beweise durchführen, sondern verweisen dafür auf entsprechende bereits vorhandene Literatur.
Im Anschluss daran werden wir die beiden zu vergleichenden Algorithmen im Detail für den einfachen V-Zyklus motivieren und an einem Zweigitter-Algorithmus herleiten, sowie eine einfache algorithmische Darstellung angeben. Außerdem wollen wir auf entscheidende Unterschiede der beiden Methoden eingehen.
Schließlich werden wir in **Kapitel 5** numerische Tests zum Vergleich der vorgestellten Methoden anhand einer zweidimensionalen nichtlinearen Diffusions-Reaktions Gleichung durchführen und evaluieren. Dabei basieren die vorgestellten Tests zu den Algorithmen auf einer eigenen Implementierung in *C++*. Die Vergleichbarkeit der Ergebnisse wird dadurch ermöglicht, dass alle Algorithmen auf der gleichen Codebasis und somit mit den selben Optimierungen durchgeführt werden.
Insbesondere vergleichen wir dabei auch die Effektivität bei der Anwendung unterschiedlicher Linearisierungsmethoden der Hilfsprobleme. Wir werden sowohl klassische Newton-Linearisierung anwenden, als auch eine einfache Defektkorrektur, um insbesondere auch die Verwendbarkeit der Methoden bei Problemen mit nur schwer oder gar nicht analytisch zu bestimmenden Jacobimatrizen zu überprüfen.
Schlussendlich werden wir ein Resümee ziehen, um die Anwendbarkeit und eventuell die Eignung der Algorithmen als Löser für nichtlineare partielle Differentialgleichungen auszuwerten und einen Ausblick auf mögliche Modifikationen und Optimierungen geben.

Kapitel 2 Das Mehrgitterprinzip

Einleitend wollen wir die Idee des Mehrgitterverfahren beschreiben und insbesondere die grundlegende Motivation, sowie die Bedeutung der einzelnen Komponenten erläutern. Auf genauere Eigenschaften und Voraussetzungen eben dieser Komponenten werden wir aber erst in den folgenden Kapiteln eingehen.

2.1 Das Grundprinzip

Das Mehrgitterverfahren

Beim Mehrgitterverfahren handelt es sich um eine iterative Methode zum Lösen von Gleichungssystemen. Diese benutzt nicht nur das zu lösende Gleichungssystem, sondern außerdem Gleichungssysteme niedrigerer Dimensionen, die durch strukturelle Zusammenhänge erzeugt werden.
Beim hier vorgestellten geometrischen Ansatz stellt eine passende Gitterhierarchie diesen Zusammenhang dar. Eine alternative Vorgehensweise ist das algebraische Mehrgitter, welches auf Basis der Koeffizienten der Systemmatrix versucht eine hierarchische Struktur von Gleichungssystemen zu erzeugen.

Die Besonderheit dieser Mehrgitterverfahren liegt in der Möglichkeit eine von der Gitterweite unabhängige Konvergenzgeschwindigkeit erzielen zu können. Im Gegensatz zu üblichen iterativen Lösungsmethoden steigt die benötigte Iterationszahl somit eben nicht bei größer werdenden Systemen.
Diese Verfahren wurden für das Lösen poissonartiger Probleme entwickelt, eignen sich jedoch häufig auch gut zur Bestimmung der Lösung andersartiger partieller Differentialgleichungen.

Die Funktionsweise

Die Grundidee des Mehrgitterverfahrens basiert auf einer Verbesserung der Lösung des Feingitters durch auf gröberen Gittern (d.h. weniger Unbekannte) bestimmte Korrekturwerte. Mit Hilfe der auf dem Feingitter bestimmen Lösung wird auf dem gröberen Gitter ein Hilfsproblem aufgestellt, das zur Ermittlung einer solchen Korrektur genutzt werden kann. Dies ist das sogenannte Defektproblem.

Defektproblem - das zur Ermittlung einer Lösungsverbesserung aufgestellte Problem

Die Aufstellung dieses Defektproblems ist im Falle von linearen Gleichungssystemen vergleichsweise einfach. Für die Anpassung an nichtlineare Probleme gibt es mehrere Umsetzungsmöglichkeiten, von denen wir später zwei genauer vorstellen wollen.

Essentiell für Korrekturen der Feingitterlösung ist ein entsprechender Transfer der Funktionen zwischen den verschiedenen Gitterleveln. Die Bezeichnung Gitterlevel steht dabei für die verschiedenen Verfeinerungsstufen des Basisgitters. Im Folgenden werden wir das Basisgitter (geringste Anzahl an Unbekannten) mit Gitterlevel $k = 0$ bezeichnen und die weiteren Gitter, bis zum feinsten Gitterlevel $k = L$, fortlaufend nummerieren.

Gitterlevel - ein Element der zugrundeliegenden Gitterhierarchie

Beim geometrischen Mehrgitteransatz kann der Gittertransfer beispielsweise durch Anwendung geeigneter Interpolationsmatrizen, die gewissen Eigenschaften genügen, bewerkstelligt werden. Dabei wird der Transfer auf ein feineres Gitter als Prolongation und der Transfer auf ein gröberes Gitter als Restriktion bezeichnet. Auf die genaue Funktionsweise werden wir ebenfalls in **Kapitel 3** eingehen.

Prolongation - Transfer auf ein feineres Gitter
Restriktion - Transfer auf ein gröberes Gitter

Die Verbesserung der Lösung allein durch einen auf einem gröberen Gitter bestimmten Korrekturwert garantiert uns jedoch nicht die gewünschte, von der Systemgröße unabhängige, Konvergenzgeschwindigkeit. Das Verfahren alleine garantiert nicht einmal Konvergenz.

Grobgitterkorrektur - Korrektur der Lösung durch den mit Hilfe des Defektproblems auf dem groben Gitter bestimmten Korrekturwert

Entscheidend für die Konvergenz ist eine weitere Komponente, die im Zusammenspiel mit der Korrektur durch den vom Defektproblem bestimmten Korrekturwert, dieses optimale Resultat erst ermöglicht. Diese Komponente ist der sogenannte Glätter. Bei diesem handelt es sich um ein Verfahren, das eigenständig als Löser nicht unbedingt gut geeignet sein muss, jedoch die Eigenschaft besitzt, auf gewissen Fehleranteilen der aktuellen Lösung zu arbeiten und diese entscheidend zu reduzieren.
Der Fehler wird dafür in hochfrequente und niederfrequente Anteile unterteilt.

nieder-/hochfrequente Fehleranteile - Aufteilung des Systemfehlers

Hochfrequente Anteile lassen sich bei der Übertragung auf ein gröberes Gitter schlecht darstellen, was zu Problemen bei der Berechnung führen kann. Außerdem werden durch eine Restriktion niederfrequente zu hochfrequenten Fehleranteilen. Daher ist der Glätter so gewählt, dass er auf dem aktuellen Gitter hochfrequente Fehleranteile in niederfrequente überführt. Diese werden dann durch eine Restriktion zu hochfrequenten Fehleranteilen auf dem gröberen Gitter und können dort wieder durch den Glätter bearbeitet werden.

Glätter - Operator der hochfrequente in niederfrequente Fehleranteile überführt

Einfache Verfahren in dieser Kategorie stellen das Jacobi- und Gauss-Seidel-Verfahren dar. Diese arbeiten bei der Anwendung auf ein Problem effektiv vor allem im hochfrequenten Fehlerbereich. Allerdings gibt es auch komplexere Varianten für diesen Glättungsoperator.

Das Zusammenspiel aus Grobgitterkorrektur und Glätter kann, unter gewissen Voraussetzungen, eine von der Systemgröße unabhängige Konvergenzgeschwindigkeit ermöglichen. Außerdem stellt die Anwendung des Glätters, unter der Annahme, dass das Defektproblem „billig“ gelöst werden kann, die numerisch aufwendigste Komponente der Lösungsroutine dar. Unter „billig“ verstehen wir hierbei, dass das entsprechende Gleichungssystem ohne großen Aufwand gelöst werden kann.

Das genaue Zusammenspiel der beiden Komponenten wollen wir in **Kapitel 4** für die verschiedenen Mehrgittervarianten beschreiben.

2.2 Zyklustypen

Beim Übergang von einem einfachen Lösungsprozess mit nur zwei Gittern zu einem mit mehreren Gittern ergibt sich für das Durchlaufen der einzelnen Gitterlevel eine Vielzahl an Möglichkeiten. Diese verschiedenen Varianten liefern durchaus unterschiedliche Konvergenz- und Robustheitseigenschaften, so dass wir im Folgenden die drei am häufigsten verwendeten Varianten vorstellen wollen.
Außerdem werden wir das „geschachtelte“ Mehrgitterverfahren beschreiben, welches insbesondere bei den hier untersuchten nichtlinearen Problemen von Bedeutung ist.

2.2.1 V-, W- und F-Zyklus

Wie angesprochen wollen wir zunächst die drei üblichen Strategien zum Durchlaufen der Gitterhierarchie vorstellen. Die Informationen zu Robustheit und numerischem Aufwand entstammen dabei hauptsächlich [7] und [8].

Ein einfacher Zyklus reicht dabei oft nicht aus, um eine Lösung mit gewünschter Genauigkeit zu bestimmen, so dass in der Regel mehrere iterative Anwendungen erfolgen.

Zum Verständnis betrachten wir nun eine Gitterhierarchie aus vier Gittern. Dabei geschieht das Durchlaufen der Struktur durch einen rekursiven Aufruf, d.h. bis hin zum gröbsten Gitter werden zum Lösen der einzelnen Gleichungssysteme Defektprobleme erstellt. Dies führt dazu, dass erst auf dem gröbsten Gitter das Defektproblem exakt, oder zumindest mit hoher Genauigkeit, gelöst wird.

V-Zyklus

Der einfachste aller Zyklustypen ist der V-Zyklus. Hierbei wird die Gitterhierarchie, wie der Name erschließen lässt, V-förmig durchlaufen.
Dies geschieht dadurch, dass das jeweilige Defektproblem pro Gitterlevel durch einen einmaligen Aufruf eines neuen Defektproblems gelöst wird. Erst auf dem gröbsten Gitter erfolgt eine exakte Lösung des Problems, so dass mit dem dort ermittelten Korrekturwert sukzessive eine Korrektur der Lösungen auf den feineren Gittern durchgeführt werden kann.

Wichtig ist hierbei, dass auf jedem Gitterlevel eine gewisse Anzahl an Glättungsschritten durchgeführt werden muss.

Eine grafische Darstellung dieses Ablaufs ist in Abbildung 2.1 zu sehen. Dort werden die Prolongation- und Restriktionsoperationen bereits in der später verwendeten Form dargestellt.

Der Vorteil des V-Zyklus liegt insbesondere im geringen numerischen Aufwand im Vergleich zu den anderen Varianten (vgl. [7, S. 50 ff.]). Jedoch stellt dieser Typ vergleichsweise hohe Anforderungen an die Lösung, sowie die rechte Seite des zu lösenden Systems. Insbesondere muss die Lösung auf dem betrachteten Gebiet Ω aus dem Sobolev-Raum $\mathcal{H}^2(\Omega)$ und die rechte Seite aus $\mathcal{L}^2(\Omega)$ stammen, um eine gute Konvergenzgeschwindigkeit erzielen zu können (siehe auch [8]).

W-Zyklus

Die nächstliegendste Variation ist der sogenannte W-Zyklus. Hierbei wird durch Mehrfachanwendung des oben beschriebenen V-Zyklus eine W-Form beim Durchlaufen der Gitterlevel erzeugt (vgl. Abbildung 2.1).

Die Lösung der Defektprobleme wird nun durch einen zweifachen anstelle eines einfachen Aufrufs des V-Zyklus berechnet. Durch diese Modifikation wird pro Iteration nicht nur einmal eine Lösung auf dem gröbsten Gitter ermittelt, sondern $2^{(L-1)}$-mal, wobei L der Nummer des feinsten Gitters entspricht.

Diese Variante stellt somit einen numerischen Mehraufwand im Vergleich zum V-Zyklus dar, ist aber sehr robust, d.h. dass hier im Vergleich zum V-Zyklus weniger Glattheitsanforderungen an die Lösung notwendig sind. Der W-Zyklus ermöglicht in der Regel bereits, wenn die Lösung aus dem Sobolev-Raum $\mathcal{H}^{1+\alpha}(\Omega)$ mit $\alpha > 0$ stammt, ein gutes Konvergenzverhalten (vgl. [8]).
Des Weiteren benötigt der W-Zyklus, durch seine mehrfache Korrektur im Allgemeinen weniger Iterationen und Glättungsschritte als der weniger komplexe V-Zyklus.

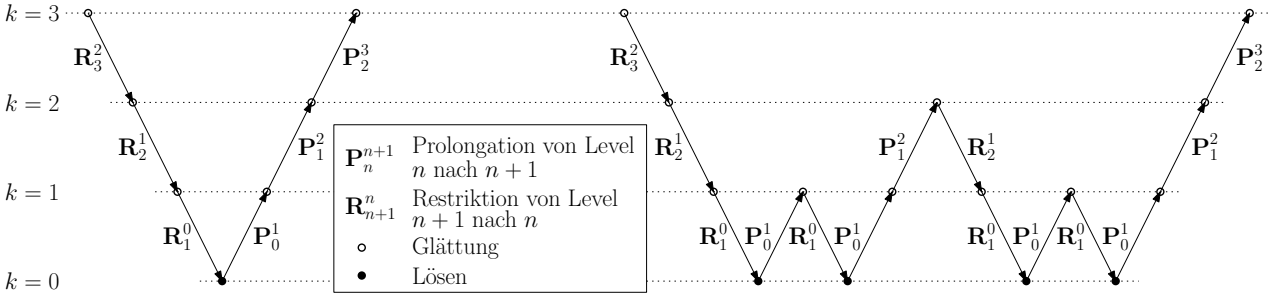


Abbildung 2.1: Grafische Darstellung des V- und W-Zyklus.

F-Zyklus

Da der vorgestellte W-Zyklus einen deutlichen Mehraufwand gegenüber dem V-Zyklus darstellt, dafür jedoch bessere Robustheitseigenschaften garantiert, ist es naheliegend zu versuchen den numerischen Aufwand zu reduzieren ohne an Robustheit zu verlieren. Dies führt uns zum F-Zyklus.

Der F-Zyklus wird als ein Kompromiss aus dem V- und W-Zyklus verstanden, ist jedoch nicht mehr einfach in rekursiver Form zu beschreiben.
Anschaulich ist in Abbildung 2.2 zu erkennen, dass anfangs, zur Bestimmung der Korrektur, bis auf das gröbste Gitter gegangen wird, um im Folgenden nach der Prolongation jeweils eine einfache Anwendung eines V-Zyklus zu Verbesserung auszuführen. Hierbei wird das Problem auf dem gröbsten Gitter nur noch L-mal gelöst.

Vom numerischen Aufwand befindet sich der F- zwischen V- und W-Zyklus, besitzt aber dennoch vergleichbare robuste Eigenschaften wie der komplexere W-Zyklus (vgl. [8]). In der Praxis wird diese Variante daher oft benutzt.

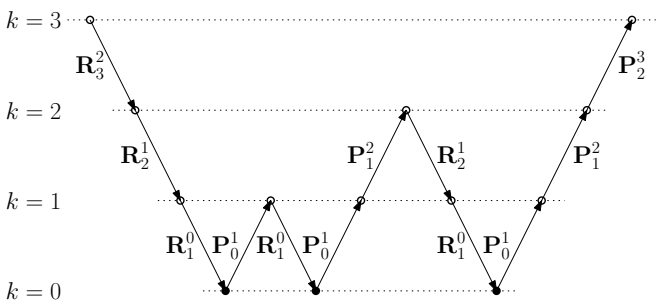


Abbildung 2.2: Grafische Darstellung des F-Zyklus.

2.2.2 „Geschachteltes“ Mehrgitter

Die bisher vorgestellten Varianten beginnen alle mit einer initialen Startlösung auf dem Feingitter, um von dort, mit Hilfe von Defektproblemen auf den gröberen Gittern, eine Lösungskorrektur durchzuführen. Dabei ist die Wahl einer geeigneten Startlösung nicht immer trivial. Eine Idee ist es daher, zuerst auf dem gröbsten Gitter, das eine geringere Komplexität (d.h. weniger Unbekannte) aufweist, eine Annäherung dieser Startlösung zu ermitteln und diese dann auf die feineren Gitter zu übertragen.

Dieser Ansatz führt uns zum sogenannten „geschachtelten“ Mehrgitter. Hierbei wird üblicherweise nicht nur eine Startlösung auf dem Grobgitter ermittelt und anschließend auf das Feingitter prolongiert, sondern pro Gitterlevel jeweils ein V-Zyklus ausgeführt, um diese weiter zu verbessern (vgl. Abbildung 2.3). Dies führt dazu, dass in der Regel bereits nach einer Iteration dieses „geschachtelten“ Verfahrens die Näherungslösung hinreichend nah, d.h. bei nichtlinearen Gleichungen und einem Newton Verfahren im Einzugsbereich der gesuchten Lösung, liegt (vgl. [3]). Durch nun folgende V-, W-, oder F-Zyklen kann die Genauigkeit der Lösung auch hier weiter verbessert werden.

Dieses Verfahren ist insbesondere für nichtlineare Probleme gut geeignet, da z.B. das Newton Verfahren, welches zur Linearisierung benutzt werden kann, erst für eine Näherungslösung in einen gewissen Einzugsbereich eine super-lineare und damit optimale Konvergenz garantiert.

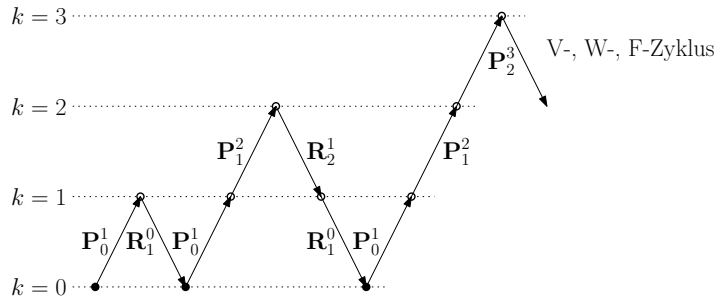


Abbildung 2.3: Grafische Darstellung des „geschachtelten“ Mehrgitters.

Auf die Verbesserung der Konvergenzeigenschaften durch Anwendung eines solchen „geschachtelten“ Mehrgitterverfahrens werden wir in den numerischen Tests genauer eingehen.

Kapitel 3 Grundlegende Strukturen und Methoden

Nun, wo wir das Prinzip des Mehrgitterverfahrens und die üblicherweise benutzten Zyklustypen vorgestellt haben, wollen wir uns etwas näher mit den einzelnen Komponenten des Algorithmus befassen.
Wir werden dabei nicht ins Detail gehen, sondern nur entscheidende Merkmale, zum Teil an Beispielen, erläutern und gegebenenfalls auf weitere Literatur verweisen. Grundlegend werden wir, der Einfachheit halber, alle Techniken auf Basis eines konformen Finite Elemente Ansatzes vorstellen.

Im Folgenden verwenden wir die Bezeichnungen

k	-	Gitterlevel mit $k \in \{0, \dots, L\}$, $0 \mathrel{\widehat{=}}$ Grobgitter, $L \mathrel{\widehat{=}}$ Feingitter
$\mathcal{V}_k$	-	Finite Elemente Funktionenraum auf Gitterlevel k
u, v, r, f	-	Funktionen (optional mittels Index an Gitterlevel gebunden)

und werden eine von einer Funktion u abhängige Matrix A durch $A[u]$ darstellen.

3.1 Gitter

Wie bereits erwähnt beschränken wir uns in unserer Vorstellung und Analyse auf den geometrischen Mehrgitteransatz. Daher sind einige Voraussetzungen an die Struktur der Gitterhierarchie zu machen, um eine Umsetzung des Verfahrens zu ermöglichen.

3.1.1 Gitterhierarchie

Grundlegende Voraussetzung für einen geometrischen Mehrgitteralgorithmus ist eine Struktur aus verschiedenen Gittern, die in fortlaufender Weise aus weniger Gitterpunkten und demzufolge weniger Unbekannten bestehen. Diese Struktur entspricht im einfachsten Fall einem beliebigen Basisgitter, welches elementweise durch regelmäßiges Unterteilen der Zellen (in 2D entspricht dies bei Viereckgittern zum Beispiel einer Vierteilung) verfeinert wird.

Diese so erzeugten Gitter wollen wir im Folgenden mittels einer einfachen Nummerierung benennen. Dabei beginnen wir mit dem Grobgitter bei $k = 0$ und nummerieren fortlaufend bis zum feinsten Gitterlevel $k = L$, d.h. dass die Gitterhierarchie aus $L + 1$ Gittern besteht.

In unseren später beschriebenen Tests werden wir immer ausgehend vom Einheitsquadrat $[0, 1]^2$ mit einer Grundverfeinerung diese Gitterhierarchie durch Viertelungen der Zellen erzeugen.

Im Folgenden wollen wir anhand eines einfachen Beispiels das Vorgehen der Verfeinerung eines Basisgitters demonstrieren, sowie es auch in den durchgeführten Tests verwendet wird. Mit Hilfe einer iterativen Anwendung dieses beispielhaften Prozesses lassen sich dann beliebig komplexe Gitterhierarchien erzeugen.

Beispiel. *Beispielhafte Verfeinerung eines Basisgitters.*

Ausgehen wollen wir von einem Basisgitter mit vier Zellen und einer zeilenweise Knotennummerierung. Die Zellen sind ebenfalls zeilenweise benannt und werden im Weiteren durch eingekreiste

Ziffern nummeriert (vgl. Abbildung 3.1).

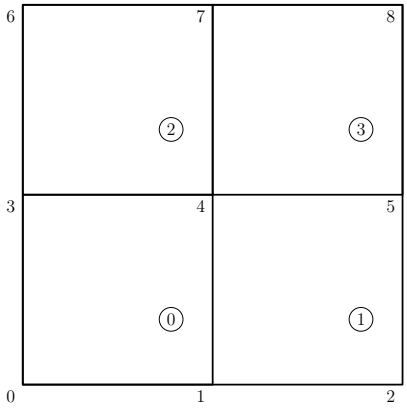


Abbildung 3.1: Basisgitter mit zeilenweiser Nummerierung.

Bevor ein solches Gitter verfeinert werden kann, müssen zuerst neue Punkte generiert werden, die im Anschluss daran als neue Knoten der Zellen genutzt werden können. Dafür wird in ansteigender Reihenfolge über die existierenden Zellen iteriert und innerhalb einer jeden Zelle gegen den Uhrzeigersinn, ausgehend von der linken unteren Ecke, ein neuer Knoten erzeugt, sofern dies nicht bereits beim Durchlaufen einer vorherigen Zelle geschehen ist.

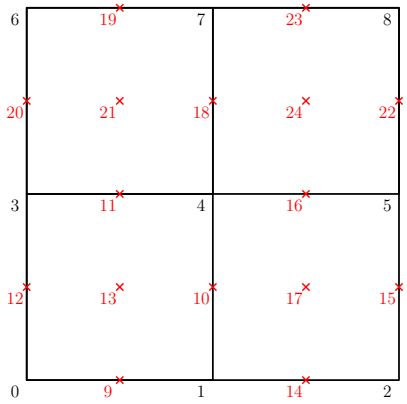


Abbildung 3.2: Neue Punkte des Basisgitters für Gitterverfeinerung.

Nach der Erzeugung der notwendigen Knoten (vgl. Abbildung 3.2) müssen nun noch die Zellen der neuen Gitterstufe angelegt werden. Da die Zellen der Struktur fortlaufend generiert werden, setzt die Nummerierung nun nach der letzten Zellennummer des Basisgitters fort. Dabei wird erneut in aufsteigender Folge über die alten Zellen iteriert und die neuen Zellen werden Zelle für Zelle gegen den Uhrzeigersinn orientiert angelegt (vgl. Abbildung 3.3). Dieser Prozess der Gitterverfeinerung wird im Allgemeinen auch 2-Level-Numbering genannt.

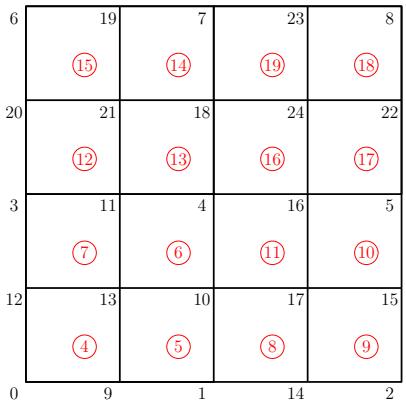


Abbildung 3.3: Einmal verfeinertes Basisgitter mit neuen Zellen.

Ein zu erwähnender Vorteil dieser Nummerierung ist, dass die Knotennummern des alten Gitters übernommen werden. Dies erleichtert es eine Lösung von einem neuen Gitter auf ein altes zu transferieren. Allerdings geht durch diese rekursive Nummerierung jegliche Bandstruktur der Systemmatrix verloren.

3.1.2 Gittertransfer

Essentiell für alle Mehrgittermethoden ist der Transfer von Korrekturwerten und Residuen (bzw. Defekten) zwischen unterschiedlichen Gitterleveln. Dabei ist zu beachten, dass ein passender Transferoperator zu wählen ist, um mit den Verfahren die gewünschten Konvergenzraten erzielen zu können.
Von den vielen Varianten, die für diese Operation existieren, wählen wir eine dem Finite Element-Ansatz angepasste makroweise Interpolation für die Prolongation (vgl. [1, S. 57 ff.]). Die Restriktion ergibt sich analog dazu als adjungierter Operator.

Die Prolongationsoperation

$$\begin{aligned} \mathbf{P}_{k-1}^{k} &: \mathcal{V}_{k-1} \longrightarrow \mathcal{V}_k \\ \mathbf{P}_{k-1}^{k} v_{k-1} &= v_k \end{aligned}$$

entspricht einem Transfer von Gitterlevel $k-1$ auf k.
Dieser Operator wird benutzt, um Lösungen als auch Korrekturwerte von einem gröberen auf ein feineres Gitter zu prolongieren. Es handelt dabei - in unserem Fall der konformen Finite Elemente -, um die Inklusion und somit die L_2-Projektion aus dem Raum $\mathcal{V}_{k-1}$ auf $\mathcal{V}_k$ (vgl. [1, S. 58]).

Die Konstruktion dieses Interpolationsoperators wollen wir anhand von biquadratischen Finiten Elementen später beispielhaft darstellen. Für den Fall der bilinearen Finiten Elemente findet sich eine Darstellung in [1, S. 60]

Wie bereits erwähnt handelt es sich beim Restriktionsoperator $\mathbf{R}$ um den zur Prolongation adjungierten Operator. Dabei ist die Adjungiertheit hier durch

$$\left(\mathbf{R}_{k}^{k-1} u_k, v_{k-1}\right)_{k-1} = \left(u_k, \mathbf{P}_{k-1}^{k} v_{k-1}\right)_{k} \qquad \forall u_k \in \mathcal{V}_k \text{ und } \forall v_{k-1} \in \mathcal{V}_{k-1}$$

definiert, wobei $(\cdot,\cdot)_k$ dem Skalarprodukt auf Gitterlevel k entspricht.
Somit erhalten wir bei konformen Finiten Elementen und unserer Wahl der Prolongation den Restriktionsoperator als

$$\begin{aligned}\mathbf{R}_k^{k-1} &: \mathcal{V}_k \longrightarrow \mathcal{V}_{k-1} \\ \mathbf{R}_k^{k-1} r_k &= \left[\mathbf{P}_{k-1}^k\right]^T r_k = r_{k-1}.\end{aligned}$$

Im Falle des nichtlinearen Mehrgitters müssen neben Residuen auch Lösungsapproximationen von Fein- auf Grobgitter transferiert werden. Hierbei handelt es sich auf den Gitterleveln um diskrete Funktionen, daher bietet es sich an, den Transfer mittels einer natürlichen Einbettung durchzuführen. Diese wird durch den Operator $\mathbf{I}$ gekennzeichnet und entspricht bei konformen Finiten Elementen anschaulich der Übernahme der Knotenwerte in den gemeinsammen Knotenpunkten.

$$\begin{aligned}\mathbf{I}_k^{k-1} &: \mathcal{V}_k \longrightarrow \mathcal{V}_{k-1} \\ \mathbf{I}_k^{k-1} u_k &= u_{k-1}\end{aligned}$$

Beispiel. *Konstruktion von Transfermatrizen für biquadratische Finite Elemente*

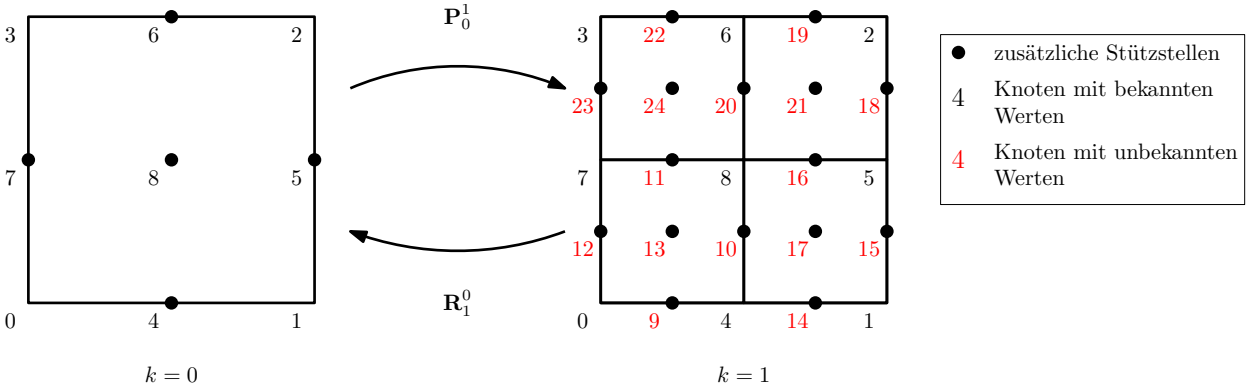


Abbildung 3.4: Gittertransfer zwischen Basisgitter $k=0$ und verfeinertem Gitter $k=1$.

Ausgehend von den Gittern in Abbildung 3.4 konstruieren wir mittels einer makroweisen Interpolation passender Ordnung - d.h. Basisfunktionen sollen exakt integriert werden - die Prolongationsmatrix. Mit diesem Ansatz lässt sich auf Basis der Wertemenge $W_{alt} := \{v_0,\dots,v_8\}$ der bekannten Knoten eine Vorschrift für die Bestimmung der Wertemenge $W_{neu} := \{w_0,\dots,w_{24}\}$ der neuen Knoten des verfeinerten Gitters bestimmen.

$$\begin{aligned} w_k &= v_k && k \in \{0,\dots,8\} \\ w_k &= \sum_{i=0}^{8} c_{ki} v_i, && k \in \{9,\dots,24\}\end{aligned}$$

Die Wahl der Koeffizienten c_{ki} ergibt sich hierbei unter Verwendung einer biquadratischen Interpolationsformel auf der Referenzzelle $[-1,1]^2$ und unter Ausnutzung der Kronecker-Delta-Eigenschaft der Basisfunktion der verwendeten konformen Finiten Elementen und ist in Tabelle 3.1 dargestellt. Ein Transfer auf beliebigen Zellen kann dann mit Hilfe der üblichen Transformation auf das Referenzelement durchgeführt werden.
Diese Interpolation lässt sich mit Hilfe von einer einfachen Matrix-Vektor Multiplikation durchführen.

c_{ki}		i								
		0	1	2	3	4	5	6	7	8
k	9	$\frac{3}{8}$	$-\frac{1}{8}$			$\frac{3}{4}$				
	10					$\frac{3}{8}$		$-\frac{1}{8}$		$\frac{3}{4}$
	11						$-\frac{1}{8}$		$\frac{3}{8}$	$\frac{3}{4}$
	12	$\frac{3}{8}$			$-\frac{1}{8}$				$\frac{3}{4}$	
	13	$\frac{9}{64}$	$-\frac{3}{64}$	$\frac{1}{64}$	$-\frac{3}{64}$	$\frac{9}{32}$	$-\frac{3}{32}$	$-\frac{3}{32}$	$\frac{9}{32}$	$\frac{9}{16}$
	14	$-\frac{1}{8}$	$\frac{3}{8}$			$\frac{3}{4}$				
	15		$\frac{3}{8}$	$-\frac{1}{8}$			$\frac{3}{4}$			
	16						$\frac{3}{8}$		$-\frac{1}{8}$	$\frac{3}{4}$
	17	$-\frac{3}{64}$	$\frac{9}{64}$	$-\frac{3}{64}$	$\frac{1}{64}$	$\frac{9}{32}$	$\frac{9}{32}$	$-\frac{3}{32}$	$-\frac{3}{32}$	$\frac{9}{16}$
	18		$-\frac{1}{8}$	$\frac{3}{8}$			$\frac{3}{4}$			
	19			$\frac{3}{8}$	$-\frac{1}{8}$			$\frac{3}{4}$		
	20					$-\frac{1}{8}$		$\frac{3}{8}$		$\frac{3}{4}$
	21	$\frac{1}{64}$	$-\frac{3}{64}$	$\frac{9}{64}$	$-\frac{3}{64}$	$-\frac{3}{32}$	$\frac{9}{32}$	$\frac{9}{32}$	$-\frac{3}{32}$	$\frac{9}{16}$
	22			$-\frac{1}{8}$	$\frac{3}{8}$			$\frac{3}{4}$		
	23	$-\frac{1}{8}$			$\frac{3}{8}$				$\frac{3}{4}$	
	24	$-\frac{3}{64}$	$\frac{1}{64}$	$-\frac{3}{64}$	$\frac{9}{64}$	$-\frac{3}{32}$	$-\frac{3}{32}$	$\frac{9}{32}$	$\frac{9}{32}$	$\frac{9}{16}$

Tabelle 3.1: Prolongationkoeffizienten für biquadratische Finite Elemente auf dem Referenzelement $[-1,1]^2$.

Ebenso lässt sich die Restriktion darstellen. Bei dieser handelt es sich in dieser Situation, wie bereits angesprochen, einfach um die Transponierte der Prolongationsmatrix.

Bei der Restriktion werden anschaulich die Werte von mehreren Knoten auf einige wenige verteilt. Diese Eigenschaft erfüllt bei der Restriktion von Residuen ihren Zweck. Beim nichtlinearen Mehrgitterverfahren müssen wir jedoch auch Lösungen restringieren. Diese würden durch den von uns definierten Restriktionsoperator verfälscht. Als einfachste und auch anschaulich verständlichste Wahl ergibt sich für diesen Transfer eine einfache Übernahme der gemeinsamen Knotenwerte.

Im Allgemeinen existieren sehr viele, auch komplexere, Möglichkeiten der Prolongation und Restriktion. Diese müssen jedoch einigen Bedingungen genügen.
Eine notwendige Bedingung dabei ist, dass

$$m_{\mathbf{P}} + m_{\mathbf{R}} > 2m$$

ist (vgl. [9, S. 71]). Hierbei sind $m_{\mathbf{P}}$ und $m_{\mathbf{R}}$ die Ordnungen der Interpolation der Transferoperatoren. Ist es möglich mit dem Operator $\mathbf{P}$ Funktionen der Ordnung q exakt zu interpolieren, so ist $m_{\mathbf{P}} = q+1$. Bei der Restriktion muss stattdessen der Operator $\mathbf{R}^T$ betrachtet werden, der in unserem Fall der konformen Finiten Elemente, wie bereits erwähnt, mit dem Operator $\mathbf{P}$ übereinstimmt, so dass $m_{\mathbf{P}} = m_{\mathbf{R}}$ gilt. Der Term $2m$ der rechten Seite entspricht der Ordnung der betrachteten partiellen Differentialgleichung.

3.2 Fehlerdämpfung

Wie bereits erwähnt ist eine zweite entscheidende Komponente im Mehrgitterverfahren der Glätter, welcher dafür sorgt, dass hochfrequente Fehleranteile gedämpft werden, um den Defektverlauf auf dem Grobgitter besser wiedergeben zu können. Essentiell ist dabei vor allem die sogenannte Glättungseigenschaft (vgl. [5, S. 116])

Gegeben sei dazu ein diskretes lineares oder nichtlineares Gleichungssystem einer partiellen Differentialgleichung in der Form

$$A[u]u = f$$

mit einer Startlösung $u^{(0)}$. Diese kann bei zeitabhängigen Problemen zum Beispiel der Lösung zum letzten Zeitschritt entsprechen oder aber allgemein die Nulllösung $u^{(0)} \equiv 0$ sein.

Im Folgenden wollen wir die Anwendung eines geeigneten Glätters auf dieses System wie folgt darstellen

$$u^{(m)} = \mathcal{S}\left(u^{(m-1)}, f\right), \qquad m \in \mathbb{N}.$$

Hierbei beschreibt der Index m die $m-$te Approximation der Lösung des Systems.

3.2.1 Einfache Glätter

Bei einfachen Glättern handelt es sich um gedämpfte Defektkorrekturverfahren, die entsprechenden Eigenschaften genügen. Dabei ist es wichtig zu erwähnen, dass nur eine geringe Anzahl von Iterationsschritten ausgeführt wird (oft nicht mehr als drei).

$$\mathcal{S}\left(u^{(m-1)}, f\right) := u^{(m-1)} + \omega M^{-1} r, \qquad m \in \mathbb{N} \tag{3.1}$$

Hierbei stellt $r = f - A[u^{(m-1)}]u^{(m-1)}$ das Residuum des Gleichungssystems zur aktuellen Näherungslösung $u^{(m-1)}$ dar und f entspricht der rechten Seite des Systems. Der Vorkonditionierer M der Richardson-Iteration (3.1) muss hier der entscheidenden Glättungseigenschaft genügen, die zur Fehlerdämpfung führt.

Übliche einfache Glätter sind gedämpfte Jacobi- oder Gauss-Seidel-Verfahren. Diese haben die Eigenschaft, dass sie die hochfrequenten Fehleranteile dämpfen und die niederfrequenten fast unberührt lassen. Dies führt bereits nach wenigen Anwendungen zu einem geglätteten Defekt, wie in der unten stehenden Abbildung 3.5 anschaulich dargestellt ist.
Die Wahl des Dämpfungsparameters ω ist jedoch im Allgemeinen nicht trivial und beeinflusst die Konvergenzgeschwindigkeit des gesamten Mehrgitterverfahrens entscheidend (für numerische Vergleiche siehe u.a. [1, S. 99 ff.]). Für ein isotropes Poissonproblem mit Jacobiglättung hat sich zum Beispiel $0.6 \leq \omega \leq 0.8$ als guter Ausgangswert zur Bestimmung eines optimalen Dämpfungsparameters erwiesen. Im Gegensatz dazu muss bei einem Gauss-Seidel-Glätter in der Regel keine Dämpfung erfolgen.

Allgemein gibt es viele Möglichkeiten für den Glättungsoperator, jedoch ist ein guter iterativer Löser nicht unbedingt ein guter Glätter. Den entscheidenden Faktor stellt dabei die Dämpfung der

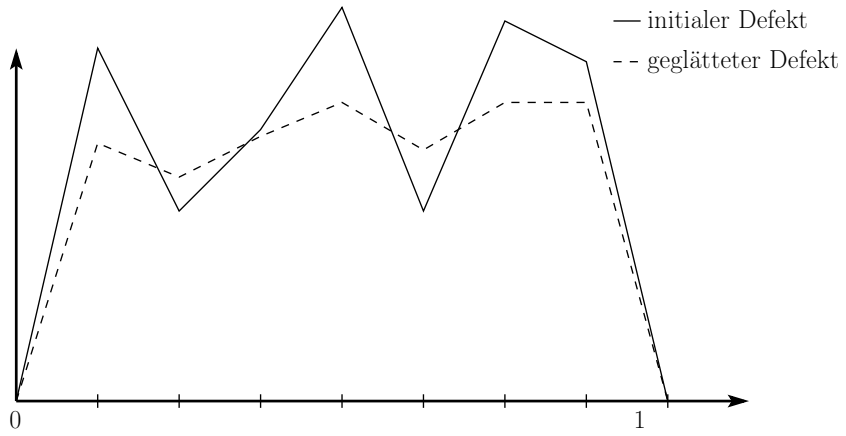


Abbildung 3.5: Modellhafter Defektverlauf vor und nach einer Glättung.

hochfrequenten Fehleranteile dar. Ein klassisches CG-Verfahren ohne Vorkonditionierer, welches im Allgemeinen schneller konvergiert als ein Jacobi-Verfahren, ist beispielsweise kein gut geeigneter Glätter.

Wir wollen aber dennoch eine Möglichkeit vorstellen, wie man andere iterativen Löser als Glättungsoperatoren verwenden kann. Insbesondere möchten wir dies für Krylowraum-Verfahren erläutern.

3.2.2 Krylowraum Glätter

Alternativ zu den gedämpften Defektkorrektur-Verfahren existieren Variationen, bei denen Krylowraum-Löser in Kombination mit einer geeigneten Vorkonditionierung verwendet werden (vgl. [1]). Hierbei wird die Wahl des optimalen Dämpfungsparameters ω durch den iterativen Löser überflüssig, da die Schritte des äußeren Krylowraum-Lösers wie eine adaptive Steuerung dieses Dämpfungsparameters wirken. Der gewählte Vorkonditionierer bewirkt dann die übliche erwünschte Dämpfung der Fehlerfrequenzen.
Dadurch werden auch klassische iterative Krylowraum-Löser, wie das CG oder BiCGStab-Verfahren, zu geeigneten Glättern. Im optimalen Fall erhöhen sie die Konvergenzgeschwindigkeit, da sie systemunabhängig in jedem Iterationsschritt einen geeigneten Dämpfungsparameter wählen. Wichtig dabei ist anzumerken, dass wir den iterativen Löser ebenfalls nicht bis zu Konvergenz iterieren lassen, sondern nur eine feste Anzahl an Schritten, ähnlich zum einfachen Glättungsoperator, durchführen.

In den hier vorgestellten numerischen Tests verwenden wir für die linearen Mehrgitteralgorithmen, sofern keine anderen Angaben erfolgen, ein BiCGStab-Verfahren mit Jacobi-Vorkonditionierer als Glätter. Dabei führt das BiCGStab-Verfahren pro Iteration zwei Matrix-Vektor-Multiplikationen durch, so dass wir einen Dämpfungsschritt dieses Verfahrens mit zwei Dämpfungsschritten der klassischen Jacobivarianten vergleichen können.

Innerhalb unserer numerischen Tests werden wir die Wahl des Glättungsoperators nur geringfügig variieren.

3.3 Techniken zur Linearisierung

Die in dieser Arbeit vorgestellten Tests sollen beim Lösen von nichtlinearen Problemen einen Vergleich von Mehrgitterverfahren ermöglichen. Dabei wird unter anderem eine auf dem klassischen linearen Mehrgitterverfahren basierende Lösungsstrategie für das nichtlineare Problem verwendet. Einen entscheidenden Faktor stellt dabei die Linearisierung von Teilproblemen dar, welche dann mit Hilfe des klassischen linearen Mehrgitteransatzes gelöst werden können. Das nichtlineare Problem an sich wird dann über einen iterativen Verbesserungsprozess mit den Lösungen der linearisierten Probleme gelöst.

Gegeben sei dazu ein Gleichungssystem der Form

$$N(u) = 0,$$

mit der Lösung u und dem unter Umständen auch nichtlinear davon abhängigen Operator N.

Auf Basis einer geeigneten Startlösung $u^{(0)}$ lässt sich dann die Lösungsapproximation allgemein durch folgende Iterationsvorschrift verbessern.

$$u^{(m)} = u^{(m-1)} - C^{-1}N(u^{(m-1)}) \qquad m = 1, \ldots \tag{3.2}$$

Da der Operator C jedoch in der Regel nicht einfach zu invertieren ist und selten bereits in inverser Form vorliegt, bedienen wir uns hier eines Tricks. Wir stellen Gleichung (3.2) nach der Korrektur $\triangle u := u^{(m)} - u^{(m-1)}$ um, so dass

$$C\triangle u = -N(u^{(m-1)})$$

zu lösen ist. Zur Lösung dieses Gleichungssystems können wir uns nun üblicher iterativer Löser bedienen.

Wir wollen im Folgenden insbesondere zwei Möglichkeiten zur Wahl des Operators C vorstellen und im Anschluss, innerhalb der numerischen Tests, analysieren wie diese Techniken zusammen mit den verschiedenen Mehrgitterverfahren die Eigenschaften des gesamten Lösungsalgorithmus verändern.

3.3.1 Newton Verfahren

Die erste verwendete Variante stellt das klassische Newton Verfahren dar. Hierbei wird mit Hilfe der Jacobimatrix des Systems ein linearisiertes Problem aufgestellt, welches mittels eines einfachen linearen Lösers gelöst werden kann, um eine verbesserte Lösung zu berechnen.

Dafür wählen wir in der Iterationsvorschrift (3.2) für den Operator C die Jacobimatrix $dN[u^{(m-1)}]$ des Systems zur aktuellen Lösungsapproximation $u^{(m-1)}$.

Für den Fall, dass sich der Operator $N(u)$ durch eine unter Umständen von der Lösung u abhängige Systemmatrix $A[u]$, sowie einem lösungsunabhängigen Teil f darstellen lässt,

$$N(u) = A[u]u - f \tag{3.3}$$

entspricht die Jacobimatrix im Falle von linearen Systemen eben der Systemmatrix A oder kann im

nichtlinearen Fall durch Informationen über die zugrunde liegende Gleichung analytisch ermittelt oder näherungsweise z.B. durch einfache Differenzenschemata approximiert werden (vgl. [10]).

Die Lösung eines nichtlinearen Systems kann also schrittweise durch mehrfache Anwendung dieses Prinzips optimiert werden. Dieses Verfahren garantiert dabei für hinreichend gute Lösungen eine asymptotische Konvergenz der Ordnung zwei.

3.3.2 Fixpunkt-Defektkorrektur Verfahren

Beim Newton Verfahren ist es notwendig, die zu dem System gehörende Jacobimatrix zu bestimmen bzw. zumindest in hinreichender Form zu approximieren. Die exakte Berechnung und Assemblierung eben dieser ist jedoch im nichtlinearen Fall durchaus mit einem enormen numerischen Aufwand verbunden. Daher bedient man sich häufig nur einfacher Approximationen oder iterativer Updates der Jacobimatrix, wie der Methode nach Broyden (vgl. [11]), welche zu sogenannten Quasi-Newton Verfahren führen.

Für den Fall, dass sich die Ableitungen des Systems jedoch nicht bestimmen lassen, wollen wir eine alternative Möglichkeit beschreiben.

Ausgehend vom iterativen Ansatz (3.2) wählen wir für den Operator C nicht die Jacobimatrix dN, sondern eine andere geeignete Matrix. Dabei kann es sich sowohl um eine Approximation der Jacobimatrix handeln, was zu einem bereits erwähnten Quasi-Newton Verfahren führt, oder aber um eine andere, gewissen Bedingungen genügende, Matrix.

Unter den Bedingungen wie in (3.3) ergibt sich als eine mögliche Wahl für den Operator die Systemmatrix $C = A[u]$. Dies führt zu folgendem iterativen Fixpunkt-Defektkorrektur Verfahren:

$$u^{(m)} = u^{(m-1)} - A[u^{(m-1)}]^{-1} N(u^{(m-1)}) \qquad m = 1, \dots$$

Dieses Verfahren wollen wir innerhalb der numerischen Tests mit dem klassischen Newton Verfahren vergleichen und überprüfen, wie es sich auf die Konvergenz der verschiedenen Mehrgitterverfahren auswirkt.

Kapitel 4 Mehrgitterverfahren

Im Weiteren werden wir nun die später untersuchten Mehrgitterverfahren beschreiben. Dafür betrachten wir eine partielle Differentialgleichung in folgender Darstellung:

$$\begin{cases} \mathcal{L}u = f & \text{in } \Omega \\ u = g_D & \text{auf } \Gamma = \partial\Omega, \end{cases}$$

wobei $\mathcal{L}$ einem nicht unbedingt linearen Differentialoperator entspricht. Außerdem beschränken wir uns auf Dirichletrandbedingungen. Durch übliche Finite Element-Diskretisierungen (vgl. z.B. [6, S. 73 ff.]) ergibt sich daraus, unter Berücksichtigung der Randbedingungen, ein diskretes Gleichungssystem:

$$A_h[u_h]u_h = f_h. \tag{4.1}$$

Im Folgenden werden wir nur diskrete Gleichungssysteme und damit diskrete Lösungen betrachten, so dass wir den Diskretisierungsindex h vernachlässigen können.

Damit ergeben sich während der weiteren Betrachtung u.a. die folgenden Bezeichnungen.

- A_k - Diskretes Gleichungssystem eines Differentialoperators auf Gitterlevel k
- f_k - diskrete rechte Seite des Systems auf Gitterlevel k
- u_k - Lösung bzw. Approximation des diskreten Gleichungssystems auf Level k
- v_k - Korrekturvektor auf Level k
- r_k - Residuum auf Level k

Ebenfalls werden wir bei nichtlinearen Problemen die Anwendung einer lösungsabhängigen Matrix $A[u]$ auf die Lösung u vereinfacht durch

$$A\langle u\rangle := A[u]u$$

darstellen.

Für die Herleitung und den Vergleich der Mehrgitterverfahren wollen wir uns auf den in **Kapitel 2** beschriebenen V-Zyklus beschränken und nutzen außerdem die in **Kapitel 3** beschriebenen Transfer- und Glättungsoperatoren.

4.1 Klassischer Mehrgitteransatz

Nun wollen wir den klassischen linearen Mehrgitteransatz genauer erläutern und neben der Umsetzung, sowohl einen formellen Algorithmus angeben, als auch eine Übertragung dieses Ansatzes auf nichtlineare Probleme beschreiben. Dieser Abschnitt orientiert sich dabei insbesondere an den Veröffentlichungen von Henson [3] und Hackbusch [5].

Sei dazu der Differentialoperator $\mathcal{L}$ zunächst ein linearer Operator und somit (4.1) ein lineares Gleichungssystem.

4.1.1 Umsetzung

Wir wollen uns zur Herleitung der Funktion des einfachen Mehrgitteralgorithmus hier anfänglich auf einen auf zwei Gitterleveln arbeitenden Algorithmus (auch Zwei-Gitter-Verfahren genannt) beschränken, um später eine allgemeine Darstellung des Prozesses mit beliebiger Anzahl an Gitterleveln angeben und verstehen zu können.
Dementsprechend sei bei $k = 1$ das Feingitter und $k = 0$ das Grobgitter. Des Weiteren sollen diese beiden Gitter den Eigenschaften aus **Kapitel 3.1** genügen, so dass wir die Lösung des linearen Gleichungssystems

$$A_1 u_1 = f_1$$

suchen.
Ausgehend von einer geeigneten Startlösung $u_1^{(0)}$, welche den Dirichletrandbedingungen der Differentialgleichung genügt, wird durch mehrfaches Anwenden des Glätters eine verbesserte Approximation der Lösung

$$u_1^{(\nu)} = \mathcal{S}^{\nu}\left(u_1^{(0)}, f_1\right)$$

ermittelt. S^{ν} entspricht hierbei der ν-fachen Anwendung des Glätters.
Auf Basis dieser Näherungslösung wird nun das Residuum bzw. der Defekt des Systems gebildet und auf das Grobgitter restringiert.

$$\begin{aligned} r_1 &= f_1 - A_1 u_1^{(\nu)} \\ r_0 &= \mathbf{R}_1^0 r_1 \end{aligned}$$

Auf diesem Level wird nun das Defektproblem

$$A_0 v_0 = r_0 \tag{4.2}$$

mit ausreichender Genauigkeit gelöst. Der Vektor v_0 entspricht dabei einer zu ermittelnden Korrektur der aktuellen Lösungsapproximation.

Dieser Zusammenhang kann unter Ausnutzung der Linearität des diskreten Operators A aus der Gleichung

$$\begin{aligned} & A_1(u_1^{(\nu)} + v_1) = f_1 \\ \Leftrightarrow \qquad & A_1(v_1) = \underbrace{f_1 - A_1 u_1^{(\nu)}}_{:=r_1}, \end{aligned}$$

wobei $u_1^{(\nu)} + v_1$ der Korrektur der Näherungslösung zur exakten Lösung entspricht, und entsprechendem Transfer auf das Grobgitter

$$A_0(\underbrace{\mathbf{I}_1^0 v_1}_{=v_0}) = \underbrace{\mathbf{R}_1^0 r_1}_{=r_0}$$

gewonnen werden. Eine geeignete Startlösung für einen iterativen Lösungsprozess stellt dabei $v_0 \equiv 0$ dar.

Insbesondere muss sichergestellt werden, dass in Punkten des Dirichlet-Randes keine Korrektur erfolgt.

Der auf diesem Gitterlevel nun ausreichend exakt berechnete Korrekturvektor v_0 wird durch Prolongation wieder auf das Feingitter transportiert und zur Verbesserung der Lösungsapproximation genutzt

$$u_1^{(\nu+1)} = u_1^{(\nu)} + \mathbf{P}_0^1 v_0.$$

Anschließendes μ-faches Nachglätten stellt sicher, dass die hochfrequenten Fehleranteile, die eventuell durch die Prolongation erneut entstanden sind, gedämpft und ans Feingitter angepasst werden

$$u_1^{(\nu+\mu+1)} = \mathcal{S}^\mu \left(u_1^{(\nu+1)}, r_1 \right).$$

Mit der nun ermittelten Lösungsapproximation $u_1^{(\nu+\mu+1)}$ als Startlösung wird der Prozess solange fortgesetzt, bis die Lösung dem geforderten Abbruchkriterium, beispielsweise einem hinreichend kleinen Residuum, genügt.

Für den Fall, dass die exakte Lösung nach der Vorglättung erzielt wurde, d.h. $f_1 - A_1 u_1^{(\nu)} = 0$, gilt für den Korrekturwert v_0 des Grobgitterproblems

$$\begin{aligned} v_0 &= A_0^{-1} r_0 \\ &= A_0^{-1} \mathbf{R}_1^0 r_1 \\ &= A_0^{-1} \left(\mathbf{R}_1^0 \left(f_1 - A_1 u_1^{(\nu)} \right) \right) \\ &= 0 \end{aligned}$$

und somit erfolgt für diesen Fall keine weitere Korrektur der Feingitterlösung. Dies begründet auch die Wahl unserer Startlösung beim Defektproblem (4.2).

Dieser Zweigitteralgorithmus lässt sich nun ohne große Probleme auf Verfahren mit weiteren Gittern ausdehnen. Dabei wird das Defektproblem, welches auf einem gröberen Gitter zu lösen ist, durch erneutes Aufrufen dieses Mehrgitterzyklus approximativ gelöst. Dieser rekursive Aufruf entspricht einer Störung des berechneten Korrekturwertes und führt nur zu einer minimalen Verringerung der Konvergenzrate (vergleiche [5, S. 160 ff.]). Jedoch sinkt der Aufwand bei der Berechnung der „exakten“ Lösung auf dem gröbsten Gitter.

Dieser rekursive Aufruf wird solange durchgeführt, bis die Problemgröße klein genug ist (d.h. ein Gitter mit ausreichend wenig Unbekannten erreicht wurde), um ohne großen numerischen Aufwand das Problem möglichst exakt lösen zu können. Dieses Prinzip führt dann zum bereits vorgestellten V-Zyklus. Durch mehrfaches Aufrufen des inneren Mehrgitteralgorithmus zur Verbesserung der Näherung des Korrekturwertes entsteht dann u.a. der W-Zyklus.

4.1.2 Algorithmus

Ausgehend von einem linearen Feingitterproblem

$$A_L u_L = f_L$$

und einer geeigneten Startlösung $u_L^{(0)}$, sowie einer Hierarchie von Systemmatrizen $\mathcal{A} = \{A_0, \ldots, A_L\}$ auf den Gitterleveln rufen wir **Algorithmus 4.1** solange auf, bis ein Abbruchkriterium erreicht wird.

$$u_L^{(m)} = \mathrm{MG}(L, \mathcal{A}, f_L, u_L^{(m-1)}) \qquad m = 1, \ldots \tag{4.3}$$

Der Aufbau der Systemmatrizen $A_k \in \mathcal{A}$ mit $k = 0, \ldots, L$ erfolgt dabei pro Gitterlevel über elementweise Assemblierung der lokalen Systemmatrizen der zugrundeliegenden partiellen Differentialgleichung.

Algorithmus 4.1: Klassischer V-Zyklus

Aufruf :
`MG(`$k, \mathcal{A}, b, u^{(0)}$`)`
Parameter :
ν - Anzahl der Vorglättungsschritte
μ - Anzahl der Nachglättungsschritte
Eingabe :
k - Gitterlevel
$\mathcal{A}$ - Menge der Systemmatrizen $\mathcal{A} = \{A_0, \ldots, A_k\}$
b - rechte Seite auf Level k
$u^{(0)}$ - initiale Lösungsapproximation auf Level k
Ausgabe :
u - Lösungsapproximation des Systems $A_k u = b$

```
if k = 0 then
    u = A_0^{-1} b                                   // Lösen des Grobgitterproblems
else
    u^(ν) = S^ν (u^(0), b)                           // ν-faches Vorglätten
    r_k = b − A_k u^(ν)                              // Berechnung des Residuums
    r_{k−1} = R_k^{k−1} r_k                          // Restriktion des Residuums
    v^(0) ≡ 0                                        // Initialisierung Korrekturwert
    v = MG(k − 1, A, r_{k−1}, v^(0))                 // rekursives Lösen des Grobgitterdefektproblems
    u^(ν+1) = u^(ν) + P_{k−1}^k v                    // Grobgitterkorrektur
    u = S^μ (u^(ν+1), b)                             // μ-faches Nachglätten
end
```

Für den hier beschriebenen linearen Fall existieren für den Algorithmus mit konformen bilinearen und biquadratischen Finiten Elementen viele bewiesene numerische Aussagen über das Konvergenzverhalten bei Veränderung der einzelnen Parameter, wie z.B. die Anzahl der Vor- und Nachglättungsschritte. Entsprechende Aussagen und Beweise finden sich unter anderem in [1] und [5].

4.1.3 Übertragung auf nichtlineare Probleme

Für den Fall eines nichtlinearen Operators $\mathcal{L}$ lässt sich der Ansatz des klassischen Mehrgitterverfahrens modifizieren. Dazu wird mit Hilfe einer aus **Kapitel 3.3** vorgestellten Linearisierungstechnik eine Reihe von linearen Teilproblemen aufgestellt, die mit dem klassischen Ansatz gelöst werden können.

Durch Anwendung dieser Techniken erhalten wir ein Gleichungssystem der Form

$$C \triangle u = f - A\left\langle u^{(m-1)} \right\rangle \tag{4.4}$$

zu einer Lösungsapproximation $u^{(m-1)}$, dass zur Bestimmung einer Lösungsverbesserung verwendet werden kann. Dieses Gleichungssystem (4.4) ist linear und kann somit mittels des klassischen linearen Mehrgitteralgorithmus gelöst werden. Anschließend kann durch $\triangle u$ mit

$$u^{(m)} = u^{(m-1)} + \triangle u \qquad m = 1, \ldots$$

ein Update der Lösungsapproximation durchgeführt werden.
Das Prinzip wird dann sukzessive zum Update der Lösung genutzt, bis der Updateterm und/oder das Residuum des nichtlinearen Problem hinreichend klein sind.

Somit lässt sich die Lösung des nichtlinearen Problems durch eine Folge von linearen Teilproblemen approximieren, wobei diese Teilprobleme eben mit dem klassischen Mehrgitterverfahren gelöst werden können. Dies verändert den durch (4.3) beschriebenen Lösungsprozess.

Zu einem nichtlinearen Problem

$$A_L \langle u_L \rangle = f_L$$

auf dem Feingitter und einer Menge von Matrizen $\mathcal{C} = \{C_0, \ldots, C_L\}$ auf den Gitterleveln zur aktuellen Lösungsapproximation, sowie einer initialen Startlösung $u^{(0)}$ sei der Lösungsprozess nun durch folgende Iteration beschrieben:

$$\left.\begin{aligned} v &\equiv 0 \\ r &= f_L - A_L\left\langle u_L^{(m-1)} \right\rangle \\ u_L^{(m)} &= u_L^{(m-1)} + \mathrm{MG}(L, \mathcal{C}, r, v) \end{aligned}\right\} m = 1, \ldots \tag{4.5}$$

Dies ist das in der Praxis am häufigsten verwendete Verfahren, um nichtlineare Probleme mittels eines Mehrgitteralgorithmus zu lösen.

4.2 Nichtlinearer Mehrgitteransatz

Bei der Übertragung des klassischen Mehrgitteransatzes auf nichtlineare Probleme wird die Grundidee des Mehrgitterverfahrens übergangen. Man nutzt nicht die Zusammenhänge der Gitterstruktur des nichtlinearen Problems aus, sondern benutzt das Konzept einfach zum effizienten Lösen von linearen Teilproblemen.
Als Alternative dazu ist das nichtlineare Mehrgitterverfahren (in der Literatur oft auch **F**ull **A**pproximation **S**cheme) zu sehen, dies versucht die Grundidee des Mehrgitterkonzeptes auf nichtlineare Probleme zu übertragen und auch Informationen der verbesserten Lösungen, die auf den gröberen Gittern bestimmt werden, in den Updateprozess einzubinden. Wir wollen uns bei der Beschreibung vor allem an einer Veröffentlichung von Alfio Borzì [2] orientieren.

4.2.1 Umsetzung

Den entscheidenden Unterschied zwischen einem linearen und nichtlinearen System stellt die Defektberechnung dar. Für einen nichtlinearen Operator ist in der Regel die Systemmatrix A auch von der Lösung u abhängig, so dass

$$A\langle u+v\rangle = f$$

nicht so wie in Gleichung (4.2) zerlegt werden kann, da

$$A\langle u+v\rangle \neq A\langle u\rangle + A\langle v\rangle .$$

Daher lassen sich bei der Berechnung der Korrektur nicht einfach nur die Defektprobleme des feineren Gitters lösen, sondern es ist notwendig eine volle Lösungsapproximation auf dem Grobgitter zu bestimmen, um einen Korrekturwert für das feinere Gitter zu ermitteln. Dies lässt erschließen, warum der Ansatz in der Literatur oft **F**ull **A**pproximation **S**cheme (FAS) genannt wird.

Wir wollen einleitend ähnlich zum klassischen Mehrgitterverfahren die Konstruktion dieses nichtlinearen Mehrgitteransatzes anhand eines Zwei-Gitter-Prozesses beschreiben.

Sei hierzu erneut bei $k = 1$ das Feingitterproblem gegeben und bei $k = 0$ das Grobgitterproblem, welches hinreichend genau gelöst werden kann.

Ausgehend von einer geeigneten Startlösung $u_1^{(0)}$ wird nun ein ν-faches Update der Lösung durchgeführt.

$$u_1^{(\nu)} = \tilde{\mathcal{S}}^{\nu}\left(u_1^{(0)}, f_1\right)$$

Hierbei entspricht $\tilde{\mathcal{S}}$ einem nichtlinearen Updateoperator, der gewisse Ähnlichkeiten zum Glättungsoperator des linearen Mehrgitterverfahrens aufweist. Auf die Wahl dieses Operators wollen wir im folgenden **Kapitel 4.2.2** genauer eingehen.

Nun überlegen wir uns, wie wir mit Hilfe der aktuellen Approximation auf dem Feingitter über ein Problem auf dem Grobgitter einen Korrekturwert der Feingitterlösung bestimmen können. Wie oben bereits dargestellt lässt sich aufgrund der Nichtlinearität die Anwendung der Systemmatrix $A[u+v]$ nicht zerlegen, so dass wir nicht einfach ein Defektproblem lösen können.

Sei v_1 hierzu die Korrektur von $u_1^{(\nu)}$, so dass $u_1^{(\nu)} + v_1$ die exakte Lösung ist, dann gilt

$$\begin{aligned} & A_1 \left\langle u_1^{(\nu)} + v_1 \right\rangle = f_1 \\ \Leftrightarrow \qquad & A_1 \left\langle u_1^{(\nu)} + v_1 \right\rangle - A_1 \left\langle u_1^{(\nu)} \right\rangle = \underbrace{f_1 - A_1 \left\langle u_1^{(\nu)} \right\rangle}_{=:r_1}. \end{aligned}$$

Nun restringieren wir den Residuumsvektor r_1, sowie die Näherungslösung $u_1^{(\nu)}$ und die Lösung $u_1^{(\nu)} + v_1$, auf das Grobgitter. Dabei wählen wir als Restriktionsoperator für das Residuum unseren üblichen Operator $\mathbf{R}_1^0$. Da die Lösungsfunktion auf dem Feingitter jedoch durch dieselbe Funktion dargestellt wird, wählen wir hier die natürliche Einbettung, d.h. im diskreten Fall eine Übernahme der Knotenwerte. Diesen Operator bezeichnen wir nach **Kapitel 3.1.2** mit $\mathbf{I}_1^0$.

Damit ergibt sich

$$\begin{aligned} & A_0 \underbrace{\left\langle \mathbf{I}_1^0 (u_1^{(\nu)} + v_1) \right\rangle}_{=:v_0} = \underbrace{A_0 \left\langle \mathbf{I}_1^0 u_1^{(\nu)} \right\rangle + \mathbf{R}_1^0 r_1}_{=:f_0} \\ \Leftrightarrow \qquad & A_0 \langle v_0 \rangle = f_0 \end{aligned} \tag{4.6}$$

als zu lösendes Grobgitterproblem.
Bei der Komponente v_0 handelt es sich nun jedoch, im Gegensatz zum klassischen linearen Mehrgitteransatz, um eine volle Approximation der Lösungskomponente auf dem Grobgitter und nicht nur einen Korrekturvektor. Insbesondere folgt für die exakte Lösung $u_1^{(\nu)}$

$$\begin{aligned} \mathbf{R}_1^0 r_1 &= \mathbf{R}_1^0 (f_1 - A_1 \left\langle u_1^{(\nu)} \right\rangle) \\ &= \mathbf{R}_1^0 (0) \\ &= 0 \end{aligned}$$

und damit

$$\begin{aligned} & A_0 \langle v_0 \rangle = A_0 \left\langle \mathbf{I}_1^0 u_1^{(\nu)} \right\rangle \\ \Leftrightarrow \qquad & v_0 = \mathbf{I}_1^0 u_1^{(\nu)}. \end{aligned}$$

Auf dem Grobgitter kann nun Gleichung (4.6) exakt gelöst werden und die Feingitterapproximation $u_1^{(\nu)}$ mit dem nun berechneten v_0 verbessert werden. Dabei ist es wichtig, dass wir nicht die Prolongation $u_1^{(\nu+1)} = \mathbf{P}_0^1 v_0$ von v_0 als neue Lösung nutzen, da dies den vollen Interpolationsfehler auf das Feingitter übertragen würden.
Stattdessen ermitteln wir auf dem Grobgitter den Korrekturwert e und fügen diesen prolongiert zur alten Feingitterapproximation hinzu

$$u_1^{(\nu+1)} = u_1^{(\nu)} + \mathbf{P}_0^1 (\underbrace{v_0 - \mathbf{I}_1^0 u_1^{(\nu)}}_{=:e}).$$

Anschließend erfolgt mit

$$u_1^{(\nu+\mu+1)} = \tilde{\mathcal{S}}^\mu (u_1^{(\nu+1)}, f_1)$$

ein μ-faches Lösungsupdate. Dieser Prozess wird nun fortgesetzt, bis das Residuum des Problems hinreichend klein ist.

Die Übertragung des einfachen Zwei-Gitter-Algorithmus auf weitere Gitter erfolgt analog zum klassischen Mehrgitteransatz. Wir ermitteln die Lösung des im Zwei-Gitter-Prozess exakt berechneten Grobgitterproblems durch einen weiteren Zwei-Gitter-Algorithmus, so dass sich das Prinzip auf eine beliebige Anzahl von Gittern übertragen lässt.

4.2.2 Der Updateoperator

Im vorherigen Kapitel haben wir von einem nichtlinearen Updateoperator $\tilde{\mathcal{S}}$ gesprochen. Auf die Wahl dieses Operators und dessen Zusammenhang mit den üblichen Glättern aus **Kapitel 3.2** wollen wir nun genauer eingehen.

Ausgehend von Gleichung (4.4), die den Verbesserungsprozess einer Lösung eines nichtlinearen Problems beschreibt, konstruieren wir einen ähnlichen Prozess wie bei den linearen Glättungsoperatoren. Sei dazu erneut

$$C \triangle u = r$$

gegeben, wobei C einem Operator aus **Kapitel 3.3**, $\triangle u$ dem Update der Lösungskomponente und $r = f - A\left\langle u^{(m-1)} \right\rangle$ dem Residuum des Systems zur aktuellen Näherungslösung $u^{(m-1)}$ entspricht.

Der Updateoperator $\tilde{\mathcal{S}}$ löst dieses System nun ausreichend exakt, um ein Lösungsupdate zu erzielen. Dabei wird Wert darauf gelegt, dass die Lösung hier analog zur linearen Glättung durch einen Löser, der die hochfrequenten Fehleranteile dämpft, approximiert wird. Dadurch erhoffen wir uns ähnliche Eigenschaften, wie sie die Glättunsoperatoren im linearen Fall liefern.

Mit dieser Updatekomponente wird dann erneut eine Korrektur der Lösung durchgeführt

$$u^{(m)} = u^{(m-1)} + \triangle u,$$

so dass wir unsere Lösung auf dem aktuellen Gitter „glätten“. Für einen weiteren Updateschritt werden dann alle lösungabhängigen Systemgrößen neu berechnet.

Wir verwenden innerhalb unserer numerischen Tests zur Ermittlung dieses Updatewertes in der Regel einen linearen Mehrgitteralgorithmus mit einem BiCGStab-Glätter und Jacobi-Vorkonditionierung. Aufgrund der nicht erforderlichen hohen Genauigkeit der Approximation dieses Updates führen wir dabei meist nur einen V-Zyklus durch.

4.2.3 Algorithmus

Zum Lösen des Gleichungssystems $A_L[u_L]u_L = f_L$ wird **Algorithmus 4.2** aufgerufen.

$$u_L^{(m)} = \mathrm{MG}(L, \mathcal{A}, f_L, u_L^{(m-1)}) \qquad m = 1, \ldots \tag{4.7}$$

Wobei $u_L^{(0)}$ eine geeignete Startlösung des Problems auf dem Feingitter darstellt und $\mathcal{A} = \{A_0, \ldots, A_L\}$ entspricht wieder einer Hierarchie aus Systemmatrizen auf den jeweiligen Gitterleveln zur aktuellen Lösungsapproximation. Diese Matrizen werden dabei im Lösungsprozess kontinuierlich aktualisiert.

Algorithmus 4.2: Nichtlinearer V-Zyklus

Aufruf :
`NL_MG(`$k, \mathcal{A}, b, u^{(0)}$`)`
Parameter :
ν - Anzahl der Vorglättungsschritte
μ - Anzahl der Nachglättungsschritte
Eingabe :
k - Gitterlevel
$\mathcal{A}$ - Menge der Systemmatrizen $\mathcal{A} = \{A_0, \ldots, A_k\}$
b - rechte Seite auf Level k
$u^{(0)}$ - initiale Lösungsapproximation auf Level k
Ausgabe :
u - Lösungsapproximation des Systems $A_k u = b$
if *k = 0* **then**
  $u = A_0^{-1} b$ `// Lösen des nichtl. Grobgitterproblems`
else
  $u^{(\nu)} = \tilde{\mathcal{S}}^{\nu}\left(u^{(0)}, b\right)$ `// ν-faches Update`
  $r_k = b - A_k u^{(\nu)}$ `// Berechnung des Residuums`
  $r_{k-1} = \mathbf{R}_k^{k-1} r_k$ `// Restriktion des Residuums`
  $v^{(0)} = \mathbf{I}_k^{k-1} u^{(\nu)}$ `// Restriktion der Lösung`
  $r_{k-1} = r_{k-1} + A_{k-1} v^{(0)}$ `// Neue rechte Seite`
  $v =$ `NL_MG(`$k-1, \mathcal{A}, r_{k-1}, v^{(0)}$`) // rekursives Lösen des nichtl. Grobgitterproblems`
  $u^{(\nu+1)} = u^{(\nu)} + \mathbf{P}_{k-1}^{k}\left(v - v^{(0)}\right)$ `// Grobgitterkorrektur`
  Aktualisierung der Systemmatrizen A_i für $i \leq k$ zur neuen Lösungsapproximation $u^{(\nu+1)}$
  $u = \tilde{\mathcal{S}}^{\mu}\left(u^{(\nu+1)}, b\right)$ `// μ-faches Update`
end

Anzumerken ist bei diesem Algorithmus, dass der Updateoperator $\tilde{\mathcal{S}}$ unter Umständen zusätzliche lösungsabhängige Matrizen benötigt, wie sie in **Kapitel 4.2.2** beschrieben wurden.

Kapitel 5 Numerischer Vergleich

Wir wollen im Folgenden einen Vergleich der vorgestellten Lösungsverfahren durchführen. Dabei untersuchen wir nicht nur den unterschiedlichen Lösungsprozess der Verfahren, sondern wollen insbesondere überprüfen, wie diese mit unterschiedlichen Linearisierungstechniken umgehen.

Es treten also folgende Testsituationen auf, welche wir im Weiteren mit entsprechenden Abkürzungen innerhalb von Grafiken und Tabellen kennzeichnen wollen:

- Linearisierung/Update mittels Newton

  NL_Newton - klassisches Mehrgitter

  FMG-NL_Newton - „geschachteltes“ klassisches Mehrgitter

  MG_Newton - nichtlineares Mehrgitter

  FMG_Newton - „geschachteltes “ nichtlineares Mehrgitter

- Linearisierung/Update mittels Defektkorrektur

  NL_Defcor - klassisches Mehrgitter

  FMG-NL_Defcor - „geschachteltes“ klassisches Mehrgitter

  MG_Defcor - nichtlineares Mehrgitter

  FMG_Defcor - „geschachteltes “ nichtlineares Mehrgitter

Hierbei verwenden wir für alle Mehrgittervarianten ausschließlich den vorgestellte V-Zyklus.

Die Glättung innerhalb der linearen Mehrgitterverfahren übernimmt ein Jacobi-Vorkonditioniertes BiCGStab-Verfahren mit je zwei Vor- und Nachglättungsschritten. Ebenso wird der Updateoperator innerhalb des nichtlinearen Mehrgitterverfahrens durch einen einzigen linearen Mehrgitterzyklus dieser Art ausgeführt. In der Regel werden wir beim nichtlinearen Mehrgitter je eine Vor- sowie Nachglättung mit dem Updateoperator durchführen. Außerdem wird bei diesem Verfahren generell eine zusätzliche initiale Glättung auf dem Feingitter durchgeführt, um eine möglichst geeignete Startlösung zu haben.

Bei den „geschachtelten“ Varianten werden wir pro Level je eine Iteration der Algorithmen zur Korrektur der prolongierten Grobgitterlösung benutzen. Dabei wird beim nichtlinearen Mehrgitterverfahren nur eine Nachglättung aber keine Vorglättung benutzt. Die klassische Variante benutzt den oben beschriebenen Glättungsoperator einmalig pro Gitterlevel. Den ersten Schritt der „geschachtelten“ Verfahren werden wir im Weiteren als FMG-Iteration bezeichnen und als eine Iteration des gesamten Lösungsprozesses betrachten.

Eine schematische Darstellung der Konfigurationen der Verfahren ist in Abbildung 5.1 dargestellt.

Die Grobgitterlösungen werden wir bis auf eine Genauigkeit von 10^{-14} iterativ mit einen BiCGStab-Löser ermitteln und das allgemeine nichtlineare Verfahren wird bis zu einer absoluten Residuumsnorm von 10^{-11} durchgeführt.

- **klassisches Mehrgitter** (Residuumsnorm: 10^{-11})
  - Aktualisierungsoperator: **lineares Mehrgitter** (Iteration: 1)
    - 2 Vor-/Nachglättungen
    - Glättungsoperator: BiCGStab-Verfahren mit Jacobivorkonditionierer

- **„geschachteltes" klassisches Mehrgitter** (Iteration: 1 pro Gitterlevel)
  - Aktualisierungsoperator: **lineares Mehrgitter** (Iteration: 1)
    - 2 Vor-/Nachglättungen
    - Glättungsoperator: BiCGStab-Verfahren mit Jacobivorkonditionierer

  $\Rightarrow$ klassisches Mehrgitter

- **nichtlineares Mehrgitter** (Residuumsnorm: 10^{-11})
  - 1 initiales Update
  - 1 Vor-/Nachupdate
  - Updateoperator: **lineares Mehrgitter** (Iteration: 1)
    - 2 Vor-/Nachglättungen
    - Glättungsoperator: BiCGStab-Verfahren mit Jacobivorkonditionierer

- **„geschachteltes" nichtlineares Mehrgitter** (Iteration: 1 pro Gitterlevel)
  - 1 Nachupdate
  - Updateoperator: **lineares Mehrgitter** (Iteration: 1)
    - 2 Vor-/Nachglättungen
    - Glättungsoperator: BiCGStab-Verfahren mit Jacobivorkonditionierer

  $\Rightarrow$ nichtlineares Mehrgitter

Abbildung 5.1: Konfigurationen der vorgestellten Verfahren.

Bei der Auswertung der Probleme werden wir vor allem Wert auf die Iterationszahl, sowie die Rechenzeit und deren Verteilung auf die Teilprozesse legen. Außerdem geben wir zu jedem Verfahren die Konvergenzrate $\mathcal{K}$ an, die wir mit

$$\mathcal{K} := \left(\frac{\|A \left\langle u^{(n)} \right\rangle - f\|_2}{\|A \left\langle u^{(0)} \right\rangle - f\|_2} \right)^{1/n}$$

ermitteln (vgl. [4, S. 12]). Hierbei entspricht $u^{(0)}$ der Startlösung, sowie $u^{(n)}$ der Näherungslösung nach n Iterationen des Mehrgitterprozesses. Wir wählen innerhalb der numerischen Tests für die Startlösung immer die Nulllösung.

5.1 Modellgleichung

Als Modellproblem wollen wir die zweidimensionale nichtlineare partielle Differentialgleichung

$$-\Delta u + \gamma u e^u = f, \qquad \gamma \in \mathbb{R}$$

betrachten. Dabei beschränken wir uns zum Lösen auf das Einheitsquadrat $\Omega = [0,1]^2$ unter Zuhilfenahme von homogenen Dirichletrandwerten auf $\Gamma = \partial\Omega$.

Dies führt uns zu folgender Problemstellung:

$$\begin{cases} -\Delta u + \gamma u e^u = f & \text{in } \Omega = [0,1]^2 \\ u = 0 & \text{auf } \Gamma \end{cases} \tag{5.1}$$

Für unsere Tests konstruieren wir in allen Fällen die Belastungsfunktion f auf Basis eines gewählten $\gamma \in \mathbb{R}$ und der Referenzlösung

$$u(x,y) = 8\left(x^2 - x^3\right)\sin\left(3\pi y\right).$$

Das Problem (5.1) wird nun mit Hilfe der üblichen Diskretisierungstechniken und bilinearen bzw. biquadratischen konformen Finiten Elementen auf einem Rechteckgitter in ein endlich-dimensionales diskretes Gleichungssystem der Form

$$A[u]u = b$$

überführt. Hierbei entspricht $A[u]$ dem nichtlinearen diskreten Operator des Systems in Abhängigkeit von der Lösung u. Nun können wir auf dieses Gleichungssystem unsere Lösungsalgorithmen anwenden.

Zusätzlich müssen wir für den klassischen Newton-Ansatz die Jacobimatrix des Systems zu der aktuellen Näherungslösung berechnen. Diese lässt sich bei dem Modellproblem verhältnismäßig einfach analytisch bestimmen und auf dieser Basis assemblieren.

Die aus der verwendeten Diskretisierung resultierenden Einträge der Systemmatrix $A[u] = (a_{ij})_{i,j=1}^N$ mit

$$a_{ij} = \varphi_i'\varphi_j' + \gamma\varphi_i\varphi_j e^u,$$

wobei $\{\varphi_1, \ldots, \varphi_N\}$ eine Basis des diskreten Finite Elemente Raumes $\mathcal{V}$ darstellt und u der letzten Näherungslösung entspricht, führen zur Jacobimatrix $J[u] = (j_{ij})_{i,j=1}^N$ mit

$$j_{ij} = a_{ij} + \gamma\varphi_i\varphi_j e^u u. \tag{5.2}$$

Alle Berechnungen werden auf einer Gitterhierarchie aus neun Gittern (d.h. $L = 8$) durchgeführt. Hierbei beginnen wir mit einer regulären Verfeinerung des Einheitsquadrates $[0,1]^2$ aus 16 Zellen und verfeinern bis hin zu der feinsten Gitterstufe mit 1.048.576 Zellen.

5.2 Testsituation 1

Wir beginnen mit einigen Tests zum Verhalten der Algorithmen bei einem vergleichsweise klein gewählten nichtlinearen Anteil. Dazu wollen wir das Modellproblem (5.1) zuerst mit $\gamma = 5$ als nichtlinearen Koeffizienten, sowie bilinearen bzw. biquadratischen Finiten Elementen und den vorgestellten Verfahren lösen.

Bei der Betrachtung der Lösungsmethoden und einer Diskretisierung mit bilinearen Finite Elemente ist der Vorteil der nichtlinearen gegenüber der klassischen Mehrgitterverfahren zu erkennen. Diese benötigen wesentlich weniger Iterationen, um eine ähnliche Norm des Residuums (hier: euklidische Norm) zu erreichen (vgl. Tabelle 5.1). Wir wollen hier nochmals darauf verweisen, dass die „geschachtelte“ Iteration der FMG-Algorithmen wie eine herkömmliche Iteration gezählt wird.

	Iterationen	**Residuumsnorm**	**Konvergenzrate**
Newton-Linearisierung			
NL_Newton	7	$2,70996 \cdot 10^{-13}$	$2,390 \cdot 10^{-2}$
FMG-NL_Newton	5	$2,66564 \cdot 10^{-13}$	$5,35 \cdot 10^{-3}$
MG_Newton	2	$1,94301 \cdot 10^{-13}$	$1,79 \cdot 10^{-6}$
FMG_Newton	2	$3,19683 \cdot 10^{-13}$	$2,29 \cdot 10^{-6}$
Defektkorrektur			
NL_Defcor	12	$3,93424 \cdot 10^{-12}$	$1,42 \cdot 10^{-1}$
FMG-NL_Defcor	7	$3,95287 \cdot 10^{-12}$	$3,51 \cdot 10^{-2}$
MG_Defcor	2	$1,9367 \cdot 10^{-13}$	$1,79 \cdot 10^{-6}$
FMG_Defcor	2	$4,77663 \cdot 10^{-13}$	$2,80 \cdot 10^{-6}$

Tabelle 5.1: Statistiken für bilineare Finite Elemente und $\gamma = 5$.

Auch bei einem Blick auf den Residuenverlauf ist ersichtlich, dass die nichtlinearen Mehrgitterverfahren offensichtlich eine höhere Konvergenzgeschwindigkeit haben als die klassischen Mehrgittervarianten (vgl. Abbildung 5.2). Auffallend ist, dass der FMG-Schritt (entspricht der ersten Iteration) die Residuumsnorm nicht merklich reduziert, jedoch gegenüber den normalen Varianten eine verbesserte Konvergenzgeschwindigkeit im folgenden Iterationsschritt bewirkt. Beim nichtlinearen Mehrgitteransatz ist hierbei zu erwähnen, dass der FMG-Schritt numerisch weniger kostet als der komplette V-Zyklus, so dass bei gleicher Iterationszahl eine Effizienzsteigerung erzielt werden kann.

Nicht zu vernachlässigen ist jedoch, dass das vorgestellte nichtlineare Mehrgitterverfahren einen numerischen Mehraufwand gegenüber seiner linearen Pendants erfordert. Dies resultiert insbesondere aus den zusätzlichen Assemblierungen auf gröberen Gittern als auch aus der aufwendigeren Updateprozedur, die bei unserer Wahl zusätzlich pro Schritt eine Assemblierung von Matrizen auf allen gröberen Gitterleveln erfordert.
In Abbildung 5.3 ist dennoch ersichtlich, dass die nichtlinearen Verfahren im Hinblick auf Rechenzeit konkurrenzfähig sind. Auffallend ist außerdem, dass die nichtlinearen Mehrgitterverfahren anscheinend bei einer einfachen Defektkorrektur zur Linearisierung einen Vorteil gegenüber den klassischen Varianten erzielen. Dies werden wir versuchen in unserer zweiten Testsituation genauer zu untersuchen.

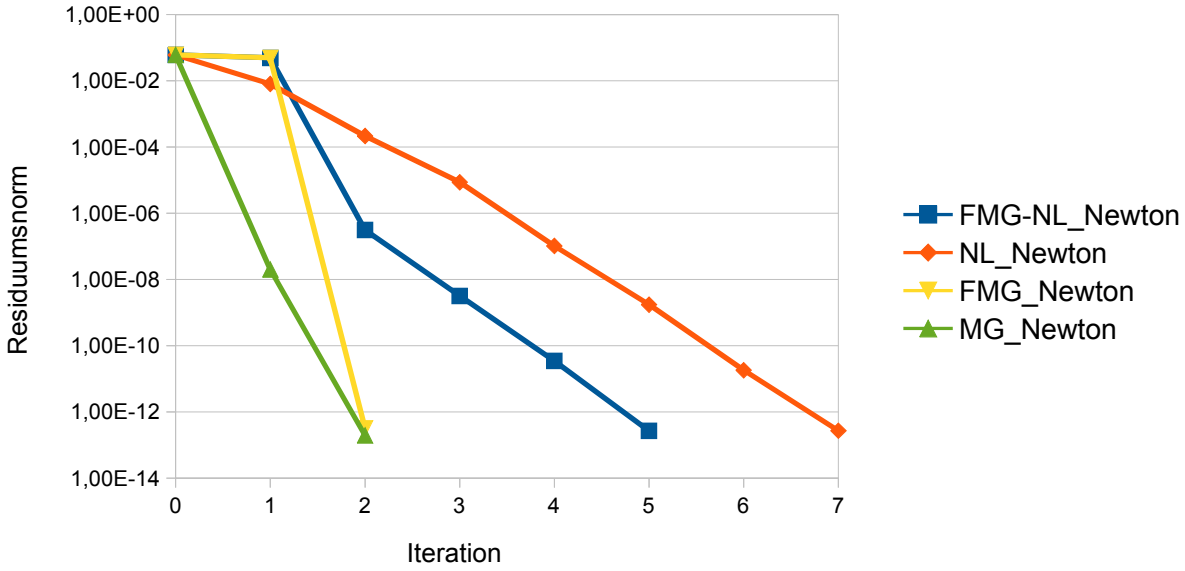


Abbildung 5.2: Residuumsverlauf bei linearen Finiten Elementen mit Newton-Linearisierung und $\gamma = 5$.

Die erstmals in Abbildung 5.3 dargestellte Unterteilung in einzelne Zeitsegmente lässt sich wie folgt verstehen:

Vorbereitung	-	umfasst die Zeit, die zur Erstellung von Gitterhierarchie, Mapping-Strukturen und einmalig zu bestimmenden Transfermatrizen sowie linearen Systemanteilen erforderlich ist
Assemblierung	-	Zeitspanne, die die gesamte Assemblierung von Matrizen innerhalb der Lösungsroutinen benötigt
Lösung linearer Probleme	-	Berechnungszeit, die zur Lösung der entstehenden linearen Probleme anfällt
weitere Operationen	-	Zeit, die zusätzliche Operationen wie z.B. Residuenberechnung und Gittertransfer benötigen

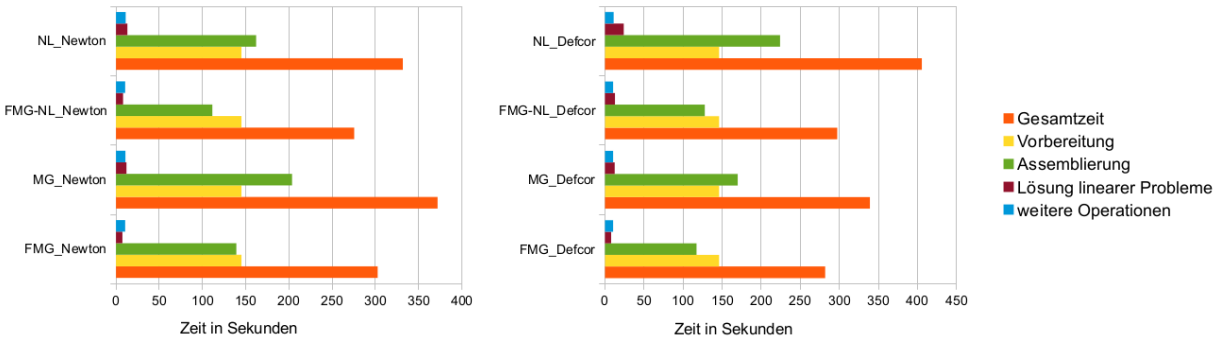


Abbildung 5.3: Rechenzeit der Verfahren bei bilinearen Finiten Elementen und $\gamma = 5$.

Bei der Betrachtung der Assemblierungen von Matrizen pro Gitterlevel ist erkennbar, dass die nichtlinearen Mehrgitterverfahren im Grobgitterbereich eine entscheidende Mehrarbeit leisten (vgl. Abbildung 5.4). Zu erwähnen ist auch hierbei, dass bei den MG-Varianten die Updateprozedur mit

Hilfe eines linearen Mehrgitterverfahrens zusätzliche Assemblierungen auf den gröberen Gittern erfordert. In der genannten Abbildung wurden diese entsprechend mitgezählt.
Aufgrund der Tatsache, dass das Verhältnis der Größe der Systeme zu den einzelnen Gitterleveln bei höherer Dimension steigt, ist anzunehmen, dass insbesondere bei höherdimensionalen Problemen ein Zeitgewinn erzielt werden könnte.

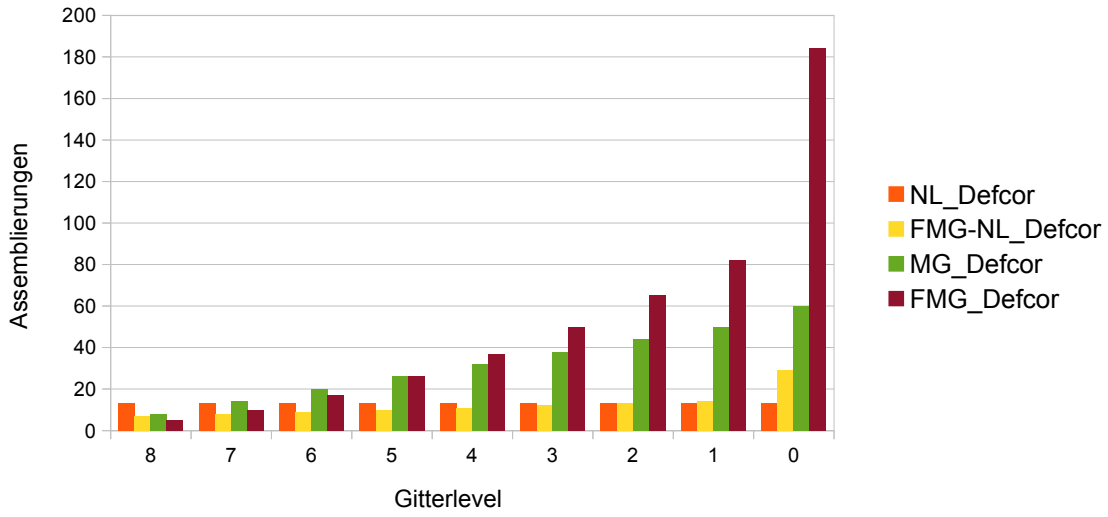


Abbildung 5.4: Anzahl der Assemblierungen von Matrizen pro Gitterlevel für bilineare Finite Elemente und $\gamma = 5$.

Der Übergang zu biquadratischen Finiten Elementen führt nur zu geringfügigen Veränderungen in den Iterationszahlen (vgl. Tabelle 5.2). Es ist jedoch auffallend, dass die Verfahren meist eine geringere Anzahl an Iterationen benötigen. Erwähnenswert ist zudem, dass das *FMG_Newton* Verfahren bereits nach der „geschachtelten“ Iteration und der initialen Glättung auf dem Feingitter das Abbruchkriterium erreicht (das „+“ soll hierbei die zusätzliche Glättung verdeutlichen).

	Iterationen	**Residuumsnorm**	**Konvergenzrate**
Newton-Linearisierung			
NL_Newton	7	$6,93115 \cdot 10^{-13}$	$2,97 \cdot 10^{-2}$
FMG-NL_Newton	2	$8,57235 \cdot 10^{-13}$	$5,04 \cdot 10^{-6}$
MG_Newton	2	$6,73614 \cdot 10^{-13}$	$4,47 \cdot 10^{-6}$
FMG_Newton	1+	$8,50381 \cdot 10^{-13}$	$2,52 \cdot 10^{-11}$
Defektkorrektur			
NL_Defcor	12	$2,92829 \cdot 10^{-12}$	$1,45 \cdot 10^{-1}$
FMG-NL_Defcor	3	$8,84975 \cdot 10^{-12}$	$6,40 \cdot 10^{-4}$
MG_Defcor	2	$6,73054 \cdot 10^{-13}$	$4,47 \cdot 10^{-6}$
FMG_Defcor	2	$6,7344 \cdot 10^{-13}$	$4,47 \cdot 10^{-6}$

Tabelle 5.2: Statistiken für biquadratische Finite Elemente und $\gamma = 5$.

Aufgrund der erhöhten Komplexität des Problems bei Diskretisierung mit biquadratischen Finiten Elementen steigt die Dauer der Rechenzeit hier merklich (vgl. Abbildung 5.5). Das Verhältnis zwischen den Verfahren ändert sich dennoch nicht gravierend. Es ist aber erkennbar, dass bei dieser Testsituation mit den Finiten Elementen höherer Ordnung das klassische „geschachtelte“ Mehrgitterverfahren auch bei einer Linearisierung mittels Defektkorrektur geringfügig überlegen ist.

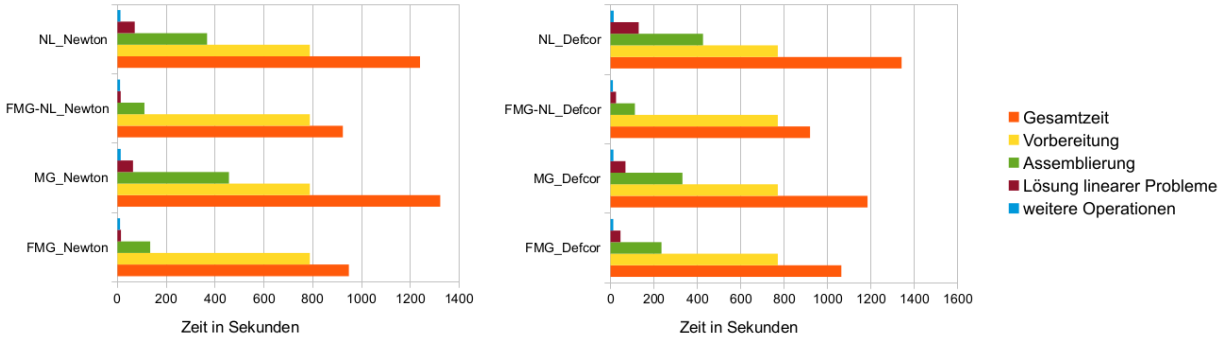


Abbildung 5.5: Rechenzeit der Verfahren bei biquadratischen Finiten Elementen und $\gamma = 5$.

5.3 Testsituation 2

Aufgrund der Tatsache, dass bei einem höheren nichtlinearen Anteil die Systemmatrix und die Jacobimatrix größere Unterschiede aufweisen (vgl. Gleichung (5.2)) erhoffen wir uns nun insbesondere eine Erkenntnis über die Eigenschaften des nichtlinearen Mehrgitterverfahrens bei einer einfachen Defektkorrektur. Außerdem lies sich in **Testsituation 1** mit linearen Finiten Elementen die Vermutung anstellen, dass das nichtlineare Mehrgitterverfahren dort Vorteile gegenüber dem klassischen Mehrgitterverfahren aufweisen könnte. Dies wollen wir nun versuchen zu bestätigen. Dazu wählen wir in unserer zweiten Testsituation, mit $\gamma = 50$, einen größeren nichtlinearen Anteil in der Modellgleichung (5.1).

Betrachten wir nun zu Beginn wieder den Fall der bilinearen Finiten Elemente.

Die Anzahl der Iterationen (vgl. Tabelle 5.3) und die daraus resultierenden Konvergenzraten verhalten sich bei der Linearisierung mit Hilfe der Jacobimatrix ähnlich wie im Falle der weniger starken Nichtlinearität in Tabelle 5.1 von **Testsituation 1**. Bei dem Übergang zur alternativen Linearisierung mittels des Defektkorrekturverfahrens sehen wir einen enormen Anstieg der Iterationszahlen der klassischen Mehrgitterverfahren. Bei den nichtlinearen Mehrgitterverfahren resultiert die Änderung der Linearisierungstechnik jedoch nur in einer geringfügigen Erhöhung.

	Iterationen	**Residuumsnorm**	**Konvergenzrate**
Newton-Linearisierung			
NL_Newton	8	$2,4418 \cdot 10^{-12}$	$4,68 \cdot 10^{-2}$
FMG-NL_Newton	5	$2,66564 \cdot 10^{-13}$	$4,78 \cdot 10^{-3}$
MG_Newton	2	$1,0667 \cdot 10^{-12}$	$3,16 \cdot 10^{-6}$
FMG_Newton	2	$3,1954 \cdot 10^{-13}$	$1,73 \cdot 10^{-6}$
Defektkorrektur			
NL_Defcor	44	$6,75166 \cdot 10^{-12}$	$5,86 \cdot 10^{-1}$
FMG-NL_Defcor	35	$5,89229 \cdot 10^{-12}$	$5,09 \cdot 10^{-1}$
MG_Defcor	3	$2,65414 \cdot 10^{-13}$	$1,36 \cdot 10^{-4}$
FMG_Defcor	3	$1,92524 \cdot 10^{-13}$	$1,22 \cdot 10^{-4}$

Tabelle 5.3: Statistiken für bilineare Finite Elemente und $\gamma = 50$.

Ein Blick auf die Verteilung der Laufzeit der Verfahren in Abbildung 5.6 stellt klar, dass im Falle

eines Defektkorrekturverfahren die nichtlinearen Mehrgitterverfahren eindeutig Vorteile gegenüber den klassischen Varianten aufweisen. In allen Bereichen außer der Vorbereitung sinkt die Rechenzeit, da durch die Nutzung der verbesserten Lösungsapproximationen auf den Grobgittern dort das zu lösende Problem sich zunehmend einem linearen Problem annähert. Bei Erreichen der exakten Lösung u nähert sich die Systemmatrix A, die hierbei zur Linearisierung benutzt wird, der Jacobimatrix des Systems an.

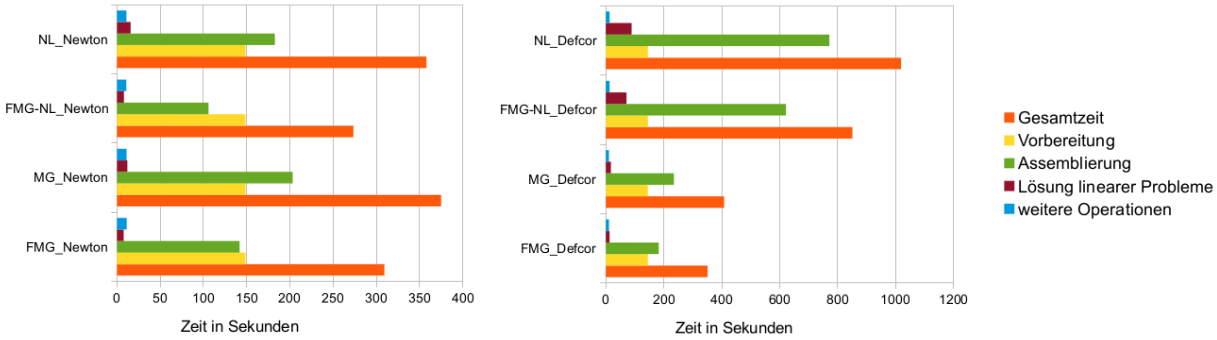


Abbildung 5.6: Rechenzeit der Verfahren bei bilinearen Finiten Elementen und $\gamma = 50$.

Auffallend ist, dass sich bei den nichtlinearen Mehrgitterverfahren die Gesamtzeit des Lösungsprozesses bei einem Wechsel der Linearisierungstechnik nur geringfügig verändert. Bei den klassischen Verfahren ist im Gegensatz dazu ein enormer Anstieg der benötigten Zeit zu erkennen.

Der Übergang zu biquadratischen Finiten Elementen führt erneut zu keinen entscheidenden Änderungen des Konvergenzverhaltens (vgl. Tabelle 5.3). Bemerkbar ist nur, dass - wie in **Testsituation 1** - insbesondere beim *FMG-NL_Newton* und *FMG-NL_Defcor* Verfahren eine Reduktion der Iterationszahlen erzielt werden kann. Erneut kann hier bei dem *FMG_Newton* Verfahren bereits nach dem FMG-Schritt und einer initialen Glättung auf dem Feingitter das Abbruchkriterium erreicht werden.

	Iterationen	**Residuumsnorm**	**Konvergenzrate**
Newton-Linearisierung			
NL_Newton	8	$3,19377 \cdot 10^{-12}$	$5,92 \cdot 10^{-2}$
FMG-NL_Newton	2	$9,18475 \cdot 10^{-13}$	$3,94 \cdot 10^{-6}$
MG_Newton	2	$6,77408 \cdot 10^{-13}$	$3,38 \cdot 10^{-6}$
FMG_Newton	1+	$8,73604 \cdot 10^{-13}$	$1,47 \cdot 10^{-11}$
Defektkorrektur			
NL_Defcor	42	$9,9369 \cdot 10^{-12}$	$5,85 \cdot 10^{-1}$
FMG-NL_Defcor	26	$8,59522 \cdot 10^{-12}$	$4,18 \cdot 10^{-1}$
MG_Defcor	2	$8,71618 \cdot 10^{-13}$	$3,84 \cdot 10^{-6}$
FMG_Defcor	2	$6,72994 \cdot 10^{-13}$	$3,37 \cdot 10^{-6}$

Tabelle 5.4: Statistiken für biquadratische Finite Elemente und $\gamma = 50$.

Zudem ist aus Abbildung 5.7 zu erkennen, dass das Verhältnis der Laufzeiten der einzelnen Komponenten des Lösungsprozesses wieder ähnlich zum Fall mit bilinearen Finiten Elementen ist. Bei der Linearisierung mit Hilfe des Defektkorrekturverfahrens ist auch hier der Vorteil der nichtlinearen Mehrgitterverfahren sichtbar.

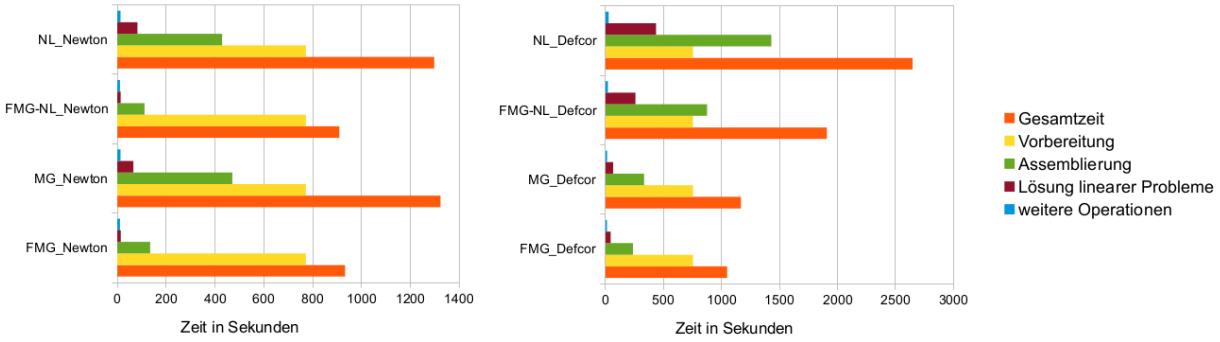


Abbildung 5.7: Rechenzeit der Verfahren bei biquadratischen Finiten Elementen und $\gamma = 50$.

Die Resultate des hier durchgeführten Tests legen nahe, im Falle einer unbekannten oder nicht zu bestimmenden Jacobimatrix und der daraus resultierenden Wahl des Defektkorrekturverfahrens zur Linearisierung, die nichtlinearen Mehrgitterverfahren den klassischen Varianten vorzuziehen. Außerdem können wir aus den beiden bisherigen Testsituationen erschließen, dass der Hauptaufwand der Verfahren in der Assemblierung der Systeme steckt. Durch die Wahl von anderen Updatestrategien lässt sich dies unter Umständen noch variieren. Dies wollen wir in unserer letzten Testsituation versuchen umzusetzen.

5.4 Testsituation 3

Wie bereits angesprochen werden wir nun noch einmal die nichtlinearen Mehrgitterverfahren mit einer einfacheren Updateprozedur durchführen. Dazu bedienen wir uns anstelle des Mehrgitterlösers einem einfachen BiCGStab-Verfahren mit Gauss-Seidel-Vorkonditionierer als Updateoperator. Ziel hierbei ist, zu untersuchen, ob sich die Berechnungszeit mit diesem einfacheren Updateoperator entscheidend verändert oder möglicherweise eine andere Verteilung der Rechenzeit erzielt werden kann. Wünschenswert wäre z.B. eine Reduktion der Assemblierungszeit und dafür unter Umständen ein Ansteigen der Lösungszeit der linearen Hilfsprobleme. Dies würde dazu führen, dass man mit entsprechenden Updateoperatoren eine Optimierung durchführen könnte, wenn gewisse Hardware und spezielle Strukturen eine extrem schnelle Assemblierungs- oder Lösungsroutine ermöglichen.

Für diese Testsituation beschränken wir uns auf das Lösen der Modellgleichung (5.1) mit Linearisierung durch Defektkorrektur und $\gamma = 50$, sowie bilinearen Finiten Elementen. Außerdem wählen wir für einen Updateschritt des vorkonditionierten BiCGStab-Verfahrens 20 Iterationen.

	Iterationen	**Residuumsnorm**	**Konvergenzrate**
MG_Defcor	3	$2,65414 \cdot 10^{-13}$	$1,36 \cdot 10^{-4}$
MG_Defcor_GS	3	$4,11715 \cdot 10^{-13}$	$1,57 \cdot 10^{-4}$
FMG_Defcor	3	$1,92524 \cdot 10^{-13}$	$1,22 \cdot 10^{-4}$
FMG_Defcor_GS	3	$4,83191 \cdot 10^{-13}$	$1,65 \cdot 10^{-4}$

Tabelle 5.5: Statistiken der mod. Verfahren für bilineare Finite Elemente und $\gamma = 50$.

Abbildung 5.8 zeigt, dass es uns durch die Wahl dieser Einstellungen sogar gelingt die Gesamtzeit der Verfahren zu reduzieren. Die modifizierten Verfahren werden hier durch die Endung *GS* gekennzeichnet. Hervorzuheben ist, dass die Iterationszahl der Verfahren unverändert gegenüber

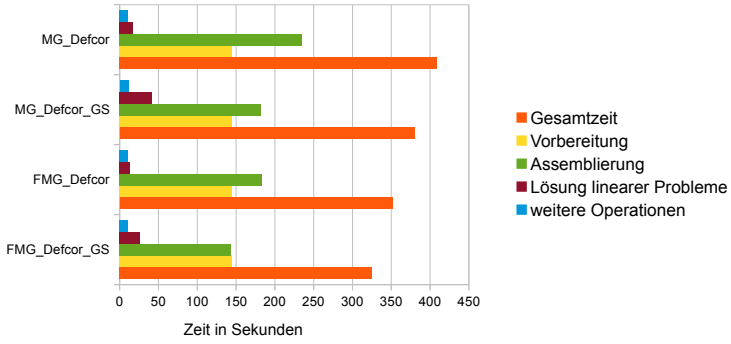


Abbildung 5.8: Vergleich der Rechenzeit der modifizierten Verfahren bei bilinearen Finiten Elementen und $\gamma = 50$.

Testsituation 1 bleibt (vgl. Tabelle 5.5). Die Reduktion der Assemblierungszeit resultiert aus der einfacheren Updateprozedur, die es nicht erfordert, auf gröberen Gittern neue Matrizen zu assemblieren (vgl Abbildung 5.9). Wie erhofft verändert sich dabei auch das Verhältnis zwischen Assemblierungszeit und Lösungszeit der linearen Hilfsproblem.

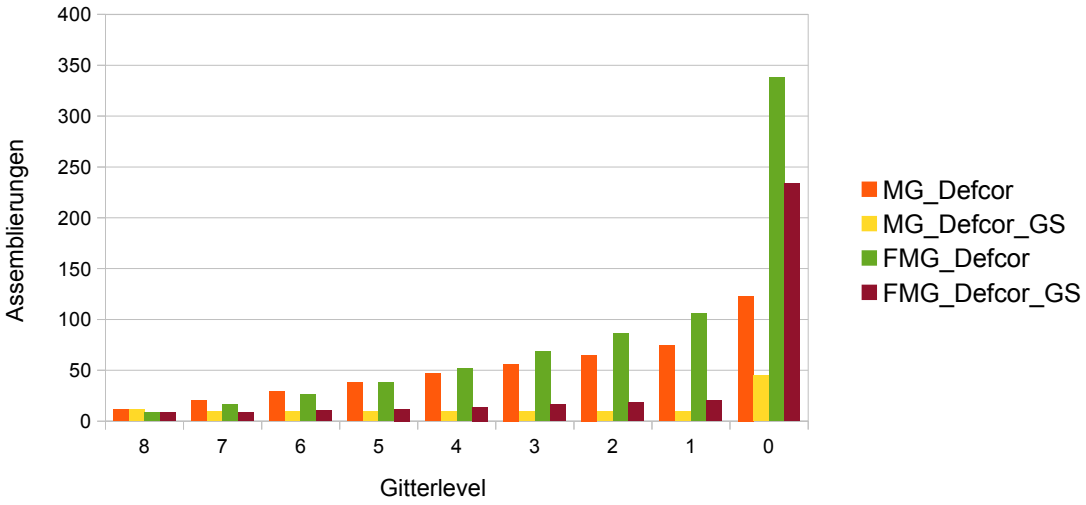


Abbildung 5.9: Anzahl der Assemblierungen der Matrizen pro Gitterlevel für bilineare Finite Elemente und $\gamma = 50$.

Neben dieser exemplarischen Optimierung haben die hier vorgestellten Verfahren natürlich noch weitere Optimierungsmöglichkeiten. So lässt sich neben der Wahl der Updateprozedur unter anderem auch die Genauigkeit dieser Prozesse variieren. Zusätzlich könnten auch mehrere Updateschritte pro Zyklus durchgeführt werden.
Selbst beim klassischen Mehrgitterverfahren lassen sich einige Optimierungen durchführen, wie z.B. eine Variation der Genauigkeit und Art und Weise mit der die linearen Teilprobleme gelöst werden.

Die Möglichkeiten der Anpassung sind vielfältig, so dass wir nicht auf weitere Varianten eingehen wollen. Schlussendlich ist nur hervorzuheben, dass beim nichtlinearen Mehrgitterverfahren und unserem Modellproblem eine einfache Verschiebung des Aufwandes von der Assemblierung hin zur Lösung der Hilfsprobleme möglich ist.

Kapitel 6 Auswertung und Ausblick

Abschließend wollen wir nun die Ergebnisse unserer Tests rekapitulieren, Vor- und Nachteile der einzelnen Verfahren zusammenfassen und bevorzugte Einsatzsituationen angeben. Außerdem wollen wir weitere mögliche Variationen der Verfahren nennen und so einen Ausblick auf zusätzliche Modifikationen geben.

Die erste Erkenntnis, die wir unseren Tests abgewinnen können ist, dass die „geschachtelten“ Iterationen eine Verringerung der Iterationszahlen aller Verfahren liefern können. Insbesondere in **Testsituation 1** konnten wir sehen, dass diese sogenannten FMG-Schritte zwar nicht zu einer erheblichen Reduktion der Residuumsnorm führen, jedoch in der folgenden Iteration eine erhöhte Konvergenzgeschwindigkeit ermöglichen. Vor allem beim nichtlinearen Mehrgitterverfahren lohnt sich die Anwendung dieses Prozesses, da dort der Aufwand der FMG-Iteration, mit unseren Konfigurationen, geringer ist als eine Anwendung des vollen V-Zyklus.

Es ließ sich feststellen, dass der numerische Aufwand einer nichtlinearen Mehrgitteriteration größer ist als der der klassischen Variante. Jedoch ergab sich eine vergleichbare Gesamtzeit bei der Lösung der Probleme mit den verschiedenen Methoden. Dabei fällt insbesondere auf, dass bei einer klassischen Newton-Linearisierung die Laufzeit der Verfahren, sowohl bei geringer als auch hoher Nichtlinearität, vergleichbar ist.

Der Hauptaufwand aller Lösungsroutinen liegt in den vorgestellten Tests in der Assemblierung der nichtlinearen Systeme. Dabei ist bei den klassischen Methoden die Anzahl der Assemblierungen gleichmäßig auf die einzelnen Gitterlevel verteilt, wohingegen beim nichtlinearen Mehrgitterverfahren ein Schwerpunkt bei der Arbeit auf den gröberen Gittern liegt. Daher ist anzunehmen, dass bei höherdimensionalen Problemen ein stärker reduzierter Assemblierungsaufwand beim nichtlinearen Mehrgitterverfahren erzielt werden kann, da dort die Verhältnisse der Systemgrößen bei einem Gitterwechsel höher sind.

Außerdem haben wir in **Testsituation 3** feststellen können, dass es beim nichtlinearen Mehrgitterverfahren mit einer einfacheren Updateroutine möglich ist den Assemblierungsaufwand zu reduzieren. Wir konnten dort mit Hilfe eines einfachen BiCGStab-Verfahrens mit Gauss-Seidel-Vorkonditionierer sogar eine Reduktion der gesamten Lösungszeit erreichen.

Beim Übergang von einer Newton-Linearisierung zu einer einfachen Linearisierung mittels Defektkorrektur war erkennbar, dass bei einem kleinen nichtlinearen Anteil die beiden Verfahren weiterhin vergleichbare Gesamtzeiten für den Lösungsprozess benötigen. Nachdem wir den nichtlinearen Koeffizienten jedoch erhöhten, fiel auf, dass die nichtlinearen Mehrgitterverfahren einen klaren Vorteil gegenüber den klassischen Varianten erreichen konnten. Zusätzlich ist dabei zu erwähnen, dass die Gesamtzeit der nichtlinearen Verfahren beim Übergang von Newton- zu Defektkorrektur-Linearisierung nur geringfügig anstieg. Da in unseren Tests die Einträge der Jacobimatrix analytisch bekannt gewesen sind und dies nicht immer der Fall ist, ist dies ein entscheidender Vorteil der nichtlinearen Mehrgitterverfahren. Wir können ohne Bestimmung der Jacobimatrix eine vergleichbare Konvergenzgeschwindigkeit erzielen.

Auffallend ist bei allen vorgestellten Verfahren, dass die Gesamtzahl der Iterationen beim Übergang zu Finiten Elementen höherer Ordnung (in unserem Fall von bilinearen zu biquadratischen) gesunken ist. Durch die zunehmende Komplexität der Systeme stieg jedoch natürlicherweise die Gesamtzeit der Lösungsprozesse gegenüber der bilinearen Variante, was aber durch den Gewinn an Genauigkeit

kompensiert werden kann.

Alles in allem können wir somit erschließen, dass die nichtlinearen Mehrgitterverfahren ihre erhöhte Komplexität gegenüber den klassischen Varianten rechtfertigen. Bei einfachen Problemen erhalten wir im Vergleich zu den herkömmlichen Varianten vergleichbare Resultate und unter gewissen Umständen, wie der erhöhten Nichtlinearität in unserer **Testsituation 2**, können wir entscheidende Vorteile mit den nichtlinearen Mehrgitterverfahren erzielen. Insbesondere ist der mögliche Verzicht auf die Assemblierung der Jacobimatrix, der sich in unseren Tests als konkurrenzfähige Alternative herausgestellt hat, eine Ersparnis an numerischem Aufwand und zugleich eine Möglichkeit flexiblere Löser zu erstellen, da nicht problemspezifisch eine Implementierung der analytisch ermittelten Besetzung dieser Matrix vonnöten ist.

Allgemein hat sich außerdem, sowohl für den klassischen Mehrgitteralgorithmus als auch für das vorgestellte nichtlineare Mehrgitterverfahren, die FMG-Iteration als lohnend herausgestellt. Sie führt mit einem vergleichsweise geringen numerischen Aufwand zu einer guten Startlösung, die eine erhöhte Konvergenzgeschwindigkeit des Gesamtverfahrens ermöglicht.

Schlussendlich wollen wir nochmals darauf hinweisen, dass es bei allen Verfahren vielfältige Optimierungsmöglichkeiten gibt. Neben der hier vorgestellten Variation des Updateoperators der nichtlinearen Mehrgitterverfahren lässt sich an vielen Schrauben der Algorithmen drehen. Die Variation der Updateschritte im äußeren Algorithmus sind neben der Wahl des Updateoperators und der von diesem Operator erwünschten Genauigkeit nur einige wenige Möglichkeiten das Konvergenzverhalten zu beeinflussen.

A Literaturverzeichnis

[1] Köster, M.: *Robuste Mehrgitter-Krylowraum-Techniken für FEM-Verfahren.* Diplomarbeit, TU Dortmund, Fakultät für Mathematik, 2004.

[2] Borzì, A.: *Introduction to multigrid methods.* Karl–Franzens–Universität Graz. http://www.uni-graz.at/imawww/borzi/mgintro.pdf.

[3] Henson, V.E.: *Multigrid methods for nonlinear problems: an overview.* Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, 5016:36-48, 2003.

[4] Gödekke, D.: *Schnelle Löser.* Vorlesungsskript, TU Dortmund, 2013.

[5] Hackbusch, W.: *Multi-Grid Methods and Applications.* Springer, 1985. Berlin-Heidelberg, ISBN 3540127615.

[6] Rannacher, R.: *Numerische Mathematik 2 (Numerik Partieller Differentialgleichungen).* Vorlesungsskript, Universität Heidelberg, 2008.

[7] Olshanskii, M. A.: *Lecture notes on multigrid methods.* Lomonossow-Universität Moskau, 2012. ISBN 978-5-901854-12-9.

[8] Mandel, J. und Parter, S.V.: *On the multigrid F-cycle.* Journal of Computational and Applied Mathematics, 37(1):19-36, 1990.

[9] Wesseling, P.: *An Introduction to Multigrid Methods.* John Wiley & Sons Ltd., 1992. http://www.mgnet.org/mgnet-books-wesseling.html.

[10] Averick, B. M., Moré, J. J. et al.: *Computing Large Sparse Jacobian Matrices Using Automatic Differentiation.* SIAM Journal on Scientific Computing, 15:285-294, 1993.

[11] Kelley, C. T.: *Frontiers in Applied Mathematics: Iterative Methods for Linear and Nonlinear Equations*, Kap. Broyden's method. SIAM, 1995. http://www.siam.org/books/textbooks/fr16_book.pdf

Eidesstattliche Versicherung

Altenbernd, Mirco 144003

Name, Vorname Matr.-Nr.

Ich versichere hiermit an Eides statt, dass ich die vorliegende Bachelorarbeit/~~Masterarbeit~~* mit dem Titel

Numerischer Vergleich nichtlinearer und linearer Mehrgitterverfahren zum Lösen von partiellen Differentialgleichungen

selbstständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate kenntlich gemacht. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Ort, Datum Unterschrift

*Nichtzutreffendes bitte streichen

Belehrung:

Wer vorsätzlich gegen eine die Täuschung über Prüfungsleistungen betreffende Regelung einer Hochschulprüfungsordnung verstößt, handelt ordnungswidrig. Die Ordnungswidrigkeit kann mit einer Geldbuße von bis zu 50.000,00 € geahndet werden. Zuständige Verwaltungsbehörde für die Verfolgung und Ahndung von Ordnungswidrigkeiten ist der Kanzler/die Kanzlerin der Technischen Universität Dortmund. Im Falle eines mehrfachen oder sonstigen schwerwiegenden Täuschungsversuches kann der Prüfling zudem exmatrikuliert werden. (§ 63 Abs. 5 Hochschulgesetz - HG -)

Die Abgabe einer falschen Versicherung an Eides statt wird mit Freiheitsstrafe bis zu 3 Jahren oder mit Geldstrafe bestraft.

Die Technische Universität Dortmund wird gfls. elektronische Vergleichswerkzeuge (wie z.B. die Software „turnitin“) zur Überprüfung von Ordnungswidrigkeiten in Prüfungsverfahren nutzen.

Die oben stehende Belehrung habe ich zur Kenntnis genommen:

Ort, Datum Unterschrift