Multigrid Methods for Nonlinear Second Order Partial Differential Operators

by

Keeran Jakob Brabazon

Submitted in accordance with the requirements for the degree of Doctor of Philosophy.

UNIVERSITY OF LEEDS

The University of Leeds
School of Computing

October 2014

The candidate confirms that the work submitted is his/her own, except where work which has formed part of jointly authored publications has been included. The contribution of the candidate and the other authors to this work has been explicitly indicated below. The candidate confirms that appropriate credit has been given within the thesis where reference has been made to the work of others.

Some parts of the work presented in Chapters 6, 7 and 8 have been published in the following articles:

K.J. Brabazon, M.E. Hubbard, and P.K. Jimack, "Nonlinear Multigrid Methods for Second Order Differential Operators with Nonlinear Diffusion Coefficient", *Computers & Mathematics with Applications*, 68:1619–1634, 2014.

The derivation of the framework presented in the above paper, as well as the design and implementation of experimental validation was carried out solely by the author of this thesis. Joint credit is taken for the presentation of the work in the paper.

Acknowledgements

There are many people that I would like to thank for their contributions during the course of my PhD. I hope that I have already made clear to people how much their support has meant to me, and request that the omission of any important name from this page should not be taken as a lack of gratitude but rather an oversight due to a leaky memory.

First and foremost I would like to thank my supervisors Peter Jimack and Matthew Hubbard for their invaluable support throughout the project. I appreciate the amount of time they took for me, and with their guidance I have created a body of work that I am proud to put my name to.

I would like to thank all the people that made the PhD as painless and stress free as possible over the last three and a half years – my friends. First of all I would like to thank friends that are also colleagues. Thanks to my office mates, past and present, for creating a great working atmosphere and providing interesting discussion on topics other than my own research. From the research group special thanks go to Tom Ranner for his clear and interesting advice on mathematical topics, and to Mark Walkley and David Head for always being around in their offices and willing to listen to me bounce ideas off of them. From other institutions I would like to thank, in particular, Eike Müller and Rob Scheichl for help in finding direction with my project, as well as for fascinating discussions. In addition, Judi and Teresa must be thanked for being my first port of call for, literally, everything.

Great thanks also go out to all of the friends I made outside of the School of Computing: anyone I played football with for giving me a great run around and not complaining when I kicked them; my Italian 'family' Giulio, Bianca, Carlo, Andrea, Tatiana and Elisa for lots of great food and engaging conversation; Rob Holbrook for chasing me around a squash court and for keeping a smile on his face when I did my best to wipe it off!; Fengwei 'Tim' Yang for being one of the most genuinely happy people I have ever met; Andrzej 'Muffin Man' Warzyński for being one of the best friends I will ever have and for helping to keep my work/life balance in a sensible ratio; long time friend Dave Harrison who has a lot to answer for in helping to form the geek that I am today; and last but not least my family who put up with my lamenting without a clue what I was talking about.

I reserve my largest thanks for last, which go to Giorgia Magnatti. The love and support she has shown me have been incredible and have kept me smiling. Although I have much to be proud of from my time in Leeds, I am most proud of her.

Abstract

This thesis is concerned with the efficient numerical solution of nonlinear partial differential equations (PDEs) of elliptic and parabolic type. Such PDEs arise frequently in models used to describe many physical phenomena, from the diffusion of a toxin in soil to the flow of viscous fluids. The main focus of this research is to better understand the implementation and performance of nonlinear multigrid methods for the solution of elliptic and parabolic PDEs, following their discretisation. For the most part finite element discretisations are considered, but other techniques are also discussed.

Following discretisation of a PDE the two most frequently used nonlinear multigrid methods are Newton-Multigrid and the Full Approximation Scheme (FAS). These are both very efficient algorithms, and have the advantage that when they are applied to practical problems, their execution times scale linearly with the size of the problem being solved. Even though this has yet to be proved in theory for most problems, these methods have been widely adopted in practice in order to solve highly complex nonlinear (systems of) PDEs.

Many research groups use either Newton-MG or FAS without much consideration as to which should be preferred, since both algorithms perform satisfactorily. In this thesis we address the question as to which method is likely to be more computationally efficient in practice. As part of this investigation the implementation of the algorithms is considered in a framework which allows the direct comparison of the computational effort of the two iterations. As well as this, the convergence properties of the methods are considered, applied to a variety of model problems. Extensive results are presented in the comparison, which are explained by available theory whenever possible. The strength and range of results presented allows us to confidently conclude that for a practical problem, discretised using a finite element discretisation, an improved efficiency and stability of a Newton-MG iteration, compared to an FAS iteration, is likely to be observed. The relative advantage of a Newton-MG method is likely to be larger the more complex the problem being solved becomes.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Partial differential equations (PDEs) are frequently used for the modelling of many physical processes. They provide an estimation of continuous phenomena arising in physical and engineering processes. Their solution is known in analytic cases for only few model problems, and when applied to real world problems their exact solution is most often not known. In this case the solution must be estimated. The most useful way in which an approximation can be gained is by solving a PDE (or system of PDEs) for the value of the unknown function(s) at discrete points, which can be done with the aid of a computer.

In order that the discussion given in this thesis may be kept concise it is assumed that the reader is familiar with the following topics. From calculus, it is assumed that the reader is familiar with differentiation on the real numbers, Riemann integration, and the numerical approximation of derivatives and integrals. A basic knowledge of function spaces is required, in particular knowledge of inner products and norms to measure functions, as well as basic concepts such as open/closedness, boundedness, compactness, *etc*. Knowledge of Hilbert spaces and Sobolev spaces is advantageous, although these are not required in depth in order to understand the main content of the thesis. From linear algebra the concept of a matrix and vector is required, and spectral properties of a matrix (such as eigenvalues, eigenvectors, spectral radius, *etc*.) are also required. For readers unfamiliar with these topics we suggest some introductory texts to read. For an introduction to the calculus required for the description of numerical methods see [46]. An introduction to functional analysis is given in [152], and a discussion of applied functional analysis can be found in [83]. A good introductory text on linear algebra is [168] and a more complete

reference is [78]. Most of the mathematics used in this thesis can be found in the excellent reference book on PDEs by Evans [71].

1.1 Subject of the Thesis

In this thesis partial differential equations of the form

$$-\nabla \cdot (a(u, \nabla u, \vec{x}) \nabla u(\vec{x})) = f(\vec{x}), \tag{1.1}$$

and

$$\frac{\partial}{\partial t} b(u, \vec{x}, t) - \nabla \cdot (a(u, \nabla u, \vec{x}, t) \nabla u(\vec{x}, t)) = f(\vec{x}, t), \tag{1.2}$$

are considered for unknown function u; (non-)linear functions a and b of u and its gradient ∇u; known function f; spatial variable $\vec{x}$; and time variable t. Equations (1.1) and (1.2) are examples of PDEs of elliptic and parabolic type, respectively. A class of algorithms which are often used in practice for these types of problems are so-called *multigrid methods* (see Chapter 4). When function a in (1.1) or (1.2) depends on u or ∇u, the PDE is nonlinear, for which there are two main variants of multigrid – *Newton-Multigrid* or *Nonlinear Multigrid* (in particular the Full Approximation Scheme - FAS). Chapter 4 gives an introduction to these. It is the performance of these algorithms applied to nonlinear problems of the form (1.1) and (1.2) which is the subject of this thesis. An introduction to the discretisation of PDEs and their solution is given in later chapters. For an outline of the content of the thesis see Section 1.3 below.

1.2 Main Achievements of the Thesis

The achievements of this thesis can be summarised as follows:

- A novel framework is developed in Chapter 6 which allows for the direct comparison of the computational effort required in Newton-Multigrid and Nonlinear Multigrid methods. Estimations of running times are sharp for simple model problems, and the framework serves to highlight the expected performance of the methods, relative to each other, when applied to increasingly complex problems.

- Application of Newton-Multigrid and Nonlinear Multigrid methods to an elliptic model problem in Chapter 7 shows important implementational considerations for

each of the algorithms which serve to highlight weaknesses and strengths of the iterations.

- A relation between the Newton-Multigrid and Nonlinear Multigrid methods is highlighted in Subsection 4.2.3, which is supported by results given in Section 7.6, and leads to the proposition that the asymptotic convergence of a Nonlinear Multigrid method is dependent upon the convergence of a Newton iteration.

- Application of Newton-Multigrid and Nonlinear Multigrid methods to an elliptic problem with discontinuous coefficients in Section 7.7 shows the robustness in convergence of Newton-Multigrid and Nonlinear Multigrid with respect to the positioning of coefficient functions which are not aligned on coarse grids. The results are tied in with existing linear theory, and this is the first time that an investigation of discontinuous coefficients has been performed in the case of a nonlinear PDE.

- Extensive results on the robustness of Newton-Multigrid and Nonlinear Multigrid with respect to initial estimates, time-step parameters and discontinuities in coefficient functions are given in Chapters 7 and 8. These allow for us to be confident in the statement that Newton-Multigrid is likely to be a much more robust and computationally efficient algorithm, compared to Nonlinear Multigrid, for practical problems. Results also highlight the necessity for assumptions to be made *a priori* regarding the solution of a nonlinear problem, before any qualitative estimate for the convergence behaviour of a nonlinear iteration can be made.

1.3 Outline of the Thesis

This thesis is split into several chapters, which aim to give a contained discussion of distinct topics of interest. Chapters 2 to 5 together give a background of the numerics and mathematics involved in the solution of discrete nonlinear PDEs using multigrid methods. Chapter 2 begins with common discretisations of PDEs onto spatial grids, with particular focus on finite element methods. A brief note regarding discretisation in time is also given. Chapter 3 discusses common techniques for the solution of PDEs discretised on grids and introduces important components of a multigrid iteration. Chapter 4 gives an introduction to multigrid methods by first considering linear multigrid methods, and then their application in a nonlinear setting. Chapter 5 outlines relevant components of the known convergence theory of multigrid methods.

The remaining Chapters 6 to 8 are related to the comparison of Newton-Multigrid and Nonlinear Multigrid methods. The main novel contribution is given in Chapter 6, in which

a framework for the direct comparison of the relative execution time of Newton-Multigrid and Nonlinear Multigrid methods is developed. This is a more in-depth discussion of the framework introduced in [22]. Quantitative estimates of the execution time of the iterations are made, and a qualitative discussion of the expected behaviour, in terms of computational efficiency of the methods, is made for increasingly complex problems.

Results are given in Chapters 7 and 8 comparing the execution time and robustness of the iterations for different model problems. These results are more extensive than results given in [22]. An elliptic type model problem of the form (1.1) is given in Chapter 7. This includes an investigation of the use of different initial estimates, discontinuous coefficients and stabilisation methods of the algorithms. The discussion in Section 7.6 is interesting as it highlights a connection between Newton-MG and Nonlinear Multigrid methods, which suggests that the asymptotic convergence of a Nonlinear Multigrid method will depend on the convergence of a Newton iteration.

Chapter 8 gives results for Newton-Multigrid and Nonlinear Multigrid methods applied to the solution of discretised PDEs of parabolic type. In particular, results demonstrate the robustness of the methods with respect to a time-step parameter. The model problems considered in this chapter are presented in order of their complexity, the most complex problem being presented last. The results for the execution time of the iterations supports predictions made using the theory in Chapter 6. At the beginning of each of the Chapters 6 to 8 the outline of the chapter is briefly introduced, and a summary of the main results and possible extensions and future work are included at the end of each chapter. These conclusions are summarised and repeated in Chapter 9.

Chapter 2

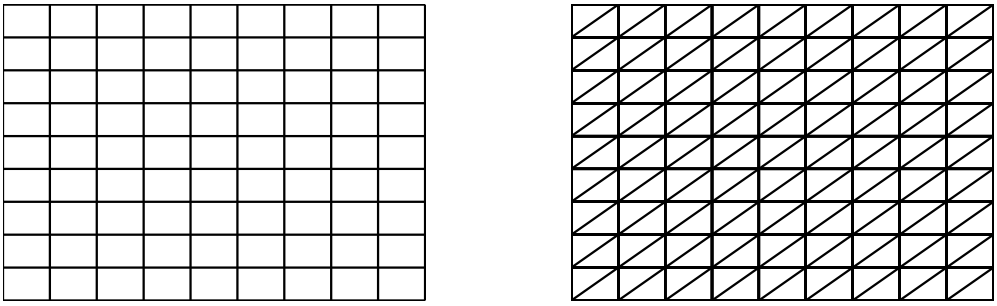
The Discretisation of PDEs

Partial Differential Equations (PDEs) are used in science and engineering to create (approximate) models of some physical process or processes. For many practical applications an analytic solution of a PDE, or system of PDEs, is not known. Hence the solution must be approximated numerically. The solution can most easily be approximated on a computer, where arbitrary continuous functions must be approximated by some discrete counterpart. The process of moving from a continuous to a discrete problem is known as *discretisation*. In discretising, care must be taken that important underlying properties of the solution to the continuous problem are preserved. In particular a discretisation should allow for a solution to be found, which is *convergent*. That is, the discrete solutions should approach the continuous solution as the number of unknowns in an approximation is increased. Discretisation schemes can be chosen to preserve properties such as continuity of mass, momentum or flux and/or continuity of derivatives of the solution.

There are many different choices for discretisation schemes. Sections 2.2, 2.3 and 2.4 give an introduction to several choices for discretisation methods. Two methods that are especially popular in practice are finite difference methods and finite element methods, which are discussed in Sections 2.2 and 2.3, respectively. Other methods are useful, but beyond the scope of this thesis. In Section 2.4 some other possible discretisation methods are briefly introduced and the interested reader is directed to the literature for more information. Before discussing different discretisation schemes, necessary mathematical preliminaries are introduced in the next section.

2.1 Discretising on a Grid

In the scope of this thesis a problem is solved on a domain $\Omega \subset \mathbb{R}^d$ for $d \in \{1, 2, 3\}$. It is assumed that Ω is polygonal with boundary $\partial\Omega$. There are a number of ways to discretise a function on Ω. Spectral methods (described in more detail in Section 2.4) take a finite combination of functions defined globally on Ω. Another approach is to consider Ω partitioned into smaller components and to define a function on Ω as a sum of functions defined locally on these components. To this end a *grid* $\Omega_h \subset \Omega$ is defined as a finite set of points $\mathcal{N}_h \in \mathbb{R}^d$ connected by a set of edges $\mathcal{E}_h$, as depicted in Figure 2.1.



(a) Example of a rectangular grid (b) Example of a triangular grid

Figure 2.1: Examples of different types of grid

Ω_h may also be referred to as a mesh, and both grid and mesh will be used synonymously in this thesis. Figure 2.1a gives an example of a rectangular grid and Figure 2.1b gives an example of a structured triangular grid, although arbitrary polygons may be formed on a set of nodes. The nodes in $\mathcal{N}_h$ need not be connected by edges in a structured manner, giving rise to an unstructured grid. Grids can also be formed in three dimensions, in which case the edges are the boundaries of faces, which enclose three-dimensional volumes. The set of faces of a grid Ω_h (where a face of a two-dimensional grid is an edge) is given by $\mathcal{F}_h$. We denote by $\#\mathcal{N}_h$, $\#\mathcal{E}_h$ and $\#\mathcal{F}_h$ the number of nodes, edges and faces on a grid, respectively. A grid is called *uniform* in $\mathbb{R}^d$ if the spacing between neighbouring nodes is uniform in each of the component directions. Both of the grids in Figure 2.1 are examples of uniform two-dimensional grids.

A *triangulation* $\mathcal{T}_h$ on Ω_h is a set of pairwise disjoint open polygons $\{T_i\}$ such that

the intersection

$$\overline{T_i} \cap \overline{T_j} = \begin{cases} \emptyset, & \text{or} \\ f \in \mathcal{F}_h, & \text{or} \\ e \in \mathcal{E}_h, & \text{or} \\ \vec{x} \in \mathcal{N}_h, & \end{cases}$$

for $i \neq j$. As presented here, a triangulation need not be defined on a triangular grid and the above definition holds in the case of arbitrary polygons. The polygons $T \in \mathcal{T}_h$ are known as *elements* and $\#\mathcal{T}_h$ gives the number of elements on a grid Ω_h. The elements in Figure 2.1a are rectangles, and the elements in Figure 2.1b are triangles. For the purposes of this thesis we are most interested in a *simplicial* grid, which is triangular in two-dimensions and tetrahedral in three.

For an element the value $h(T)$ is the *diameter* of element $T \in \mathcal{T}_h$, which is defined as the longest distance between two points on the element. For a triangle this is the magnitude of the longest edge. We also define

$$h \equiv \max_{T \in \mathcal{T}_h} h(T) \tag{2.1}$$

and let $\rho(T)$ be the diameter of the largest ball contained in $T \in \mathcal{T}_h$. A triangulation is said to be *shape regular* if there exists some $\kappa > 0$ such that

$$\frac{\rho(T)}{h(T)} \geq \kappa, \quad \forall T \in \mathcal{T}_h. \tag{2.2}$$

A triangulation is *quasi-regular* if condition (2.2) is changed to

$$\frac{\rho(T)}{h} \geq \kappa, \quad \forall T \in \mathcal{T}_h. \tag{2.3}$$

There are two main ways in which a grid is used to describe a discrete function. Using a finite difference (see Section 2.2) or a finite volume (see Section 2.4) discretisation a discrete function is approximated using point-wise values on the grid. This may be the grid nodes, the mid-points between edges or the centres of gravity of the elements. For a finite element approximation (see Section 2.3) the approximation is formed by taking a combination of functions with support on a finite number of neighbouring elements. The

functions can be defined using a finite number of values of the function or its derivatives at unique points on an element, which are called the degrees of freedom of the function. The degrees of freedom may be defined at grid points, edges or faces on the element or the element itself, or combinations of the above.

In the next sections we describe in more detail the way in which a function may be discretised on a grid in the case of finite difference and finite element schemes. Section 2.4 outlines some other methods used for discretising PDEs.

2.2 Finite Difference Methods

In this section the finite difference method is introduced by considering the one-dimensional discretisation of the equation

$$-a\nabla \cdot [\nabla u(\vec{x})] + bu(\vec{x}) = f(\vec{x}), \tag{2.4}$$

where constants $a, b > 0$. The function u is the unknown in (2.4) and for the remainder of this section and the remainder of the thesis the explicit dependence on $\vec{x}$ of the functions is dropped whenever the notation is clear. Discretisation in two and three dimensions follows using the same principles discussed here. For this brief discussion boundary values are ignored, but some difficulties in dealing with boundary values are discussed below.

Consider that the uniform grid Ω_h shown in Figure 2.2 has been given and that the values of the solution u are sought at the nodes $\mathcal{N}_h$. As mentioned previously it is possible

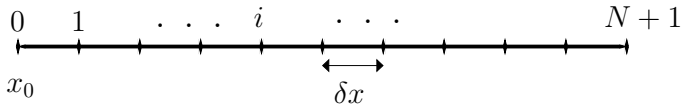


Figure 2.2: Example one-dimensional grid

to use other points on the grid at which to find the value of the solution and discretisation at these points is performed in a similar manner to that described below. Let δx be the spacing between the grid nodes, $x_i = x_0 + i\delta x$ be a nodal value, $N = \#\mathcal{N}_h$, $u_i = u(x_i)$, $\{\nabla \cdot \nabla u\}_i = \nabla \cdot \nabla u(x_i)$ and $f_i = f(x_i)$. Applying (2.4) at each grid point individually gives the following system of equations to solve

$$-a\left\{\nabla \cdot \nabla u\right\}_i + bu_i = f_i, \quad i = 1, \ldots, N. \tag{2.5}$$

The nodal values $u_i \in \mathbb{R}$ may be considered constants to solve for, even though they are function evaluations at a given point in space. Equation (2.5) also contains values $\{\nabla \cdot \nabla u\}_i$, which are not known, and which we do not wish to solve for as we are interested in the function values u_i. It is possible, using a divided difference approximation, to estimate the values of the derivatives of the function u in terms of the nodal values on the grid. Using a Taylor series expansion in one dimension about grid node x_i a second order accurate approximation (with respect to the grid spacing) to the second derivative is given by the centred difference formula [166, pg. 7]

$$\{\nabla \cdot \nabla u\}_i = \left\{ \frac{\mathrm{d}^2}{\mathrm{d}x^2} u \right\}_i \approx \frac{u_{i+1} - 2u_i + u_{i-1}}{\delta x^2}. \tag{2.6}$$

Substituting (2.6) into (2.5) gives the system of equations

$$a\frac{(2u_i - u_{i-1} - u_{i+1})}{\delta x^2} + bu_i = f_i, \quad i = 1, \ldots, N, \tag{2.7}$$

which can be solved for $\{u_i\}_{i=1}^N$, as there are N equations for N unknowns. Equation (2.7) gives the equations to be solved on the interior of the domain. On or close to the boundary the equation to solve may be a little different, and care must be taken that the boundary conditions are treated correctly. For a more detailed discussion on finite difference methods see [166]. The approximation to the derivative given in (2.6) is not unique. Divided differences can be taken using more nodes on the grid to give more accurate approximations to the derivatives. It is possible to approximate derivatives of any order to any degree of accuracy [73] using finite differences on a regular grid. This only makes sense when it is known that high order derivatives exist for the solution to the PDE. Also, the more nodes that are used in an approximation to the derivative the more the boundary effects come into consideration. For complex boundary conditions this can be a problem.

The discretisation of a nonlinear problem, or of a problem that is in two or three dimensions, follows by taking appropriate divided difference formulas and substituting them into the original equations. As can be seen, formulating a finite difference discretisation is simple and a benefit of a finite difference scheme is that it is cheap to set up the system of algebraic equations to solve. However, a large restriction for finite difference schemes is the requirement that the grid used be topologically rectangular. Some simple non-rectangular meshes can be used in conjunction with a finite difference discretisation, so long as the grid may be transformed into a rectangular grid, which is not

generally possible (and rarely easy) in practical applications. Difficulties also arise on or close to boundaries (internal and external) where the geometry is more complicated. The treatment of the boundary conditions then becomes difficult, especially if higher order difference schemes are used.

Finite difference schemes are useful in science and engineering in industrial and research applications (see, amongst others, [6], [63], [75], [82], [151], [163], [192]) and are a powerful tool in obtaining approximations to problems in scientific computing. However, major drawbacks include difficulties in modelling complicated or curved boundaries and also that uniqueness and existence results for solutions require a lot of smoothness of the solution [89]. Finite element methods address some of the shortcomings of finite difference methods. In particular they provide a very powerful framework for the analysis of discrete problems and allow for boundaries to be discretised more easily. Finite element methods are explored in more detail in the next section.

2.3 Finite Element Methods

In order to describe the finite element method the weak formulation of a boundary value problem is introduced.

2.3.1 Weak Formulation

Given a domain $\Omega \subset \mathbb{R}^d$, where $d \in \{1, 2, 3\}$, let $\Gamma^D \subset \partial\Omega$, with $\Gamma^D \neq \emptyset$, be the part of the boundary on which a Dirichlet boundary condition is specified and let $\Gamma^N \subset \partial\Omega$ be the part of the boundary on which a Neumann condition is specified such that $\Gamma^D \cap \Gamma^N = \emptyset$ and $\bar{\Gamma}^D \cup \bar{\Gamma}^N = \Omega$. Let the (linear) boundary value problem to be solved be given by

$$\begin{aligned} -\nabla \cdot [a\nabla u] + bu &= f, \quad \vec{x} \in \Omega, \\ u &= g_D, \quad \vec{x} \in \Gamma^D, \\ \nabla u \cdot \vec{\nu} &= g_N, \quad \vec{x} \in \Gamma^N, \end{aligned} \tag{2.8}$$

where $a(\vec{x}), b(\vec{x}) > 0$, and $\vec{\nu}$ is the outward facing normal on Γ^N. The function u is unknown and is to be solved for, and f is given data. We consider, for ease of discussion, that $g_D \equiv 0$. The case of a non-homogeneous Dirichlet boundary is trivial to deal with, as it is possible to solve for function $\tilde{u} = u - u_D$ where u_D is an extension by zero into the interior of the domain of function g_D. Let v be a function such that $v = 0$ for $\vec{x} \in \Gamma^D$. Instead of looking for the solution of (2.8) directly it is possible to consider the solution

of the *weak formulation*. To form this the first equation in (2.8) is multiplied by a *trial function* v, after which the entire expression is integrated. Making use of the divergence theorem [71, pg. 627] the following is obtained:

$$\int_\Omega a\nabla u \cdot \nabla v \,\mathrm{d}\vec{x} + \int_\Omega buv \,\mathrm{d}\vec{x} = \int_\Omega fv \,\mathrm{d}\vec{x} + \int_{\Gamma^N} ag_N v \,\mathrm{d}S. \tag{2.9}$$

In this case the assumption has been made that the trial function v is zero on Γ^D and has bounded first derivative such that the application of the divergence theorem is possible. Given an appropriate function space $\mathcal{V}$ (in this case $\mathcal{V} \subset H_0^1(\Omega)$, see (2.16)), a function $u \in \mathcal{V}$ is said to be a *weak solution* of (2.8) if it satisfies Equation (2.9) for all $v \in \mathcal{V}$. The known function f is in the *dual space* [23] $\mathcal{V}'$. In the case where $g_N = 0$ on Γ^N in (2.9), it is common to write (2.9) in the dual form given by

$$(u, v)_A = \langle f, v \rangle \tag{2.10}$$

with $\langle v, w \rangle$ the duality pairing between $\mathcal{V}$ and $\mathcal{V}'$, and

$$(v, w)_A \equiv \int_\Omega a\nabla v \nabla w \,\mathrm{d}\vec{x} + \int_\Omega bvw \,\mathrm{d}\vec{x}, \tag{2.11}$$

in the case of Equation (2.9). In the case that $(\,\cdot\,,\,\cdot\,)_A$ defines an inner product (as in (2.11)) the Riesz representation theorem [71, pg. 639] can be used to define a bounded linear operator $A : \mathcal{V} \to \mathcal{V}'$ such that

$$(v, w)_A = \langle Av, w \rangle. \tag{2.12}$$

This notation is useful when discussing general partial differential equations, and will be used in this thesis, excepting the brief discussion given below.

2.3.2 Discretisation of the Weak Formulation

Let $\mathcal{V}$ be a space in which a weak solution to (2.9) is known to exist, and let $V \subset \mathcal{V}$ be a finite dimensional subspace with basis $\{\varphi_i\}_{i=1}^n$ where $n = \dim V$. A function $w \in V$

may then be written as a linear combination of basis functions

$$w = \sum_{i=1}^{n} w_i \varphi_i, \tag{2.13}$$

for $w_i \in \mathbb{R}$. We now are interested in finding an appropriate finite dimensional subspace V of $\mathcal{V}$. To do this we partition the domain Ω into a grid Ω_h with triangulation $\mathcal{T}_h$ and nodes $\mathcal{N}_h$. In the finite element method the basis functions for the set V have compact support on a small subset of neighbouring elements. The definition of the functions can be given in terms of functions over the elements. A standard family of finite element subspaces is given by

$$\mathcal{S}_k(\Omega) = \{\varphi \in C^0(\Omega) \mid \varphi|_T \in P_{k+1}(T),\ \forall\, T \in \mathcal{T}\}, \tag{2.14}$$

where $P_k(T)$ is the space of polynomials of degree k on element T. Letting $\mathcal{N}_h^D$ be the Dirichlet boundary nodes on Ω_h and δ_{ij} the Kronecker delta, a typical basis for the space $\mathcal{S}_k$ is taken from the set

$$\{\varphi_i \in S_k(\Omega) \mid \varphi_i(\vec{x}_j) = \delta_{ij},\ \vec{x}_j \in \mathcal{N}_h / \mathcal{N}_h^D\}. \tag{2.15}$$

An example of a two-dimensional basis function in space $\mathcal{S}_0$ is shown in Figure 2.3.

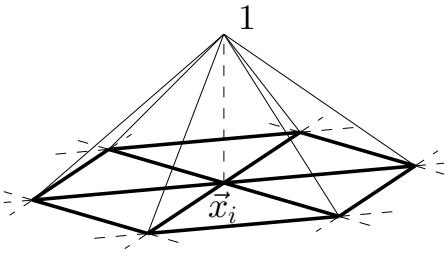


Figure 2.3: Linear basis function $\varphi_i \in \mathcal{S}_0$ centred at node $\vec{x}_i$

A weak solution of (2.9) is known to be unique in the space

$$H_0^1(\Omega) = \{w \mid (w, w) + (\nabla w, \nabla w) < \infty,\ w|_{\Gamma^D} = 0\}\,, \tag{2.16}$$

for $(\,\cdot\,,\,\cdot\,)$ the L_2 inner product on Ω. H_0^1 is the space of functions with L_2 measurable weak first partial derivatives. We note that $\mathcal{S}_0 \subset H_0^1$ [52, Theorem 2.1.1] as S_0 is a linear

combination of piecewise H^1 functions. Therefore Equation (2.9) may be discretised by taking $u, v \in \mathcal{S}_0$ such that $v \in \{\varphi_i\}$ and

$$u = \sum_{i=1}^{n} u_i \varphi_i, \tag{2.17}$$

where $n = \dim \mathcal{S}_0$. This then gives

$$\int_\Omega a \left(\sum_{j=1}^{n} u_j \nabla \varphi_j \right) \nabla \varphi_i \, \mathrm{d}\vec{x} + \int_\Omega b \left(\sum_j u_j \varphi_j \right) \varphi_i \, \mathrm{d}\vec{x} = \int_\Omega f \varphi_i \, \mathrm{d}\vec{x} + \int_{\Gamma^N} a g_N \varphi_i \, \mathrm{d}S, \quad i = 1, \ldots, n. \tag{2.18}$$

Taking the sum outside of the integral, gives

$$\sum_{j=1}^{n} A_{ij} u_j = f_i, \quad i = 1, \ldots, n, \tag{2.19}$$

where

$$A_{ij} = \int_\Omega a \nabla \varphi_i \nabla \varphi_j \, \mathrm{d}\vec{x} + \int_\Omega b \varphi_i \varphi_j \, \mathrm{d}\vec{x},$$
$$f_i = \int_\Omega f \varphi_i \, \mathrm{d}\vec{x} + \int_{\Gamma^N} a g_N \varphi_i \, \mathrm{d}S.$$

This is a system of algebraic equations which can be solved for $[u_i] \in \mathbb{R}^n$, which are the nodal values of the function $u \in \mathcal{S}_0$. The integrals required to calculate A_{ij} and f_i can be performed on an element-by-element basis, counting the contributions to the global linear system of equations. For example, the calculation of A_{ij} is performed as

$$\int_\Omega a \nabla \varphi_i \nabla \varphi_j \, \mathrm{d}\vec{x} + \int_\Omega b \varphi_i \varphi_j \, \mathrm{d}\vec{x} = \sum_{T \in \mathcal{T}_h} \left(\int_T a \nabla \varphi_i \nabla \varphi_j \, \mathrm{d}\vec{x} + \int_T b \varphi_i \varphi_j \, \mathrm{d}\vec{x} \right). \tag{2.20}$$

The integral on the right-hand side is non-zero only for a small number of elements $T \subset \mathrm{supp}(\varphi_i)$. Thus A_{ij} is non-zero for indices j where $\mathrm{supp}(\varphi_i) \cap \mathrm{supp}(\varphi_j) \neq \emptyset$. Noting that Equation (2.19) describes a vector-vector multiplication for $i = 1, \ldots, n$, (2.19) can

be written as

$$A\vec{u} = \vec{f} \tag{2.21}$$

for matrix $A = \left[A_{ij}\right]_{i,j=1}^{n}$, vector $\vec{u} = [u_i]_{i=1}^{n}$ and vector $\vec{f} = [f_i]_{i=1}^{n}$, such that the solution of the discretised formulation (2.18) can be found by inverting the sparse matrix A. There is, in fact, a one-to-one correspondence between linear operators and matrices, which is described more formally in Section 2.5. In this case the matrix A is the matrix representation of the linear operator given in (2.12).

Setting up the linear system of equations to solve requires integration over each element of a grid Ω_h. Depending on the complexity of the operator, and on the number of degrees of freedom defined for a basis function, this can be an expensive operation. This is in contrast to the finite difference method, where the creation of the linear system of equations to solve is fast. An advantage of the finite element method, though, is flexibility in application. In particular it is simple to define an algorithm on an unstructured mesh, once a suitable ordering of the unknowns on the grid has been chosen. This in itself may be a very difficult problem (see [89]).

In this section an example finite element discretisation of a simple linear operator was given in order for the fundamentals of a finite element discretisation to be introduced. Using the same methods outlined above, finite element discretisations can be gained for a nonlinear operator, in which case a system of nonlinear equations is gained. Other basis functions may also be used in a similar manner. For an application of the finite element method to some nonlinear problems see Chapter 7 and 8. In Chapter 7, functions are discretised using $\mathcal{S}_0$, whereas in Chapter 8 the lowest order Raviart-Thomas space RT_0 is also used (see [135, 148]).

For more detail on the finite element method, including sample applications and analysis, the reader is directed to [23], [39], [52], [169] and [208].

Whilst finite element and finite difference methods are very popular, there are also many other discretisation methods, a selection of which are outlined in the next section.

2.4 Other Discretisation Methods

A popular alternative discretisation scheme to finite differences and finite elements is the *finite volume* method. In this method a solution is approximated on a grid using a cell averaged integral. That is, the function is approximated by a constant on an element which is equal to the average of the integral over that element. This approach is very popular for

the solution of hyperbolic problems, such as wave propagation and advective flows [114] due to the fact that they can easily be used to satisfy integral conservation laws, which often arise in the discretisation of hyperbolic PDEs [114]. These methods can also be used in the solution of elliptic and parabolic type PDEs, e.g. [119, 200], although for elliptic and parabolic problems finite element discretisations are much more prevalent. For a detailed introduction to finite volume methods we direct the reader to [114] and [183].

Discretisation methods also exist without the need for a grid. One example is the *collocation* method [95]. In this method *collocation* points are sought in a spatial or temporal domain at which the approximation should satisfy a given PDE. The basis for the space used will give a different discretisation. A popular choice is the space of polynomials, such that solving at m collocation points will give a polynomial of order $m - 1$ [95]. These methods are not used widely to solve problems on a large multi-dimensional domain, and are most widely used in the form of implicit Runge-Kutta schemes [95, §3] for discretising a PDE in time (see [111] for an application).

Another class of discretisation schemes are *spectral* methods, which may or may not make use of a grid. In this family of methods a weak solution is sought, similarly to the case of finite elements. Early variants used infinitely differentiable basis functions defined globally, which are (nearly) orthogonal [47], leading to very sparse systems of equations to solve. In the case that an orthogonal basis is used the resulting system of equations is diagonal [47]. However, the construction of these systems requires the integration of a set of global functions, which can become very expensive for complex nonlinear problems. The time required to set up the system of equations increases with the complexity of the problem also in the case of finite element methods, but the increase is much more severe for spectral methods. More sophisticated (pseudo-)spectral methods are available, and the investigation of these methods is an active research area today. For a thorough introduction to the matter we direct the reader to [47].

2.5 Matrix Representations of Linear Operators

The previous sections describe how to form a discretisation of a given PDE. In the case that a PDE is linear, once a set of basis functions has been chosen, there is a one-to-one correspondence between an operator representation and a matrix representation of the discretisation. This was shown for a specific example in Section 2.3. In this section the equivalence is shown for a general finite element discretisation, as discussions in the rest of the thesis will switch between discrete linear operators and matrices, depending on which is most appropriate in a given context. It is straightforward to obtain the equiv-

alence of discrete linear operators and matrices also in the case of a finite difference or finite volume discretisation.

We consider the general continuous linear operator equation given by

$$\mathcal{A}u = f, \tag{2.22}$$

for $u \in \mathcal{V}$, and its weak form

$$(u, v)_{\mathcal{A}} = (f, v), \quad \forall v \in \mathcal{V}. \tag{2.23}$$

Given a finite dimensional subspace $V \subset \mathcal{V}$ we choose a basis $\{\varphi_i\}_{i=1}^{\#V}$ such that $u \in V$ can be expanded as

$$u = \sum_{i=1}^{\#V} u_i \varphi_i.$$

The discrete weak formulation then reads

$$(u, \varphi_i)_{\mathcal{A}} = \langle f, \varphi_i \rangle, \quad i = 1, \ldots, \#V. \tag{2.24}$$

Using the expansion of u in terms of basis functions and the linearity of the inner product, (2.24) can be written as

$$A\vec{u} = \vec{f} \tag{2.25}$$

with

$$\vec{f} = [f_1, \ldots, f_N]^T, \quad f_i = \langle f, \varphi_i \rangle,$$
$$A = \left[A_{i,j}\right]_{i,j=1}^{N} \quad \text{and} \quad A_{i,j} = (\varphi_j, \varphi_i)_{\mathcal{A}}.$$

Using this equivalence discussions in the rest of the thesis will refer to matrix A as an operator or as a matrix, without confusion.

2.6 Discretising in Time

The discussion so far has been for problems which do not vary over time. In this thesis discussion will also include problems in which the solutions are time-dependent. Since it is not possible to store a solution at every point in time, some discrete time points need to be chosen at which an approximation to the solution is to be obtained. There is a wealth of time-discretisation methods to choose from, but the details of these are outside of the scope of this thesis. For a more complete introduction the reader is directed to [95, §1] and [46, §5].

To demonstrate the application of a common type of discretisation procedure a problem of the form

$$\begin{aligned} \frac{\partial}{\partial t}u &= A(u,t), \\ u &= u^{(0)}, \quad t = t_0, \end{aligned} \tag{2.26}$$

is considered, where A is some operator (linear or nonlinear) that involves only spatial derivatives of $u = u(\vec{x}, t)$. Boundary conditions are not salient in this section and are therefore ignored. Before (2.26) is discretised in space we choose an appropriate discretisation in time. Assume that the solution is known at time t_k, and that the solution at time $t_{k+1} > t_k$ is sought. Let δt be the constant time step such that

$$t_k = t_0 + k\delta t, \qquad u^{(k)} = u(\vec{x}, t_k).$$

It is possible to use variable time-step sizes in an application to improve the accuracy or stability of a method, but for the current introduction only a constant time step is considered. In order to approximate the values of the solution at discrete times (2.26) is integrated between t_k and t_{k+1} to give

$$u^{(k+1)} - u^{(k)} = \int_{t_k}^{t_{k+1}} A(u,t)\,\mathrm{d}t. \tag{2.27}$$

In order to complete the discretisation the right-hand side of (2.27) is approximated using values of the solution at the discrete time-steps. Two simple time discretisation schemes are the explicit Euler and implicit Euler method [46, §5]. The explicit Euler method uses

the approximation

$$A(u,t) \approx A(u^{(k)}, t_k), \quad t \in [t_k, t_{k+1}]$$

to give the scheme

$$u^{(k+1)} - u^{(k)} = \delta t A(u^{(k)}, t_k). \tag{2.28}$$

The implicit Euler method uses the approximation

$$A(u,t) \approx A(u^{(k+1)}, t_{k+1}), \quad t \in [t_k, t_{k+1}]$$

to give the scheme

$$u^{(k+1)} - u^{(k)} = \delta t A(u^{(k+1)}, t_{k+1}). \tag{2.29}$$

Note that the right-hand side of (2.29) depends on the unknown solution at time t_{k+1}, whereas (2.28) depends only on the known values of the solution at time t_k. The term *explicit* in the explicit Euler method refers to the fact that the solution at the unknown time step is determined by known values, and can be calculated by the application of operators at previous time-steps. In an implicit time-stepping procedure, such as the implicit Euler scheme (2.29), the solution of a system of equations at each time-step is required in order to obtain the solution at the latest time. The advantage of implicit time-stepping procedures are that they are often more *stable* than explicit methods [46,92] and therefore permit the use of larger time steps.

Implicit time-stepping procedures are used in this thesis. We consider the implicit Euler method, which is a first order accurate scheme [46, §5]. We also consider a second order scheme, in which the integral in (2.27) is approximated using a trapezium rule to give the scheme

$$u^{(k+1)} - u^{(k)} = \frac{\delta t}{2} \left(A(u^{(k)}, t_k) + A(u^{(k+1)}, t_{k+1}) \right). \tag{2.30}$$

This is referred to as the Crank-Nicolson method in the remainder of this thesis. These are not the most sophisticated methods available, but are sufficient for this work in order to highlight the differences in nonlinear multigrid methods, as introduced in Chapter 4.

For a discussion of more sophisticated time discretisation methods the reader is directed to [46, 77, 95, 112].

In this chapter some of the most widely used methods for the discretisation of PDEs are discussed. It is shown that each discretisation method leads to a system of equations to solve, whether the PDE is linear or nonlinear. The solution of these systems of equations has not been discussed yet, though. There is a wealth of methods available to solve a discretised system. In the next chapter some of these methods are outlined, with particular attention being paid to iterative methods. The focus of this thesis is multigrid methods, which, although they are examples of iterative methods, are left to Chapter 4 to discuss, as these will be treated in more detail.

Chapter 3

The Solution of Discretised Systems of Equations

This chapter gives an introduction to some of the multitude of ways of solving a system of discrete equations. There is no single 'best' solver for a system of equations, as a good solver depends on the properties of the discrete system. As well as introducing different methods, a brief outline into which situations each method is most effective is also given.

In this section the notation

$$Au = f \tag{3.1}$$

is used for a linear system of equations, and

$$A(u) = f \qquad \text{or} \qquad F(u) = 0 \tag{3.2}$$

is used for a nonlinear system of equations where $F(u) = A(u) - f$ for known f. When using A on its own it will be made clear whether this represents a linear or nonlinear operator. The methods introduced in this chapter give some background regarding the different techniques without going into much detail. The solution methods which are of most interest in this thesis are multigrid methods and Chapter 4 is devoted to the introduction of this family of methods.

The chapter is organised as follows. Section 3.1 introduces direct solvers for linear systems of equations. These methods do not exist in the nonlinear case. In Section 3.2 the concepts behind an iterative procedure are introduced in a general setting, which covers the linear and nonlinear case. Explicit examples of linear iterations important for this thesis are introduced in Section 3.3. Finally, in Section 3.4 we introduce the most important nonlinear iteration – Newton's method – as well as giving an introduction to some basic nonlinear iterations which will be useful in the rest of the thesis.

3.1 Direct Solvers for Linear Systems

As mentioned in the introduction a direct solver is only applicable for a linear system of equations

$$Au = f.$$

Direct methods represent the operation of the inverse matrix A^{-1} such that $u = A^{-1}f$ is found. The matrix A^{-1} is not generally explicitly formed [79, pg. 111], but the action of the matrix is sought in an efficient manner. Here we assume that the matrix A is non-singular, such that a unique solution u does exist. Gaussian elimination – which, in effect, involves an LU-decomposition [95, §9], [166, pg. 258] – is applicable to any non-singular matrix, and guarantees to find the answer in $\mathcal{O}(n^3)$ running time, in the case that exact arithmetic is used [79, §3]. In practice problems may arise when a matrix is ill-conditioned, where it is possible for division by small numbers to take place, which can lead to significant rounding errors on a computer [95]. Rows in an ill-conditioned matrix can be swapped (known as row pivoting) to try to reduce the effect of these rounding errors. Since Gaussian elimination is an established algorithm there are many software libraries that efficiently perform Gaussian elimination in serial and parallel. Two popular libraries (amongst many others) are LAPACK [2] and ScaLAPACK [19].

The Gaussian elimination described above does not take advantage of any special properties that a matrix may possess. We note that the discretisation of linear PDEs most often leads to a large sparse matrix that needs to be inverted. In this case there are many operations that need not be calculated as they involve multiplication by zero. There are direct methods that take advantage of this sparsity, and are described in [79, §11.1] and [153, §3.6]. The most popular algorithms used are based upon Cholesky, QR and LU factorisations. Sparse matrices are desirable to be used in practice, as they can be stored on a computer with much reduced storage requirements over a dense matrix. This

is because the zero values, which have no effect on a matrix-vector or matrix-matrix multiplication, need not be stored. However, it is not generally true that a sparse matrix will have a sparse inverse or a sparse factorisation [79, pg. 601]. Care must be taken to order the rows in such a way as to reduce *fill-in*, which is where a zero entry is replaced by a non-zero entry in the process of factorisation. This is not a simple problem. As well as this, since numerical stability of the method needs to be taken into account a reordering of unknowns typically needs to be performed to balance the amount of fill-in along with the stability of the factorisation [79]. For a complete introduction to direct methods for sparse matrices see [56]. For a matrix arising from the discretisation of a PDE on a regular grid the running time of a sparse direct method can be expected to be $\mathcal{O}(n^{3/2})$ ($\mathcal{O}(n^2)$) in two (three) dimensions [56], where n is the number of unknowns on the grid. However, for an unstructured grid where a 'good' ordering of the unknowns is not known, or is difficult to find, the worst case time complexity of the sparse direct solvers is $\mathcal{O}(n^3)$, which quickly becomes impractical for large n. Again there are software libraries that provide efficient implementations of direct methods for sparse systems, such as [57, 62, 129].

As an alternative to direct methods, iterative methods are presented in the next section.

3.2 Iterative Solvers

An iterative method may be applied to a linear or a nonlinear problem. Therefore we consider that the operator A may be either linear or nonlinear in the operator equation

$$A(u) = f. \tag{3.3}$$

Let u^* be the exact solution to (3.3), and u^0 some initial approximation to u^*. The desired output of an iterative method is a sequence of approximations $u^{(i)}$, $i = 1, \ldots$ such that $\lim_{i\to\infty} u^{(i)} = u^*$. Many iterative methods, including those investigated in more detail in this thesis, can be described as the correction procedure

$$u^{(i)} = u^{(i-1)} + \tilde{A}^{(i)} \left(f - A(u^{(i-1)}) \right) \tag{3.4}$$

for some linear operator $\tilde{A}^{(i)}$ which may depend on the current iteration. In the literature it is more common to write the linear iteration as

$$u^{(i)} = \hat{A}^{(i)} u^{(i-1)} + b^{(i)} \tag{3.5}$$

with

$$b^{(i)} = \tilde{A}^{(i)} f,$$
$$\hat{A}^{(i)} = (I - \tilde{A}^{(i)} A).$$

A linear iteration in the form (3.5) is said to be a *stationary linear iteration* if $\hat{A}^{(i)}$ is independent of the iteration number i. This gives an important class of iterations, and will be studied in Section 3.3.

A *fixed point* of an iteration is a function v such that

$$v = v + \tilde{A}^{(i)} \left(f - A(v) \right). \tag{3.6}$$

Inspection of (3.4) shows that the exact solution u^* is a fixed point of the iteration, using $A(u^*) = f$. Assume that the exact solution $u^* \in \mathcal{V}$ and let B_v be a neighbourhood of a function v in $\mathcal{V}$. The function v is said to be a *point of attraction* of an iteration in B_v if, for all $u^0 \in B_v$ it holds that $\lim_{i\to\infty} u^{(i)} = v$. To determine whether u^* is a point of attraction of the iteration (3.4) we require to know more information about the operators $A(\cdot)$ and $\tilde{A}^{(i)}$, which will be discussed for specific algorithms in the next sections. Letting $e^{(i)} = u^* - u^{(i)}$ be the error in approximation at step i, the convergence of a method is characterised by $\lim_{i\to\infty} \left\| e^{(i)} \right\| = 0$ and divergence by $\lim_{i\to\infty} \left\| e^{(i)} \right\| = \infty$. The linear *convergence factor* of an iteration is defined as

$$\lim_{i\to\infty} \frac{\left\| e^{(i)} \right\|}{\| e^{(i-1)} \|}. \tag{3.7}$$

This measures the speed at which an iteration converges if convergence is linear, i.e. in the case that the error at a given iteration is directly proportional to the error at the previous iteration. A convergence factor of zero indicates that an iteration achieves super-linear convergence such that $\left\| e^{(i)} \right\| \propto \left\| e^{(i-1)} \right\|^p$ for $p > 1$.

In order to discuss the concepts introduced here in more detail some popular iterative methods are introduced in the next sections. Section 3.3 outlines classical iterative methods such as Jacobi and Gauss-Seidel [95, §10] and also the powerful Krylov subspace methods [153]. Nonlinear iterations are introduced in Section 3.4, in which it is shown how Newton's method can be used in the definition of classical nonlinear iterations [139]. These concepts will be useful in the definition of multigrid methods in the next chapter.

3.3 Linear Iterative Methods

In the next subsections the classical Jacobi and Gauss-Seidel iterations are introduced. These are most easily described in terms of an operator on a vector $\vec{u}$, whereas the Krylov subspace methods described in Subsection 3.3.3 are most easily described using functions. Using the discussion in Section 2.5 the descriptions are actually equivalent, and the different notation is used to simplify the discussion of the different iterations.

Before describing specific instances of linear iterations we introduce some important results for stationary linear iterations. Consider the stationary linear iteration

$$u^{(i+1)} = \hat{A}u^{(i)} + b. \tag{3.8}$$

The operator $\hat{A}$ is called the error propagation operator, as the error $e^{(i)} = u^{(i)} - u^{(i-1)}$ satisfies

$$e^{(i)} = \hat{A}e^{(i-1)}. \tag{3.9}$$

The following lemma then holds.

Lemma 3.1. *Given an arbitrary linear system $Au = f$ a linear iteration of the form* (3.8) *converges for arbitrary u^0 if and only if $\rho(\hat{A}) < 1$, where $\rho(\cdot)$ denotes the spectral radius. The iterates converge to the solution of the linear system if $b = (I - \hat{A})A^{-1}f$.*

Proof. See [95, Lemma 10.1]. □

It is also possible to say something about the speed of convergence of a stationary linear iteration.

Lemma 3.2. *Let $\hat{A}$ be a full rank square matrix which is the error propagation operator in a convergent stationary linear equation such that $\rho(\hat{A}) < 1$. Then the convergence factor of the iteration is given by $\rho(\hat{A})$.*

Proof. The fact that the spectral radius gives the convergence factor follows from the recursion for the error $e^{(i)} = u^* - u^{(i)}$ given by $e^{(i)} = \hat{A}e^{(i-1)}$ and that there is a unique eigenvalue of largest magnitude. See [166, pg. 270] for a full description. □

Using Lemma 3.1 it can be seen that a convergent linear iteration is given by a choice of

$$\hat{A} = \frac{1}{C}I \tag{3.10}$$

where I is the identity operator, and constant $C > \rho(A)$, assuming that the eigenvalues of A are positive and real. More sophisticated choices for the operator $\hat{A}$ give iterations with some interesting properties. Some of the most important iterations used in the multigrid literature are the linear Jacobi and Gauss-Seidel methods, and weighted versions thereof, as described in the next section.

3.3.1 Jacobi and Gauss-Seidel as Solvers

Let $\vec{u} = [u_1, \ldots, u_n]^T$ for $n > 1$ and $u_j \in \mathbb{R}$ and consider the linear operator $A = [A_1, \ldots, A_n]^T$ where

$$A_j : \mathbb{R}^n \to \mathbb{R}, \quad j = 1, \ldots, n. \tag{3.11}$$

Let the system of equations to be solved be given by

$$A\vec{u} = \vec{f}. \tag{3.12}$$

Each equation in the system is of the form

$$A_j(u_1, \ldots, u_n) = f_j, \quad j = 1, \ldots, n. \tag{3.13}$$

Let $\vec{u}^{(i)}$ be the approximation to the solution after the i^{th} iteration of some iterative method. The Jacobi and Gauss-Seidel iterations can then be defined as follows. For the Jacobi iteration let v be the solution to the equation

$$A_j(u_1^{(i)}, \ldots, u_{j-1}^{(i)}, v, u_{j+1}^{(i)}, \ldots, u_n^{(i)}) = f_j, \tag{3.14}$$

where the $u_j^{(i)}$ values are all known. The iterate $u_j^{(i+1)}$ is given by

$$u_j^{(i+1)} = u_j^{(i)} + \omega(v - u_j^{(i)}), \tag{3.15}$$

for some $\omega \in (0, 1)$. The weighting parameter ω may be used to improve the robustness of the algorithm, and is discussed in [95, 166], although its discussion is not of importance here. Application of Equations (3.14) and (3.15) for each $j = 1, \ldots, n$ gives a single sweep of a Jacobi iteration.

The Gauss-Seidel iteration is defined very similarly, excepting that (3.14) is replaced by

$$A_j(u_1^{(i+1)}, \ldots, u_{j-1}^{(i+1)}, v, u_{j+1}^{(i)}, \ldots, u_n^{(i)}) = f_j, \tag{3.16}$$

where it is assumed that the $u_k^{(i+1)}$ are known for $k < j$. The Gauss-Seidel iteration allows a weighting parameter $\omega \in (0, 2)$ [95, Theorem 10.9] in the update step (3.15) and this weighted version is often called *successive over-relaxation* (SOR). Some textbooks reserve this term for $\omega \in [1, 2)$, but in this thesis we allow $\omega \in (0, 2)$ for SOR. Gauss-Seidel is a special case of SOR where $\omega = 1$.

Another formulation of the Jacobi and Gauss-Seidel iterations which is of use is the matrix representation of the operators. Let $A \in \mathbb{R}^n \times \mathbb{R}^n$ and represent A as

$$A = D - L - U \tag{3.17}$$

where D is the strictly diagonal part of A, and L (U) is strictly lower (upper) triangular. Let S_{J} (S_{SOR}) be the matrix representation of a single linear Jacobi (Gauss-Seidel) iteration. It is possible to show that [166, pg. 266-8] for the Jacobi iteration

$$u^{(i+1)} = u^{(i)} + \omega D^{-1}(f - Au^{(i)}) = S_{\text{J}}u^{(i)} + \omega D^{-1}f, \tag{3.18}$$

and for the SOR iteration

$$u^{(i+1)} = u^{(i)} + (D - \omega L)^{-1}\omega \left(f - Au^{(i)}\right) = S_{\text{SOR}}u^{(i)} + (D - \omega L)^{-1}\omega f. \tag{3.19}$$

This representation of the linear iterations will be useful later in the thesis. The convergence of Jacobi and SOR iterations depend on the properties of matrix A in the linear operator Equation (3.3). We are not interested in the technicalities of the proofs here, but make a note that for a symmetric positive definite matrix A (where the eigenvalues are all necessarily positive and real) which is diagonally dominant then both the Jacobi iteration and the SOR iteration (with $\omega \in (0, 2)$) converge. For a proof see [95, Theorem 10.3] and [95, Theorem 10.9]. This result is important as symmetric positive definite matrices arise often in the discretisation of linear elliptic PDEs. Hence Jacobi and SOR type iterations appear often in the literature. For the interested reader it is possible to formulate these linear iterations for symmetric positive definite operators as subspace correction

methods [165, 196], in which the proofs of the above statements become simple.

The algorithms described above are known as point Jacobi and Gauss-Seidel iterations, as they solve an equation at a single 'point' or entry in the vector. The concepts used to formulate the Jacobi and Gauss-Seidel iterations can easily be carried over to the case that a block iteration instead of a point iteration is to be performed, as is outlined below.

Consider that the matrix $A \in \mathbb{R}^{n\times n}$ is split up into sub-matrices as follows

$$A = \begin{bmatrix} A_{11} & A_{12} & \dots & A_{1p} \\ A_{21} & A_{22} & \dots & A_{2p} \\ \vdots & & \ddots & \vdots \\ A_{p1} & A_{p2} & \dots & A_{pp} \end{bmatrix}, \tag{3.20}$$

where $A_{jk} \in \mathbb{R}^{n_j \times n_k}$. Let the linear system of equations to solve be given by

$$A\vec{u} = \vec{f} \tag{3.21}$$

where $\vec{u} = [\vec{u}_1, \dots, \vec{u}_p]^T$ for $\vec{u}_j \in \mathbb{R}^{n_j}$. Let $\vec{u}^{(i)}$ be the approximation at the i^{th} iteration and let

$$A_j(\vec{u}_1, \dots, \vec{u}_p) \equiv \sum_{k=1}^{p} A_{jk}\vec{u}_k. \tag{3.22}$$

The Jacobi iteration is then formed by solving the linear system of equations

$$A_j(\vec{u}_1^{(i)}, \dots, \vec{u}_{j-1}^{(i)}, \vec{v}, \vec{u}_{j+1}^{(i)}, \dots, \vec{u}_p^{(i)}) = \vec{f}_j \tag{3.23}$$

for $\vec{v}$, and setting

$$\vec{u}_j^{(i+1)} = \vec{u}_j^{(i)} + \omega(\vec{v} - \vec{u}_j^{(i)}). \tag{3.24}$$

The block Gauss-Seidel iteration is formed similarly. Block forms of the Jacobi and Gauss-Seidel iterations may be useful when the matrix is derived from a PDE including strong anisotropies [190].

From Lemma 3.2 we see that the convergence factor of a linear iteration depends on the spectral radius of the underlying matrix. It turns out that this feature makes these

methods unsuitable as solvers for large systems of equations arising from the discretisation of PDEs. As an illustrative example we consider the case that the linear operator comes from the discretisation of a linear elliptic PDE on a grid Ω with quasi-regular triangulation $\mathcal{T}$. As is often the case for elliptic PDEs we assume that the matrix representation of the operator is symmetric positive definite. Let the notation Ω_h and $\mathcal{T}_h$ denote the grid and triangulation where the size of the elements is proportional to h, which is defined in (2.1) to be the largest diameter of an element $T \in \mathcal{T}$. For a finite element discretisation let

$$A_h \vec{u} = \vec{f} \tag{3.25}$$

represent the system of equations arising from discretising on a grid Ω_h. Let $A_{J,h}$ ($A_{GS,h}$) be the error propagation matrix for the Jacobi (Gauss-Seidel) iteration on grid Ω_h (see Equation (3.8)). Explicit forms of these may be found in, for example [95, 139, 166, 196]. It can be shown that for a linear finite element discretisation of a Poisson equation [162, §3.1]

$$\lim_{h \to 0} \rho(A_{J,h}) = 1, \tag{3.26}$$

and similarly for Gauss-Seidel. This behaviour is observed for a large class of elliptic problems, and is typical of a finite element discretisation. In two dimensions it is the case that

$$1 - \rho(A_{J,h}) = \mathcal{O}(h^2). \tag{3.27}$$

Since the grid spacing is inversely proportional to the number of unknowns on a grid the amount of computational effort required to reduce the residual in approximation to a required degree of accuracy is $\mathcal{O}(n^3)$ for these methods. This is the same as for the direct linear methods introduced in the previous chapter, which are unsuitable for large systems of equations.

There are other linear iterations that have better convergence properties than these stationary linear iterations, which are described in later sections. However, in the next section we discuss again the Jacobi and Gauss-Seidel iterations and in particular their use as *smoothers*, where the etymology of this term is made clear in the discussion that follows.

3.3.2 Jacobi and Gauss-Seidel as Smoothers

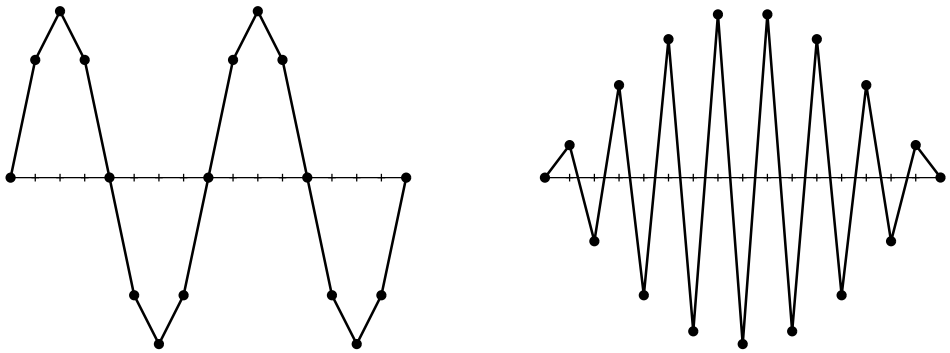
As discussed in the previous section Jacobi and Gauss-Seidel type iterations are not suitable to be used as solvers for systems arising from typical PDE discretisations. However, their use is widespread in practice due to the fact that they act very well as *smoothers* [32, 177], which means that they are good at 'smoothing' the error in an approximation. The meaning of the term 'smooth' will be made clear in the discussion in this subsection. Before beginning the discussion some necessary notation is introduced. Let

$$Au = f \tag{3.28}$$

denote the discrete linear operator equation to solve, and let u^* be the exact solution of (3.28) and let $u^{(i)}$ be the approximation to u^* after i iterations of either Jacobi or SOR, and let $e^{(i)} = u^* - u^{(i)}$ be the error at iteration i. Define a linear operator S called the smoothing operator. Jacobi and Gauss-Seidel iterations are smoothing operators themselves, so that the linear operator S can be identified with the matrix representation of the Jacobi or SOR iterations, as given in (3.18) and (3.19), respectively. These representations will be useful in this thesis and are therefore denoted by S_{J} and S_{SOR}, respectively. Returning to a general smoothing discussion, assume that S has full rank and that its eigenbasis is given by $\{\psi_i\}$, which is partitioned into 'low' and 'high' frequency functions. The error is represented in this eigenbasis as

$$e^{(i)} = \sum_{j=1}^{n} c_j^{(i)} \psi_j \tag{3.29}$$

for $c_j^{(i)} \in \mathbb{R}$ and n the number of nodes on grid Ω. To support the discussion we consider what a 'high' and a 'low' frequency function may look like in a one-dimensional setting. Figure 3.1 gives an example of a high and low frequency function on a grid with $n = 16$. Note that the function in Figure 3.1b is the highest frequency function that can be represented on the grid shown, as it changes sign at every grid point. This demonstrates that the definition of low and high frequency, for a discrete function, will depend on the grid on which a function is represented. As previously mentioned the linear Jacobi and Gauss-Seidel iterations are good smoothers, which means that they are effective at removing high frequency components of the error. For now we simply assume that $\mathcal{I}_{\text{hf}}$ is the index set of basis functions which are effectively removed by a Jacobi (Gauss-Seidel) iteration, and define these to be the high frequency components. It is then expected that



(a) Example of a low frequency function (b) Example of a high frequency function

Figure 3.1: High and low frequency functions on a one-dimensional grid

the error will be well represented by the expansion

$$e^{(i)} \approx \sum_{j \notin \mathcal{I}_{\text{hf}}} c_j^{(i)} \psi_j, \tag{3.30}$$

when $i > C$ for some small integer C, as $c_j^{(i)} \approx 0$ for $j \in \mathcal{I}_{\text{hf}}$. The smoothing property becomes important in the discussion of multigrid iterations in Chapter 4. The *smoothing factor* can be measured as

$$\max_{j \in \mathcal{I}_{\text{hf}}} \{\lambda_j \mid S\psi_j = \lambda_j \psi_j\} \tag{3.31}$$

which is simply the largest eigenvalue associated with a high frequency function [177, pg. 31]. The smaller this value the more effective a smoother is.

We will return to the smoothing operators in Chapter 4 where these become more useful. For now we return to other linear iterations which are popular in practice - the Krylov subspace methods.

3.3.3 Krylov Subspace Methods

Krylov subspace methods can be considered as belonging to various families of iterative methods. They fit into the framework of domain decomposition and subspace correction methods [165, 196] and also into projection methods [153]. The subspace correction framework is usually reserved for describing the methods in the case that an operator is symmetric positive definite. Krylov subspace methods are very powerful and are easy to

implement and are the focus of many active research projects (*e.g.* [65, 106, 122, 145, 174, 209] amongst many others) as well as being very well established in the literature [108, 164]. Before describing the Krylov subspace methods some useful preliminary material is presented below, related to projection methods. The discussion given will follow the outline in [153], although there are many other excellent resources available for Krylov subspace methods (e.g. [69] and [105]).

We consider again the solution to the linear equation

$$Au = f, \tag{3.32}$$

where the exact solution is given by $u^* \in \mathcal{V}$, where $\mathcal{V}$ is a suitable discrete function space. Assume that u^* is unique in $\mathcal{V}$ and for approximation u define the residual as

$$r(u) = f - Au. \tag{3.33}$$

Note that the residual is orthogonal to the space $\mathcal{V}$, i.e.

$$(f - Au^*, v) = 0, \quad \forall v \in \mathcal{V}. \tag{3.34}$$

The inner product $(\,\cdot\,,\,\cdot\,)$ is taken to be the L^2 inner product. The solution in space $\mathcal{V}$ with $\dim \mathcal{V} = N$ for large N can be difficult to obtain. It is easier to approximately solve (3.32) on a smaller space $\mathcal{U} \subset \mathcal{V}$ under the condition that the residual be orthogonal to some space $\mathcal{W} \subset \mathcal{V}$, i.e.

$$(f - A\tilde{u}, w) = 0, \quad \forall w \in \mathcal{W}, \tag{3.35}$$

with $\tilde{u} \in \mathcal{U}$. Equation (3.35) permits a solution $\tilde{u} \in \mathcal{U}$. However, it may be useful to make use of an initial approximation $u^0 \in \mathcal{V}$ to u^*, in which case we write

$$\tilde{u} = u^0 + \delta \tag{3.36}$$

for $\delta \in \mathcal{U}$. Using that

$$r^0 = f - Au^0 \tag{3.37}$$

Equation (3.35) can be written as

$$(r^0 - A\delta, w) = 0, \quad \forall w \in \mathcal{W}. \tag{3.38}$$

The condition (3.38) (equivalently (3.35)) is known as a Petrov-Galerkin condition [153, §5] in the case that $\mathcal{U} \neq \mathcal{W}$ and a Galerkin condition otherwise. A projection method based on the Galerkin condition is called an *orthogonal* projection method, and a projection based on a Petrov-Galerkin condition is called an *oblique* projection method. Two optimality results are introduced that will be useful in the discussion of Krylov subspace methods, which first require the following definition.

Defintion 3.1. *Let A be a symmetric positive definite matrix. The A-*norm *is given by*

$$\|u\|_A^2 = (u, u)_A = (Au, u). \tag{3.39}$$

Using this definition of the A-norm the following holds true

Proposition 3.3. *Assume that A is symmetric positive definite and that $\mathcal{U} = \mathcal{W}$. A function $\tilde{u}$ is the result of an orthogonal projection method onto $\mathcal{U}$ with starting value u^0 if and only if it minimises the A-norm of the error over $u^0 + \mathcal{U}$, i.e.*

$$\|u^* - \tilde{u}\|_A = \min_{u \in u^0 + \mathcal{U}} \|u^* - u\|_A . \tag{3.40}$$

Proof. See [153, Proposition 5.2]. □

There is also an optimality result for oblique projections. Using the notation $A\mathcal{U}$ to denote the range of the operator A with domain in $\mathcal{U}$ this is stated as:

Proposition 3.4. *Let A be an arbitrary discrete linear operator, and let $\mathcal{W} = A\mathcal{U}$. A function $\tilde{u}$ is the result of an oblique projection method onto $\mathcal{U}$ orthogonally to $\mathcal{W}$ with initial value u^0 if and only if it minimises the L_2 norm of the residual over $u^0 + \mathcal{U}$, i.e.*

$$\|f - A\tilde{u}\|_2 = \min_{u \in u^0 + \mathcal{U}} \|f - Au\|_2 \tag{3.41}$$

Proof. See [153, Proposition 5.3]. □

Projection methods are widespread and an important class of these are the one-dimensional projection methods. That is, we have that $\dim \mathcal{U} = \dim \mathcal{W} = 1$. Jacobi and Gauss-Seidel iteration methods fit into the framework of one-dimensional projection methods. Consider that a problem has been discretised on grid Ω_h with nodes $\mathcal{N}_h$ where $\#\mathcal{N}_h = N$. We then define a set of subspaces

$$\mathcal{U}_j = \operatorname{span}\{\varphi_j\}, \quad j = 1, \ldots, N \tag{3.42}$$

with φ_j (for example) a nodal basis function in space $\mathcal{S}_0$ as shown in Figure 2.3. Letting $\mathcal{W}_j = \mathcal{U}_j$ and $u^{(i)}$ be the approximation after the i^{th} Gauss-Seidel iteration, a single sweep of the Gauss-Seidel iteration is defined as

$$u^{(i+j/N)} = u^{(i+(j-1)/N)} + \delta_j, \quad j = 1, \ldots, N, \tag{3.43}$$

where $\delta_j \in \mathcal{U}_j$ satisfies the orthogonality condition

$$(r^{(i+(j-1)/N)} - A\delta_j, w) = 0, \quad \forall w \in \mathcal{W}_j. \tag{3.44}$$

The Jacobi iteration is defined by replacing $r^{(i+(j-1)/N)}$ with $r^{(i)}$ in (3.44).

More sophisticated choices of one-dimensional subspaces $\mathcal{U}$ and $\mathcal{W}$ give iterations with improved convergence results over Gauss-Seidel and Jacobi iterations. However, we are more interested in the case where the subspaces are multidimensional. In particular we are interested in *Krylov* subspaces. Letting u^0 be the initial approximation to the solution of problem (3.32) the corresponding Krylov subspaces are defined as [153]

$$\mathcal{K}_m = \operatorname{span}\left\{r^0, Ar^0, A^2r^0, \ldots, A^{m-1}r^0\right\}, \quad m = 1, 2, 3, \ldots \tag{3.45}$$

for $r^0 = f - Au^0$. In the m^{th} step of a Krylov subspace method the problem is projected in to the space $\mathcal{K}_m$ and a solution is sought there using an orthogonal or oblique projection. Below we outline two of the most popular Krylov subspace methods - Generalised Minimum RESidual (GMRES) and Conjugate Gradients (CG). There are other Krylov subspace methods, an introduction to which is given in [153, §6]. It is worth noting that as a result of the Cayley-Hamilton theorem [69] and the optimality result in Proposition 3.3 (Proposition 3.4) a CG (GMRES) method is guaranteed (neglecting rounding errors) to give the exact solution in $n = \dim A$ steps. In practice it is not practical to perform

this many steps, though, and Krylov subspace methods also give good approximations in many situations with $m \ll n$.

An important process necessary for Krylov subspace methods is the construction of an orthonormal basis. This is done using Arnoldi's method [153, §6.3], in which some orthogonalisation procedure is performed. Popular choices are Gram-Schmidt [168], Householder [153] and Lanczos [153] orthogonalisation. Householder orthogonalisation is numerically more stable than Gram-Schmidt, but is algebraically equivalent. For brevity only the Arnoldi-Gram-Schmidt method is presented, as the implementation is faster. Lanczos orthogonalisation is used when an operator is symmetric positive definite and is used in conjunction with CG.

Although the GMRES algorithm is the more general solver it is more natural to begin with the description of this method before briefly describing the CG algorithm. The GMRES algorithm will be used more extensively in this thesis and so is described in more detail.

3.3.3.1 GMRES

GMRES is an oblique projection method from the space $\mathcal{K}_m$ to $A\mathcal{K}_m = \mathcal{K}_{m+1}$. We first give the algorithm and then describe what is being performed at each step. Let u^0 be the initial approximation to the solution of Equation (3.32), and let e_1 be the first canonical basis vector. The GMRES iteration is then given in Algorithm 3.1 (see [153, Algorithm 6.9]). Lines 4 to 6 perform the orthogonalisation of the function $w_j \in A\mathcal{K}_j =$

Algorithm 3.1 GMRES

Require: The number of steps to perform $m \leq n$

1: $r^0 \leftarrow f - Au^0$, $\beta \leftarrow \|r^0\|_2$, $v_1 \leftarrow r^0/\beta$
2: **for** $j = 1, 2, \ldots, m$ **do**
3: $\quad w_j \leftarrow Av_j$
4: $\quad$ **for** $i = 1, \ldots, j$ **do** ▷ *Orthogonalise using modified Gram-Schmidt*
5: $\quad\quad h_{ij} \leftarrow (w_j, v_i)$
6: $\quad\quad w_j \leftarrow w_j - h_{ij}v_i$
7: $\quad h_{j+1,j} \leftarrow \|w_j\|_2$
8: $\quad v_{j+1} \leftarrow w_j/h_{j+1,j}$
9: Set $H_m \leftarrow [h_{ij}]_{1\leq i\leq m+1,\ 1\leq j\leq m}$
10: Set y_m to minimise $\|\beta e_1 - H_m y\|_2$ ▷ *Solve least squares problem* $H_m y = \beta e_1$
11: $\tilde{u} \leftarrow u^0 + V_m y_m$

$\mathcal{K}_{j+1}$ with respect to the spaces $\mathcal{K}_i$, $i \leq j$. Hence the v_j, $j = 1, \ldots, m+1$ will be orthonormal vectors, and V_m (being the matrix of column vectors v_j, $j = 1, \ldots, m$) is

also orthonormal. The entries in the Hessenberg matrix H_m are set in lines 5 and 7. In line 10 an $(m+1) \times m$ least squares problem is solved, which will be a small system provided that the number of steps m taken is small. It is assumed that $w_j \neq 0$ in Algorithm 3.1 such that the least squares problem in line 10 is well defined. The case that $w_j = 0$ is not a problem, and in fact if this condition arises the GMRES iterate u^j will be the exact solution of (3.32) [153, Proposition 6.10]. Under the assumption that $w_j \neq 0$ for $j = 1, \ldots, m$ the following lemma (which is adapted from arguments given in [153, §6]) gives an optimal result for the GMRES iteration:

Lemma 3.5. *Let $\tilde{u} = u^0 + V_m y_m$ be the approximation gained from taking m steps of the GMRES iteration from initial guess u^0 to problem* (3.32)*. Then*

$$\tilde{u} = \underset{u \in u^0 + \mathcal{K}_m}{\arg\min} \|f - Au\|_2 . \tag{3.46}$$

Proof. From lines 3-6 of Algorithm 3.1 we see that

$$w_j = Av_j - \sum_{i=1}^{j} h_{ij} v_i. \tag{3.47}$$

Combining this with the representation of v_{j+1} in line 8 we get

$$\begin{aligned} Av_j &= \sum_{i=1}^{j+1} h_{ij} v_i \\ &= V_{j+1} H_{j+1,j} \end{aligned} \tag{3.48}$$

where V_{j+1} is the matrix of the column vectors v_i, $i = 1, \ldots, j+1$, and $H_{j+1,j}$ is the j^{th} column of the Hessenberg matrix H_{j+1}, as defined in Algorithm 3.1. Since (3.48) holds for all $j = 1, \ldots, m$ it follows that

$$AV_m = V_{m+1} H_m. \tag{3.49}$$

Using the representation $\tilde{u} = u^0 + V_m y_m$ we get

$$\begin{aligned} f - A\tilde{u} &= f - A(u^0 + V_m y_m) \\ &= r^0 - AV_m y_m \\ &= r^0 - V_{m+1} H_m y_m \\ &= V_{m+1} \left(\beta e_1 - H_m y_m\right). \end{aligned} \tag{3.50}$$

Using the fact that V_{m+1} is orthonormal and that $v_1 = r_0/\beta$ the equality

$$\|f - A\tilde{u}\|_2 = \|\beta e_1 - H_m y_m\|_2 \tag{3.51}$$

follows. As V_m is invertible $\tilde{u} \in u^0 + \mathcal{K}_m$ is identified by y_m, so that if y_m minimises the right-hand side of (3.51) then $\tilde{u}$ minimises the left hand side. This holds since y_m is the solution of the least squares problem $H_m y = \beta e_1$. □

The convergence analysis of GMRES is too involved to include in this thesis, and a general approximation of the convergence is only available in the case that the matrix A is diagonalisable. We note, however, that in practice the GMRES iteration converges similarly to the CG iteration, where the convergence properties of CG are described in the next subsection. A complete discussion of the convergence of GMRES can be found in [153, §6.11.4]. There are implementational issues that may arise, which have not been discussed here. For a full discussion regarding a robust and efficient implementation of GMRES the reader is directed to [153, §6.5.3].

3.3.3.2 CG

The conjugate gradient method is an orthogonal projection method defined on the Krylov subspace $\mathcal{K}_m$, but is defined only for symmetric positive definite operators. The concepts used are very similar to those of the GMRES iteration, in that an orthogonalisation step is performed to create an orthonormal basis, and the minimiser is found by solving an equivalent (small) linear system of equations. Using properties specific to symmetric positive definite matrices the CG method obtains the solution without the need to store the basis vectors, as is the case for GMRES, and it inverts the small linear system of equations implicitly in the construction. The method is therefore less memory intensive and computationally less expensive than GMRES for symmetric positive definite matrices. For a description of the algorithm see [153, §6.7]. Let u^m be the approximation obtained from

performing m steps of the CG algorithm (see [153, Algorithm 6.18]), then

$$u^m = \underset{u \in u^0 + \mathcal{K}_m}{\arg\min} \|u^* - u\|_A \,. \tag{3.52}$$

The proof of this statement follows from the formulation of the CG algorithm and the optimality result in Proposition 3.3. A full discussion is given in [153, §6.7].

The convergence factor of the CG iteration is given by a relation of the error before and after m steps of the CG iteration are performed as

$$\|u^* - u^m\|_A \leq 2 \left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} \right)^m \left\|u^* - u^0\right\|_A , \tag{3.53}$$

where

$$\kappa = \frac{\lambda_{\max}}{\lambda_{\min}} \tag{3.54}$$

is the spectral condition number of symmetric positive definite operator A. We consider what this means for practical problems. Consider the case that A_h is symmetric positive definite and represents the discretisation of a second order linear elliptic PDE on a grid Ω_h, where h is the characteristic grid spacing on grid Ω_h. Let $n = \#\mathcal{N}_h$ be the number of unknown points on grid Ω_h. In two dimensions the spectral condition number κ_h of the matrix A_h typically grows like [162]

$$\kappa_h = \mathcal{O}(n^2). \tag{3.55}$$

Hence, the convergence factor of CG will be $\mathcal{O}(n)$, and the overall running time of the algorithm will be $\mathcal{O}(n^2)$ to reach a desired accuracy. This is a large improvement over the direct methods and simple iterative solvers that we have encountered previously, but is still not optimal and the running times for large problems soon become problematic.

Although GMRES and (in particular) CG have a low cost per iteration the convergence of the methods depends on the spectrum of the operator A. As mentioned above, for many physical problems the spectrum of the arising discrete linear operator becomes more ill-conditioned as the grid resolution is increased. What this means is that the eigenvalues are not clustered into distinct sets as the dimension n increases. The reason why it is important to have the eigenvalues clustered into distinct sets is a technical result which is

part of the convergence analysis of Krylov subspace methods. The result is outside of the scope of this thesis, but the interested reader may see [153, §6.11.3] for a justification.

If a well-conditioned matrix is used in a GMRES or CG iteration the convergence factor will be of order $\mathcal{O}(1)$ with respect to the number of unknowns in the system. In order to take advantage of this fact it is possible to transform a problem into an equivalent one, where the matrix in the transformed problem is better conditioned. This transformation process is commonly known as preconditioning, and is discussed in the next section.

3.3.4 Preconditioning

Consider again the linear system (3.32). Preconditioning transforms the system of equations to solve into an equivalent system which has the same solution, but the properties of the operators are (ideally) better suited to an iterative algorithm. Preconditioning can be used to improve the robustness and/or efficiency of an iterative method, and is popular in practice in conjunction with Krylov subspace methods, *e.g.* [64, 125, 158, 182] amongst many others. Instead of the system (3.32) we consider the transformed system

$$P_L^{-1} A P_R^{-1} P_R u = P_L^{-1} f, \tag{3.56}$$

where the matrix P_L is known as a left preconditioner and P_R is known as a right preconditioner. It is possible to perform both left and right preconditioning, although it is more common to perform one or the other. In this thesis we consider right preconditioning only, as the norm in which a residual is calculated remains unchanged [153, Proposition 9.1]. The right preconditioned system is written as

$$A P_R^{-1} P_R u = f. \tag{3.57}$$

The problem

$$A P_R^{-1} v = f \tag{3.58}$$

is solved, where $u = P_R^{-1} v$. Clearly P_R should be invertible. As mentioned previously, a problem with CG and GMRES iterations is that the convergence of the methods depends on the spectrum of the operator A. We say that a matrix A is poorly-conditioned if the spectrum is not bounded or if the eigenvalues are not clustered into groups. It is well known that for many discretisations of PDEs the conditioning of the arising matrix dete-

riorates as the resolution on a grid is increased. It is this deterioration in the conditioning that we seek to eliminate through the use of a preconditioner. To be considered a good preconditioner the operator P_R should satisfy the following conditions

I The matrix P_R should be a good approximation to A and the eigenvalues of AP_R^{-1} should be bounded away from zero and infinity and should be grouped into distinct small sets.

II The action of P_R^{-1} should be easy to compute, *i.e*. the memory and computational time required to invert P_R should be small.

The discussion of what makes a good preconditioner is outside of the scope of this thesis other than in the case of multigrid methods, which are discussed separately. An effective preconditioner is often tailored to a specific application and the formulation and analysis of preconditioners is an active research topic (see, for example, [14, 68, 115, 170, 194]). We do note that linear multigrid methods are widely used in many applications as preconditioners for Krylov subspace methods. This approach has enjoyed success in many application areas, the reasons for which should be made clear in the discussion of linear multigrid methods in Chapter 4.

As an example of how to precondition a Krylov subspace method we consider the case in which preconditioning is used in conjunction with a GMRES iteration. As before right preconditioning is used. Algorithm 3.2 gives a realisation of a right preconditioned GMRES iteration. This variant of GMRES is known as flexible GMRES (FGMRES)

Algorithm 3.2 Flexible GMRES (FGMRES)

Require: Number of steps m to perform ; Preconditioners $\{M_j\}$, $j = 1, \ldots, m$

1: $r^0 \leftarrow f - Au^0$, $\beta \leftarrow \|r^0\|_2$, $v_1 \leftarrow r^0/\beta$
2: **for** $j = 1, \ldots, m$ **do**
3: $\quad z_j \leftarrow M_j^{-1} v_j$ ▷ *Perform preconditioning*
4: $\quad w \leftarrow Az_j$
5: $\quad$ **for** $i = 1, \ldots, j$ **do**
6: $\quad\quad h_{i,j} \leftarrow (w, v_i)$ ▷ *Orthogonalise using modified Gram-Schmidt*
7: $\quad\quad w \leftarrow w - h_{i,j} v_i$
8: $\quad h_{j+1,j} = \|w\|_2$
9: $\quad v_{j+1} = w/h_{j+1,j}$
10: Set $Z_m \leftarrow [z_1, \ldots, z_m]$
11: Set $H_m \leftarrow [h_{ij}]_{1 \le i \le m+1,\ 1 \le j \le m}$
12: Set y_m to minimise $\|\beta e_1 - H_m y\|_2$ ▷ *Solve least squares problem* $H_m y = \beta e_1$
13: Set $x_m = x_0 + Z_m y_m$

as it is noted that in line 3 the preconditioner used may depend on the current step in

the iteration. This allows for the preconditioning operator to change in each GMRES iteration. In a practical application the preconditioning matrix M_j is often not available, or it is not desirable to explicitly form this operator. Instead the effect of M_j is performed by implicitly applying the preconditioning operator, which is often an iterative procedure (such as Gauss-Seidel or Jacobi) applied to the system

$$Az_j = v_j. \tag{3.59}$$

Applying an iterative process a number of times to Equation (3.59) is equivalent to applying the inverse of the preconditioning operator in line 3 of Algorithm 3.2. Since the preconditioner depends on the right-hand side (in this case v_j) the preconditioner is likely to be different in each iteration. Therefore FGMRES is no longer a Krylov subspace method, but we do have the following optimality result:

Proposition 3.6. *Let Z_m be defined as in line 10 of Algorithm 3.2. Then, after m FGMRES iterations, the approximation $\tilde{u} = u^0 + Z_m y_m$ satisfies*

$$\tilde{u} = \underset{u \in u^0 + \mathrm{span}\{Z_m\}}{\arg\min} \|f - Au\|_2 \tag{3.60}$$

when Z_m is a full rank matrix.

Proof. A proof of the above result can be found in [153, Proposition 9.2]. □

FGMRES requires the extra storage of the vectors z_j, $j = 1, \ldots, m$. In the case when multigrid is used as a preconditioner the extra storage requirements are usually modest as m is typically small. The convergence properties of FGMRES are beyond the scope of this thesis, but the same principle applies as in the case of GMRES - so long as the matrices AM_j^{-1} are well-conditioned then the convergence of FGMRES will be independent of the spectrum of the matrix A.

The introduction to linear iterations in this section is far from complete, and we direct the reader to the extensive literature on this area (*e.g.* [23, 69, 153] and references therein). The concepts introduced here are sufficient to follow the rest of this thesis, and many of the ideas introduced here are used in nonlinear and multigrid iterations. Nonlinear iterations are covered in the next Section 3.4 and multigrid methods are described in more detail in Chapter 4.

3.4 Nonlinear Iterations

In this section the solution of nonlinear equations is considered. A nonlinear operator F is characterised by the fact that the output does not scale with the input, *i.e.* that in general the condition

$$F(\alpha u + \beta v) \neq \alpha F(u) + \beta F(v) \tag{3.61}$$

holds true for nonlinear operator F. As is standard in the literature the general nonlinear equation given by

$$F(u) = 0 \tag{3.62}$$

is considered, where

$$F : \mathcal{V} \to \mathcal{W} \quad \text{and} \quad u \in \mathcal{V}.$$

There exists no general direct solver for a nonlinear problem, as the properties of the operator F depend also on the current approximation u. Assuming that there exists a $u^* \in \mathcal{V}$ that satisfies Equation (3.62) we are then left to find an iterative procedure to move from an initial approximation u^0 to the solution u^*. In general u^* may not be unique in $\mathcal{V}$, but we do assume that it is unique in the open ball

$$B_{u^*}(\delta) \equiv \{u \in \mathcal{V} \mid \|u - u^*\| < \delta\} \tag{3.63}$$

for some suitable norm $\|\cdot\|$. Clearly $B_{u^*}(\delta)$ is a subset of $\mathcal{V}$. If u^* is not unique in $\mathcal{V}$ then F is in general not invertible. However, it is possible to find subsets of $\mathcal{V}$ on which F is invertible. The first assumption that we make is that F is *differentiable* at the solution u^*. There are two definitions of differentiability of which we will make use in this thesis - Gateaux and Frèchet differentiability. The Gateaux derivative of F at $u \in \mathcal{V}$ is a linear operator $F'(u) \in \mathcal{V} \to \mathcal{W}$ such that

$$\lim_{h \to 0} \frac{\|F(u + hv) - F(u) - hF'(u)(v)\|}{h} = 0 \tag{3.64}$$

for all $v \in \mathcal{V}$. The Frèchet derivative of F at $u \in \mathcal{V}$ is defined by a linear operator $F'(u) : \mathcal{V} \to \mathcal{W}$ satisfying

$$\lim_{\|h\| \to 0} \frac{\|F(u+h) - F(u) - F'(u)(h)\|}{\|h\|} = 0 \tag{3.65}$$

for all $h \in \mathcal{V}$. In the remainder of this thesis we assume that a nonlinear operator is Frèchet differentiable and note that whenever a function is Frèchet differentiable that it is also Gateaux differentiable [139, §3]. The Gateaux derivative will be explicitly used in the formulation of a variant of nonlinear multigrid in Chapter 4. We note that, in the case that $F(u) = 0$, Fréchet differentiability (3.65) immediately implies

$$F(u+h) - F'(u)(h) = o(h), \quad \text{as } \|h\| \to 0, \tag{3.66}$$

such that as the perturbation becomes smaller the linearisation approximates the action of the nonlinear operator locally. This is an important property, especially for Newton's method, which is given in the next subsection. In the remainder of this thesis we assume that a nonlinear operator is continuously Fréchet differentiable. We may then make use of the following proposition.

Proposition 3.7. *Assume that a nonlinear operator $F : \mathcal{V} \to \mathcal{W}$ is continuously Frèchet differentiable at $u^* \in \mathcal{V}$. Then there exists a δ such that F is a homeomorphism (*i.e. *F and F^{-1} are continuous) on $B_{u^*}(\delta)$. If F' exists and is continuous on $B_{u^*}(\delta)$ then F'^{-1} exists and is continuous.*

Proof. This is the inverse mapping theorem, a full statement and proof of which is given in [139, 5.2.1]. □

Assuming that u^* is unique and F is invertible in $B_{u^*}(\delta)$ we are interested in finding an iterative method of the form

$$u^{(i+1)} = u^{(i)} - \left(A^{(i)}\right)^{-1} F(u^{(i)}) \tag{3.67}$$

for operators $A^{(i)}$ (which may or may not depend on the iteration count i) with the property that

$$u^{(i)} \in B_{u^*}(\delta),\ i = 1, 2, 3, \ldots \qquad \text{and} \qquad \lim_{i \to \infty} u^{(i)} = u^*. \tag{3.68}$$

It is very common to let $A^{(i)}$ be some linear operator, which is much easier to find and apply than a nonlinear operator. There are many choices for the linear operator. We start by describing the most popular choice - taking the derivative of F at $u^{(i)}$. This gives rise to Newton's method, which is introduced in more detail below.

3.4.1 Newton's Method

Consider that we are solving Equation (3.62). For approximation $u = u^* - \delta$ a Taylor expansion [15, Theorem 2.1.33] neglecting higher order terms reads

$$F(u+\delta) = F(u) + F'(u)(\delta) + \mathcal{O}(\delta^2) \tag{3.69}$$

which gives rise to the approximation

$$\delta \approx -F'(u)^{-1}F(u) \tag{3.70}$$

and hence the Newton iteration given in Algorithm 3.3.

Algorithm 3.3 Newton's Method

$$u^{(i+1)} = u^{(i)} - F'(u^{(i)})^{-1}F(u^{(i)})$$

Newton's method is one of the first iterative methods conceived. It is very well established, a brief history of which can be found in [59]. The modern formulation, as given in Algorithm 3.3, for general operators on Banach spaces dates back to Kantorovich [103] and Mysovskikh [132]. The method is very popular due to its local *quadratic* convergence properties, which are hinted at in the Taylor expansion (3.69) where the error term is proportional to the square of the correction. The monograph [58] splits Newton methods into different classes. For the purposes of this thesis we are interested in the class of *affine contravariant* methods [58], which use the residual norm of an approximation to control the termination of an iteration.

A Newton iteration for (3.62) will converge in the residual norm under the following conditions:

Theorem 3.8. *Let $F : \mathcal{V} \to \mathcal{W}$ be Frèchet differentiable on open and convex $\mathcal{V}$, and assume that $F'(u)$ is invertible for all $u \in \mathcal{V}$. Assume that the Lipschitz condition*

$$\|(F'(u) - F'(v))(u-v)\| \leq \omega \|F'(v)(u-v)\|^2, \quad u, v \in \mathcal{V} \tag{3.71}$$

holds for some positive $\omega \in \mathbb{R}$*. Define*

$$\mathcal{L}_\omega \equiv \{u \in \mathcal{V} \mid \|F(u)\| < 2/\omega\} \tag{3.72}$$

and assume $\bar{\mathcal{L}}_\omega$ *is bounded. Choose* u^0 *to be in* $\mathcal{L}_\omega$*. Then the Newton iterates* $\{u^{(i)}\}$ *obtained from Algorithm 3.3 converge to a fixed point* $u^* \in \mathcal{L}_\omega$ *with* $F(u^*) = 0$*. The residuals* $\{F(u^{(i)}\}$ *are related by*

$$\left\|F(u^{(i+1)})\right\| \leq \frac{1}{2}\omega \left\|F(u^{(i)})\right\|^2. \tag{3.73}$$

Proof. See the proof of [58, Theorem 2.12]. □

The above result states that given a differentiable and invertible nonlinear C^1 function F the Newton iteration will converge, given that an initial guess is 'good enough'. This type of result is known as a *local* convergence estimate, and it depends on the quality of the initial estimate. For the implicit solution of initial value problems such as parabolic PDEs, for example, the initial condition is known, and successive approximations can be chosen 'close enough' to the solution at the next time step by controlling the size of the time stepping parameter. However, for boundary value problems, where an initial approximation is often not given, it may be difficult to find a good initial estimate. It may also be that an initial approximation is outside of a ball of guaranteed convergence, so that Algorithm 3.3 diverges. In this case a *global* Newton method may be more appropriate, as given in Algorithm 3.4. The only difference in this algorithm is that there is a damping

Algorithm 3.4 Global Newton Method

$$u^{(i+1)} = u^{(i)} - \gamma^{(i)} F'(u^{(i)})^{-1} F(u^{(i)})$$

factor $\gamma^{(i)}$ that should be chosen such that

$$\left\|F(u^{(i+1)})\right\| \leq \left\|F(u^{(i)})\right\|. \tag{3.74}$$

If $\lim_{i\to\infty} \gamma^{(i)} = 1$ then quadratic convergence will be achieved for $i \geq n$ for some $n \in \mathbb{N}$.

Note that Algorithms 3.3 and 3.4 require the inversion of the derivative $F'(u^{(i)})$ in each step. The derivative is a linear operator, and in a finite-dimensional setting this is equivalent to inverting a matrix, known as the *Jacobian* matrix. Assuming that $\dim \mathcal{V} = N$ and $\{\varphi_i\}_{i=1}^N$ is a basis for $\mathcal{V}$, then we identify $u \in \mathcal{V}$ with $\vec{u} = [u_1, \ldots, u_N]^T$, as

in Section 2.5. It is assumed that a finite element discretisation is used, and that $F(u)$ represents a system of nonlinear equations posed in a weak form (see Subsection 2.3.2). That is, there exists a function $\tilde{F} : \mathcal{V} \to \mathcal{V}'$ such that in the finite dimensional setting $F(u)$ can be written as the vector of dual pairings

$$F(u) = [\langle \tilde{F}(u), \varphi_j \rangle]_{j=1}^{N}.$$

Denoting $F_j = \langle \tilde{F}(u), \varphi_j \rangle$ the Jacobian matrix is then given by

$$F'(u) = \begin{bmatrix} \frac{\partial F_1}{\partial u_1} & \frac{\partial F_1}{\partial u_2} & \cdots & \frac{\partial F_1}{\partial u_N} \\ \frac{\partial F_2}{\partial u_1} & \frac{\partial F_2}{\partial u_2} & \cdots & \frac{\partial F_2}{\partial u_N} \\ \vdots & & \ddots & \vdots \\ \frac{\partial F_N}{\partial u_1} & \frac{\partial F_N}{\partial u_2} & \cdots & \frac{\partial F_N}{\partial u_N} \end{bmatrix}. \tag{3.75}$$

If F is obtained through the discretisation of a partial differential equation it is often the case that the Jacobian matrix is sparse. Therefore, sparse direct solvers may be applied in order to obtain a standard Newton method that executes in less time than using a dense solver. However, it is possible to perform an inexact solve of the linear system to obtain an *inexact* Newton method, as given in Algorithm 3.5. In this algorithm it is required

Algorithm 3.5 Inexact Newton Method

$$\delta = (A^{(i)})^{-1} F(u^{(i)})$$
$$u^{(i+1)} = u^{(i)} - \delta$$

that $A^{(i)} \approx F'(u^{(i)})$. Convergence results similar to Theorem 3.8 are available for inexact Newton methods, assuming that the linear system of equations is solved accurately enough (see [58, Theorem 2.17] for a proof of convergence of a Newton-GMRES iteration). The exact definition of 'sufficient accuracy' will depend on the problem being solved. The Newton iteration here is termed the *outer* iteration, and the linear iterative procedure used to find an approximate update is termed the *inner* iteration.

Another important result for Newton methods is that of mesh independent convergence. That is, for a discrete problem arising from a PDE, the convergence of a Newton method can be shown to be independent of a mesh on which a problem is discretised for sufficiently fine meshes. The theory for mesh independent convergence is most developed in the case of elliptic problems in which the derivative F' is positive definite [58, §8], [61], although analysis does exist for other problems [45, 104, 107]. Experimentally mesh inde-

pendent convergence can be observed for much more complex problems than the theory covers (*e.g.* [13, 99, 100, 108, 184] and many others).

This combination of mesh independent and local quadratic convergence make Newton methods very popular and widely used. There is another family of nonlinear iterations which also displays a mesh independent convergence property, and performs very similarly to Newton's method when applied to elliptic model problems - Nonlinear Multigrid Methods. These are discussed in more detail in Chapter 4. Before moving on to the nonlinear multigrid methods we present some simple nonlinear iterations, which are used as part of the multigrid algorithm.

3.4.2 Jacobi and Gauss-Seidel Type Iterations

Nonlinear Jacobi and Gauss-Seidel iterations are the nonlinear counterpart of their linear versions introduced in Section 3.3. Using the notation introduced in Section 3.3 we consider the nonlinear system of equations

$$F(u) = 0 \tag{3.76}$$

with $u \in \mathbb{R}^n$, and $F = [F_1, \ldots, F_n]^T$ with $F_j : \mathbb{R}^n \to \mathbb{R}$. Given an approximation $u^{(i)}$ to the exact solution u^* to (3.76) the iteration $u^{(i+1)}$ is gained by solving, for each $j = 1, \ldots, n$ the equation

$$F_j(u_1^{(i)}, \ldots, u_{j-1}^{(i)}, v, u_{j+1}^{(i)}, \ldots, u_n^{(i)}) = 0 \tag{3.77}$$

for the unknown v and setting

$$u_j^{(i+1)} = u_j^{(i)} + \omega(v - u_j^{(i)}) \tag{3.78}$$

for damping parameter $\omega \in \mathbb{R}$. This is exactly the same iteration as given in Equations (3.14) and (3.15) except that (3.77) is now a nonlinear equation to solve. This nonlinear equation may be solved using Newton's method, giving rise to a *Jacobi-Newton* iteration. It is usual to use only a single Newton iteration at a point to give an appropriate update, although the nonlinear equation may be solved more accurately if desired, but this does not improve the asymptotic converge rate [139, pg. 327]. The nonlinear SOR iteration is

given by replacing (3.77) with

$$F_j(u_1^{(i+1)}, \ldots, u_{j-1}^{(i+1)}, v, u_{j+1}^{(i)}, \ldots, u_n^{(i)}) = 0. \tag{3.79}$$

Some useful insight into the value of these nonlinear iterations can be gained from a consideration of the convergence properties of the methods. To this end we compare the Jacobi-Newton method with a *Newton-Jacobi* method, where linear Jacobi iterations are performed as an approximate solve of a Newton iteration. A first useful result is the following

Lemma 3.9. *A Jacobi-Newton iteration with a single Newton step per unknown is equivalent to a Newton-Jacobi iteration with a single linear Jacobi iteration (using a zero initial guess) per Newton iteration when applied to the nonlinear Equation* (3.76)*.*

Proof. The proof is formed by construction of both of the iterations. Let $J = F'(u^{(i)})$ be the Jacobian matrix represented as the sum

$$J = D - L - U$$

for D the diagonal part, and L (U) the strictly lower (upper) triangular part of J. First the Jacobi-Newton iteration is formed. For this the solution of each of the nonlinear equations given in (3.77) is approximated using a single Newton iteration. Hence, the updated approximation v in the Jacobi update (3.78) is calculated as

$$v \approx u_j^{(i)} - \left(\frac{\partial F_j}{\partial u_j^{(i)}}\right)^{-1} F_j(u_1^{(i)}, \ldots, u_N^{(i)}).$$

Inspection of (3.75) shows that the term $\partial F_j / \partial u_j^{(i)}$ is simply the diagonal on the j^{th} row of the Jacobian matrix for the current iterate $u^{(i)}$. The term F_j is the j^{th} right-hand side term in the nonlinear system of equations. Therefore a full Jacobi-Newton sweep over all unknowns can be described by

$$u^{(i+1)} = u^{(i)} - \omega D^{-1} F(u^{(i)}). \tag{3.80}$$

The representation of the Newton-Jacobi iteration is gained from the Newton iteration

$$u^{(i+1)} = u^{(i)} + \delta^{(i)}$$

where the correction $\delta^{(i)}$ is gained by approximating the solution of $J\delta = -F(u^{(i)})$ using a single linear Jacobi iteration with a zero initial guess. Using (3.18) the representation of the correction is

$$\delta^{(i)} = -\omega D^{-1} F(u^{(i)}), \tag{3.81}$$

and the Newton-Jacobi iteration is written the same as the Jacobi-Newton iteration in (3.80). □

Lemma 3.9 may be used in the proof of the following lemma:

Lemma 3.10. *For problem* (3.76) *let* F *be continuously differentiable and assume that* $(F')^{-1}$ *exists on convex and bounded* $\mathcal{D} \subset \mathbb{R}^d$*. Assuming that* u^* *is a unique solution of* (3.76) *in* $\mathcal{D}$ *then the asymptotic convergence rate* σ *of a Jacobi-Newton iteration is given by*

$$\sigma = \lim_{i \to \infty} \frac{\left\| F(u^{(i+1)}) \right\|}{\| F(u^{(i)}) \|} = \rho(H(u^*)) \tag{3.82}$$

where H *is the error propagation matrix for the linear Jacobi iteration applied to the equation*

$$F'(u^*)\delta u = F(u). \tag{3.83}$$

Proof. See the proof of [139, Lemma 10.3.1]. □

A similar result holds also in the case of an SOR iteration [139, Lemma 10.3.3]. Hence the nonlinear Jacobi and Gauss-Seidel type iterations converge at the same asymptotic rate as their linear counterparts. The iterations work best when the Frèchet derivative is symmetric positive definite, and in this case the condition given in (3.27) holds. As discussed in Subsection 3.3.1, the spectral radius of the error propagation matrices for linear Jacobi and Gauss-Seidel iterations is $\mathcal{O}(1 - \rho(H(u^*)))$, and the running time of the methods is $\mathcal{O}(N^3)$, for N the number of unknowns in the system. As the nonlinear iterations have the same asymptotic rate of convergence, the order of the running time of the methods is also $\mathcal{O}(N^3)$. Hence a Jacobi-Newton iteration is not well suited as a solver. However, the nonlinear stationary Jacobi and Gauss-Seidel iterations can be used as smoothers, as they

are in the linear case. This makes these iterations an important part of nonlinear multigrid methods, which are described in detail in the next chapter.

The methods introduced in this chapter give a grounding in the field of iterative solvers, and provide a basis upon which to discuss some of the most powerful classes of algorithms for solving systems of equations arising from the discretisation of linear and nonlinear PDEs of elliptic type - multigrid methods. These methods are introduced in more detail in the next chapter. Theory relevant to their convergence is given in Chapter 5.

Chapter 4

Introduction to Multigrid Methods

Multigrid methods are a very powerful class of iterative solvers for PDEs discretised by a local mesh-based scheme. They are most often used for the iterative solution of PDEs modelling some physical phenomenon that is diffusion dominated. Most commonly this involves second order PDEs, although fourth order and higher PDEs may also be dealt with by multigrid methods (see [204, 205] and references therein). The reason that multigrid methods are so popular is that when they are applicable to a problem they often display *optimal convergence behaviour*. What this means is that the convergence rate of a multigrid method is mesh-independent and the amount of computational effort per iteration is directly proportional to the number of unknowns N on a grid. Hence, the amount of computational effort required to solve a problem on a grid Ω with N unknowns to a desired degree of accuracy is $\mathcal{O}(N)$. Multigrid methods are most easily applicable to diffusion-dominated PDEs of elliptic and parabolic type. Advection dominated PDEs of hyperbolic type may be treated using multigrid techniques, although great care is required in the formulation of the algorithms (*e.g.* [17, 138, 146, 187]). The reasons why multigrid methods are so effective should be made clear in the discussions given in the following sections.

There are two main types of multigrid methods - geometric and algebraic. This thesis is concerned with geometric multigrid methods, in which geometric information regarding a problem is used to form a solution algorithm. Algebraic multigrid methods are very popular and their development and analysis forms an active research area ([40, 117, 175, 180, 198] are just a handful of recent publications). They use algebraic

information about a matrix associated with the discrete system of equations, rather than geometric information on a sequence of grids, and are therefore more general algorithms than the geometric variants. However, in cases where geometric information is salient, a geometric multigrid algorithm will be a very competitive choice of solver. Hence we concentrate on the cases in which geometric multigrid is desired as a solver. In this thesis we also make a comparison between Newton-Multigrid and Nonlinear Multigrid methods (to be introduced in Section 4.2). There exists no general algebraic variant of the Nonlinear Multigrid iteration, so that a fair comparison of these algorithms should be performed using the geometric algorithms.

Before introducing linear and nonlinear geometric multigrid methods some common notation is outlined here. As the name multigrid suggests, multiple grids are required in the formulation. Let $\{G_j\}_{j=1}^{J}$ be a set of grids such that

$$\Omega_j \subset \Omega_J, \quad j \neq J, \tag{4.1}$$

where Ω_J is a fine grid on which a solution is sought. It is common to choose the grids such that they form a hierarchy

$$\Omega_1 \subset \Omega_2 \subset \ldots \subset \Omega_J \tag{4.2}$$

although multigrid methods can be defined for sequences of non-nested grids (*e.g.* [18, 60]). The discussion presented here is for the much more common case that the sequence of grids is nested. An example of a two-dimensional sequence of nested grids is shown in Figure 4.1.

Let $\mathcal{N}_j$, $\mathcal{E}_j$ and $\mathcal{T}_j$ represent the set of nodes, edges and elements on a grid Ω_j. The characteristic grid spacing on Ω_j, as defined in (2.1), is given by h_j. Let $\mathcal{V}_J$ represent the space of functions in which the solution to an algebraic problem of the form (3.1) or (3.62) is sought on grid Ω_J. Associate with each grid Ω_j, $j = 1, \ldots, J$ a space $\mathcal{V}_j$, which for the purposes of this thesis is a space of finite element functions (see Subsection 2.3.2) defined using geometrical information of grid Ω_j, $j = 1, \ldots, J$. Note that even though we assume that $\Omega_j \subset \Omega_{j'}$ for $j < j'$ it is not necessary for $\mathcal{V}_j$ to be a subspace of $\mathcal{V}_{j'}$. In order to move functions between the different function spaces the *interpolation* operators

$$\begin{aligned} R_j^{j-1} &: \mathcal{V}_j \to \mathcal{V}_{j-1}, \quad j = 2, \ldots, J \\ P_{j-1}^{j} &: \mathcal{V}_{j-1} \to \mathcal{V}_j, \quad j = 2, \ldots, J \end{aligned} \tag{4.3}$$

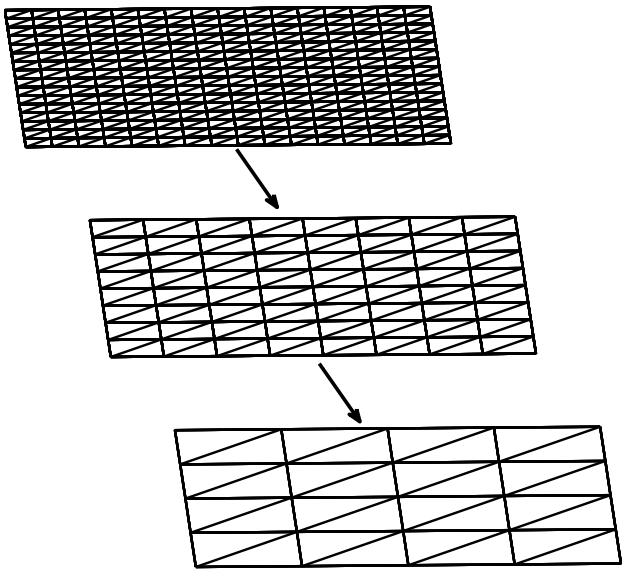


Figure 4.1: Example of a two-dimensional hierarchy of nested grids. All nodes on a coarse grid are also present on the finer grids. Each coarse grid element is the union of four fine grid elements.

are used. These operators may be suited to a specific problem, and are described in more detail in Subsection 4.1.2. R_j^{j-1} will be termed a *restriction* operator, and P_{j-1}^j will be termed a *prolongation* operator.

We are now ready to start the discussion of multigrid methods. Section 4.1 gives an introduction to linear multigrid methods. The concepts introduced are directly used in the formulation of the nonlinear multigrid methods described in Section 4.2. The convergence theory required for the methods is left for Chapter 5.

4.1 Linear Multigrid

Consider the solution of the discrete operator equation

$$A_J u_J = f_J \tag{4.4}$$

for $A_J : \mathcal{V}_J \to \mathcal{V}_J$ where $\mathcal{V}_J$ is defined above. Associate with $\mathcal{V}_J$ a grid Ω_J. Consider that Ω_J is the finest grid in a hierarchy of nested grids, as given in (4.2), and that we are given interpolation operators (4.3). The multigrid algorithm can be split into two parts

- *smoothing* and *coarse grid correction*. The process of smoothing has been mentioned already, but is explained in more detail in Subsection 4.1.1. Simply put the smoothing step seeks to remove high frequency components from the error. Coarse grid correction involves solving for a correction term on a coarser grid, and this process is described in more detail below. Let $u_j^* \in \mathcal{V}_j$, $j = 1, \ldots, J$ be the exact solution to the system of equations

$$A_j u_j = f_j \tag{4.5}$$

for $A_j : \mathcal{V}_j \to \mathcal{V}_j$. In order to gain an accurate solution the operators A_j, $j = 1, \ldots, J$ should be an accurate approximation of the continuous differential operator restricted to spaces $\mathcal{V}_j$, $j = 1, \ldots, J$. Let $u_j \in \mathcal{V}_j$ represent some approximation to the exact solution. The error e_j and residual r_j are defined by

$$e_j \equiv u_j^* - u_j, \qquad r_j \equiv f_j - A_j u_j \tag{4.6}$$

and satisfy the residual equation

$$A_j e_j = r_j \tag{4.7}$$

by the substitution of $f_j = A_j u_j^*$. The linear multigrid algorithm solves the residual equation on a coarse grid in order to calculate an approximate update term on the fine grid, *i.e.* let u_j represent an approximation to (4.5) on grid Ω_j, and e_{j-1}^* represent the exact solution to the residual equation on grid Ω_{j-1}. The coarse grid correction step then defines an updated approximation $\tilde{u}_j$ to u_j^* to be

$$\tilde{u}_j = u_j + P_{j-1}^j e_{j-1}^*. \tag{4.8}$$

This type of update is called an exact coarse grid correction, as e_{j-1}^* is the exact correction on the coarse grid. However, an approximate solution on the coarse grid may also be sufficient to get a good update on the fine grid. This concept is used in the linear multigrid algorithm, which we give after the introduction of some notation. Let S_j represent a smoothing operator such that

$$S_j : \mathcal{V}_j \times \mathcal{V}_j \to \mathcal{V}_j. \tag{4.9}$$

Using this notation the Jacobi iteration, for example, would be given by (see (3.18))

$$S_j(f_j, u_j) = u_j + \omega D^{-1}(f_j - A_j u_j), \tag{4.10}$$

although S_j may be any smoother. Multiple smooths are given by the recursion

$$\begin{aligned} S_j^{\nu}(f_j, u_j) &= S_j^{\nu-1}(f_j, S_j(f_j, u_j)), \quad \nu > 1 \\ S_j^{1}(f_j, u_j) &= S_j(f_j, u_j). \end{aligned} \tag{4.11}$$

Linear multigrid is then given by Algorithm 4.1. Lines 2 and 11 perform the smoothing

Algorithm 4.1 Linear Multigrid

Require: Operators A_j S_j, $j = 1, \ldots, J$ and R_j^{j-1}, P_{j-1}^{j}, $j = 2, \ldots, J$

1: **function** LIN-MG($j, u_j, f_j, \gamma, \nu_1, \nu_2$)
2: $u_j \leftarrow S_j^{\nu_1}(f_j, u_j)$ ▷ *Perform ν_1 smooths*
3: $r_j \leftarrow f_j - A_j u_j$
4: $e_{j-1} \leftarrow 0, \quad r_{j-1} = R_j^{j-1} r_j$
5: **if** $j = 2$ **then**
6: $e_{j-1} \leftarrow A_{j-1}^{-1} r_{j-1}$
7: **else**
8: **for** $i = 1, \ldots, \gamma$ **do** ▷ *Approximate correction on coarse grid*
9: $e_{j-1} \leftarrow$ LIN-MG($j - 1, e_{j-1}, r_{j-1}, \gamma, \nu_1, \nu_2$)
10: $u_j \leftarrow u_j + P_{j-1}^{j} e_{j-1}$
11: $u_j \leftarrow S_j^{\nu_2}(f_j, u_j)$ ▷ *Perform ν_2 smooths*
12: **return** u_j

and lines 5-10 perform coarse grid correction. The coarse grid correction will be exact if there are only two grids in the hierarchy. This case will be referred to as the two-grid iteration. For an elliptic PDE discretised on a hierarchy of quasi-regular grids it can be shown that the two-grid and multigrid iterations converge independent of mesh parameters so long as the discrete operator is symmetric positive definite [23, 42, 156, 177, 196] or 'close to' symmetric positive definite [26]. Non-symmetric and indefinite operators may be treated so long as the smoothing property and approximation property [85, §6.5] are satisfied. These properties are difficult to show, and a discussion of the convergence of multigrid methods is left until Chapter 5. In the next sections we explain the smoothing and coarse grid correction step of Algorithm 4.1 and then move on to describe the grid transfer operators in more detail.

4.1.1 Smoothing and Coarse Grid Correction

The concept of a smoother has been introduced without a clear definition of what a smooth or oscillatory function should be. In a multigrid context a smooth function on a grid Ω_j is a function that may be well represented on the next coarsest grid in the hierarchy Ω_{j-1}. As demonstrated in Subsection 3.3.2 there is a limit on the frequency of a function that can be represented on a given grid. For ease of representation the one-dimensional case is considered, although the following arguments apply directly to the two- and three-dimensional cases as well.

Consider that a one-dimensional grid Ω_j has $N_j + 1$ equally spaced points on the range $[a, b]$. Let $L = b - a$ and assume homogeneous Dirichlet boundary conditions. The highest frequency function that can be represented on Ω_j is one that changes sign at most $N_j - 1$ times, such as the *Fourier mode* $e^{(i(N_j-1)\pi x)/L}$, for $i = \sqrt{-1}$. The same reasoning shows that the highest frequency function on grid Ω_{j-1} has frequency $N_{j-1} - 1$. From this we gain a simple definition of high and low frequency functions. Let

$$\varphi^j_{k,x} = e^{(ik\pi x)/L}, \quad i = \sqrt{-1}. \tag{4.12}$$

Then the set of high frequency functions on grid Ω_j are given by the set Φ^j_{hf}, which is defined as (in one dimension)

$$\Phi^j_{\text{hf}} = \left\{ \varphi^j_{k,x} \mid k \geq N_{j-1} \right\}. \tag{4.13}$$

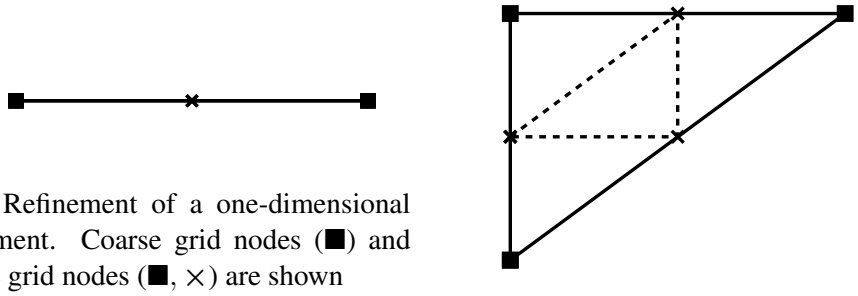
In d dimensional space, where the coordinate directions are given by x_l, $l = 1, \ldots, d$, the high frequency functions on a square grid with equal number of points in each grid direction and side length L are given by

$$\Phi^j_{\text{hf}} = \left\{ \prod_{l=1}^{d} \varphi^j_{k_l,x_l} \mid \exists l, k_l \geq N_{j-1} \right\} \tag{4.14}$$

Any function that is not high frequency is low frequency. The exponential functions used above are just an example of periodic oscillatory functions that can form a basis for the functions defined on a space. However, any oscillatory functions can be used. In order to use Fourier analysis (see Chapter 5) it is typical for the exponential functions to be defined in the range $k \in [-N_j/2, N_j/2)$, which would require a simple redefinition of the above set of oscillatory functions. The exact definition of high and low frequency depends

on the relationship between grids Ω_j and Ω_{j-1}, in particular the way in which a grid is coarsened. Thus we now consider the way in which a grid is typically coarsened.

Note that we have assumed that the grids are nested such that all of the points on grid Ω_{j-1} are also on Ω_j. Nested grids are often constructed through refinement of a coarse grid rather than coarsening of a fine grid, and the most common way to do this is to join the mid-points of edges on an element. We call this *regular* refinement and Figure 4.2



(a) Refinement of a one-dimensional element. Coarse grid nodes (■) and fine grid nodes (■, ×) are shown

(b) Refinement of a two-dimensional element. Coarse grid nodes (■) and fine grid nodes (■, ×) are shown

Figure 4.2: Regular refinement of one- and two-dimensional simplicial elements

shows the refinement of elements in one and two dimensions. Regular refinement in three dimensions follows the same principle, although extra considerations need to be taken (see [16, 172]) which are outside of the scope of this thesis. A naïve refinement may lead to triangulations with undesirable qualities, especially when the grid is unstructured [161, 203]. However, for the purposes of this thesis we assume that regular refinement will give a usable mesh. Using this refinement strategy it is noted that in one dimension

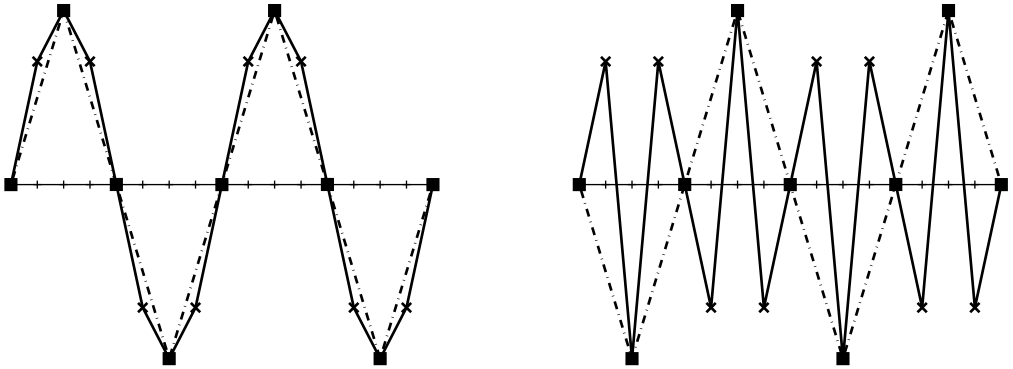
$$N_j = 2N_{j-1} \tag{4.15}$$

holds, and so the set of high frequency functions in one dimension is given by

$$\Phi^j_{\text{hf}} = \{\varphi_{k,x} \mid k \geq N_j/2\}\,. \tag{4.16}$$

Figure 4.3 demonstrates that low frequency functions on a fine grid are well approximated on a coarse grid, whilst high frequency functions are not well represented. Let Ω_j represent the fine grid in Figure 4.3. In this case $N_j = 16$ and $N_{j-1} = 8$ and a high frequency function on Ω_j has an index greater than or equal to 8. Figure 4.3a shows the representation of a low frequency function with index 4, and Figure 4.3b shows the representation

of a high frequency function with index 12. For ease of presentation the representation of



(a) Smooth function represented on fine and coarse grid. Coarse grid nodes (■) and fine grid nodes (■, ×) are shown.

(b) Oscillatory function represented on fine and coarse grid. Coarse grid nodes (■) and fine grid nodes (■, ×) are shown.

Figure 4.3: Representations of a smooth and oscillatory function on a fine and coarse grid

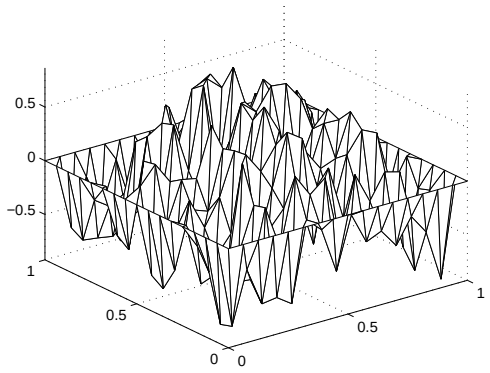
a sine function has been shown, although the same concept holds for the functions (4.12). As can be seen the low frequency function is well represented and the high frequency function is not well represented at all. This is due to an effect known as *aliasing* in which different fine grid functions appear to have the same frequency on a coarse grid, as in Figure 4.3. The relation

$$\sin(\pi i/L)|_{\Omega_{j-1}} = -\sin(\pi(N_j - i)/L)|_{\Omega_{j-1}}, \quad i = 1, \ldots, N_j/2 \tag{4.17}$$

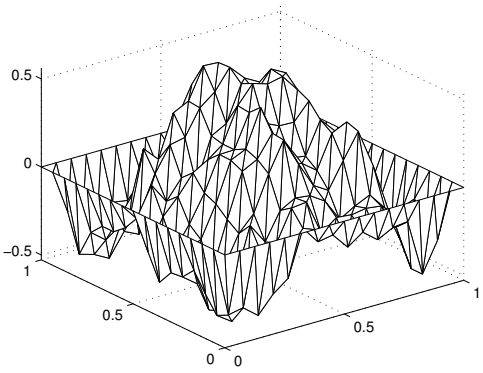
can be observed in the representation of the coarse grid functions in Figure 4.3, where $f|_{\Omega_{j-1}}$ is the natural inclusion of function f in the space of piecewise linear functions on grid Ω_{j-1}. Hence high frequency data must be removed from a function before it is transferred to a coarser grid. A smoothing operator removes high frequency components from the *error* in an approximation, as is demonstrated in Figure 4.4. The error in approximation is shown for the first 5 SOR iterations applied to a random initial guess for the linear Laplacian problem

$$\begin{aligned} -\nabla \cdot \nabla u &= f, \quad \vec{x} \in \Omega \\ u &= 0, \quad \vec{x} \in \partial\Omega, \end{aligned} \tag{4.18}$$

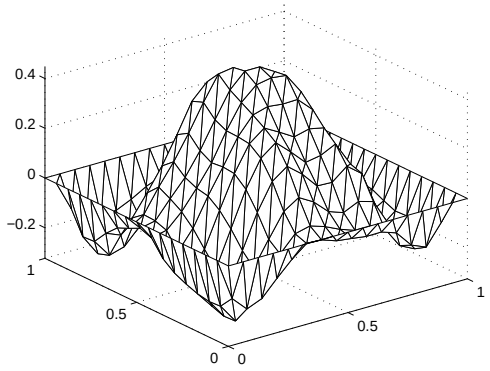
with exact solution $u^* = \sin(\pi x)\sin(\pi y)$ for a finite element discretisation using continuous piecewise linear elements. As can be seen the high frequency components are



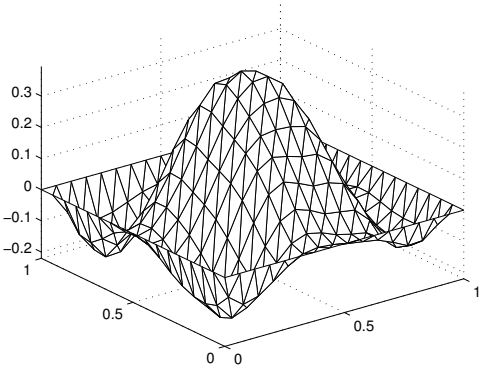
(a) Initial error with oscillatory components



(b) Error after 1 SOR iteration



(c) Error after 3 SOR iterations



(d) Error after 5 SOR iterations

Figure 4.4: Demonstration of smoothing of the error of an approximation to a discrete linear Laplacian equation using SOR iterations with a weighting factor 0.8

quickly removed from the error, leaving only the low frequency error components. These low frequency components are not damped very well by successive applications of the smoothing operator, and in fact the rate at which the smoother converges deteriorates as more smoothing iterations are performed, as shown in Figure 4.5. To understand why this

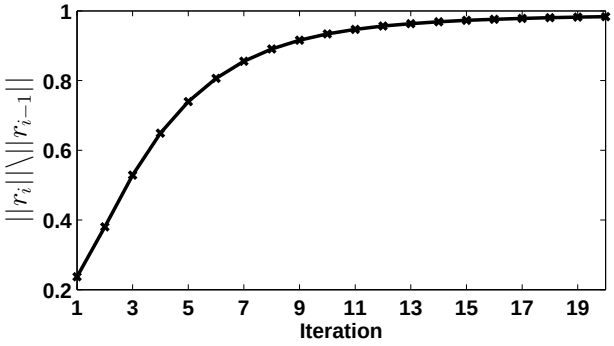


Figure 4.5: Convergence factors plotted against iteration number for an SOR smoothing operator. The convergence factor is much better for the first few smoothing iterations.

should be, we consider the case of a Jacobi or Gauss-Seidel smoothing operator in two dimensions. In this case a linear equation is solved over the support of a basis function for each node on the grid. Figure 4.6 shows the support of a basis function on a fine and coarse grid centred at a node present on both. It can be seen that the support of a basis function is directly proportional to the grid spacing, and that the basis function on the coarser grid has a lower frequency. In Equations (3.42) - (3.44) we show that a smoothing iteration uses a high frequency subspace in order to calculate a point-wise correction. It is therefore sensible that the high frequency data is captured more easily in a smoothing iteration than the low frequency data. In order to capture lower frequency data a smoother can be applied with lower frequency basis functions, which can be achieved by performing a smoothing iteration on a coarser grid. Another way of considering this is that functions that are smooth on grid Ω_j contain components that are high frequency on the coarser grid Ω_{j-1}. Hence moving to a coarser grid and performing smoothing will remove these components. This is what is performed in the coarse grid correction step (lines 5-10) of linear multigrid (Algorithm 4.1).

Once a correction term has been transferred to the fine grid from the coarse grid it is expected that the approximation to the error on the coarse grid nodes is close to the exact solution, whereas the correction at fine grid nodes which are not on the coarse grid are further from the true correction. This causes the error after the correction to contain high frequency data, as illustrated in Figure 4.7. Hence the post-smoothing step in line 11 in Algorithm 4.1 is performed to remove these high frequency components, and the error

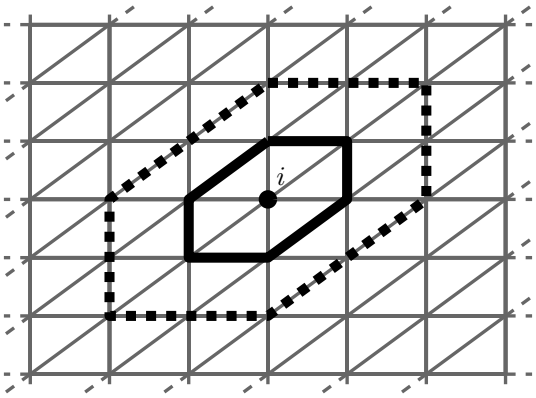
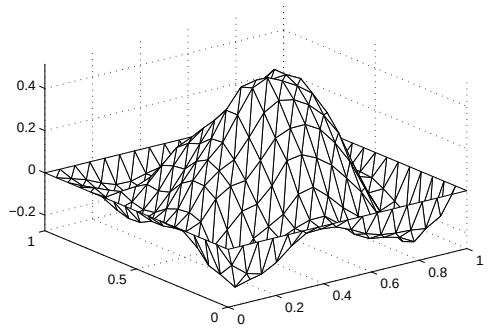


Figure 4.6: The support of a basis function centred at node i on a fine grid (solid black line) and a coarse grid (broken black line). The diameter of the support of a basis function is directly proportional to the grid spacing.

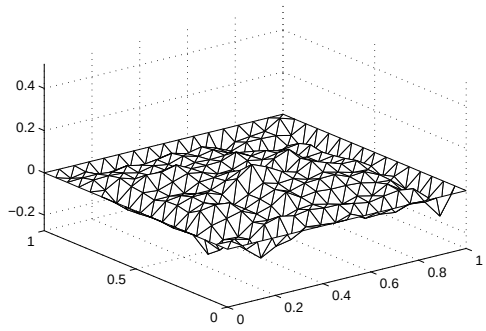
after a single multigrid iteration is greatly reduced compared to performing smooths on a single level only.

To increase the accuracy to which a coarse grid problem is solved the number of multigrid iterations performed per level may be increased. This is done in the loop in line 5 of Algorithm 4.1, which is controlled by the scalar γ. Varying γ gives rise to different patterns in which each grid is visited and the algorithm is named differently for different values of γ. The case that $\gamma = 1$ is called the *V-cycle* and $\gamma = 2$ the *W-cycle*, as the order in which the grids are visited resembles a V and a W, respectively (see Figure 4.8). It is also possible to vary γ to produce other cycles. For example, a popular cycle is the F-cycle in which a W-cycle is performed the first time a grid is visited and a V-Cycle the second time. However, the most popular implementations of multigrid use either the V- or W-cycle. In this thesis we are particularly interested in the V-cycle, as this is often more efficient to use than the W-cycle. This is because the improvement in convergence using the W-cycle is not generally sufficient to offset the increase in execution time relative to the V-cycle, unless the latter is particularly slow to converge.

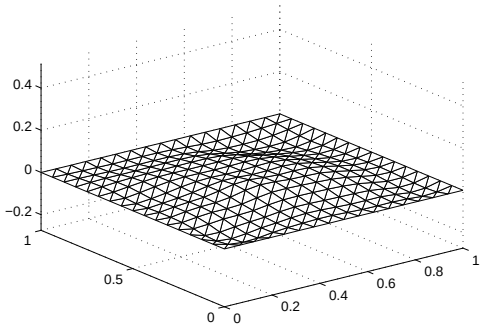
As yet the grid transfer operators given in (4.3) have not been defined, and we have assumed that the transfer operators are appropriate for any given problem. A suitable choice of grid transfer operators depends on the underlying properties of the grids, the finite element function spaces, and the PDE being solved. In the next section a discussion of how to choose an appropriate grid transfer operator is given. This discussion ties in with the importance of an appropriate choice of coarse grid linear operators A_j, $j \neq J$ which should approximate the action of the fine grid operator A_J on the coarse grids Ω_j,



(a) Error after pre-smoothing (three SOR iterations)

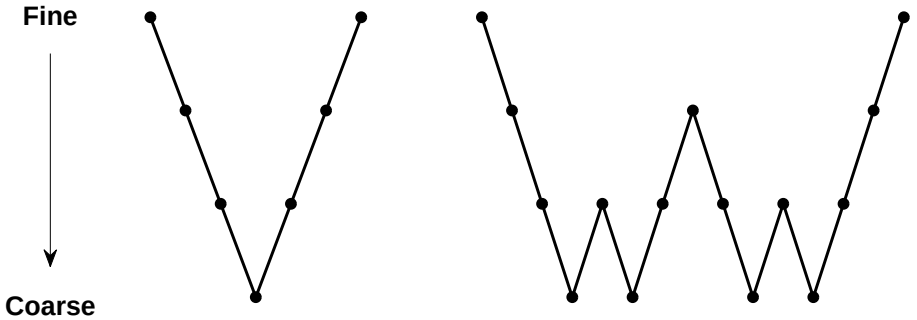


(b) Error after coarse grid correction, in which high frequency components are introduced into the error



(c) Error after post-smoothing (three SOR iterations)

Figure 4.7: Progression of the error through a single multigrid iteration.



(a) Multigrid V-Cycle (b) Multigrid W-Cycle

Figure 4.8: Order in which grids are visited for multigrid V- and W-cycles.

$j \neq J$.

4.1.2 Grid Transfer and Coarse Grid Operators

The convergence of a multigrid iteration is sensitive to the grid transfer operators that are used. Transfer operators should be chosen such that an approximation property (see [85, §6.1.3]) is satisfied, which gives a bound for the error in the solution transferred from a fine to a coarse grid. This gives a condition in which a correction made on a coarse grid is appropriate to be used on a fine grid. Considering the solution of (4.5) the approximation property requires a bound for the value

$$\left\| A_j^{-1} - P_{j-1}^j A_{j-1}^{-1} R_j^{j-1} \right\|. \tag{4.19}$$

In this estimate the operator on a coarse grid is used, which is not defined as part of problem (4.4). The coarse grid operator may be defined in different ways. If the discrete operator is defined by a discretisation on a given grid, it is possible to simply re-discretise the operator on all grids in a hierarchy. Another option is to use the *Galerkin* coarse grid operator (see [42, pg. 75], [177, §7.7.4]) defined by

$$A_{j-1} \equiv R_j^{j-1} A_j P_{j-1}^j, \tag{4.20}$$

where typically R_j^{j-1} is the adjoint of P_{j-1}^j. This is useful in particular in the case when the operator A_J is symmetric positive definite [23,160,196], in which case the coarse grid operator is symmetric positive definite in the energy norm

$$\|\cdot\|_{A_j}^2 = (\,\cdot\,,\,\cdot\,)_{A_j} \tag{4.21}$$

for $(\,\cdot\,,\,\cdot\,)_{A_j}$ as defined in (2.10). This is easily verified. The usefulness of this fact is made clear in the convergence theory discussed in Chapter 5. The coarse grid operators are stored separately in an implementation, since to apply an operator on the coarsest grid using the definition (4.20) would require the application of the operator on the finest grid. In this thesis both the Galerkin coarse grid operator and re-discretised operators are used, and it will be stated when which one is used.

As part of the Galerkin coarse grid operator we again make use of the grid transfer operators. It is plain to see that the transfer operators used will change the coarse grid operator, and so appropriate transfer operators need to be chosen. It is desirable for a grid

transfer operator to be 'optimal' in some norm, *i.e*. it should be a projection operator. As an example we consider a hierarchy of two-dimensional grids as given in (4.2) with $\mathcal{V}_j$ the space of piecewise linear C^0 nodal functions $S_0(\Omega_j)$ (see (2.14)). Note that $\mathcal{V}_{j-1} \subset \mathcal{V}_j$. This simplifies the definition of the grid transfer operators, but it is also possible to find transfer operators for the non-nested case (see Chapter 8 for an example and [60] for a more detailed treatment).

We start with the definition of the prolongation operator. We want this to satisfy the orthogonality condition

$$(u_j, P_{j-1}^j w_{j-1}) = (u_j, w_{j-1}), \quad \forall u_j \in \mathcal{V}_j, \quad w_{j-1} \in \mathcal{V}_{j-1}, \tag{4.22}$$

for $(\,\cdot\,,\,\cdot\,)$ the L_2 inner product. Since $w_{j-1} \in \mathcal{V}_{j-1} \subset \mathcal{V}_j$ the prolongation is just the natural inclusion operator. Hence the mathematical representation is just the identity transformation. The matrix representation of the prolongation operator performs a change of basis from $\mathcal{V}_{j-1}$ to $\mathcal{V}_j$. Recalling the representation of the functions expanded in a nodal basis as

$$u_j = \sum_{i=1}^{N_j} u_{j,i}\varphi_{j,i}, \quad w_{j-1} = \sum_{i=1}^{N_{j-1}} w_{j-1,i}\varphi_{j-1,i} \tag{4.23}$$

where $\mathcal{V}_j = \operatorname{span}\{\varphi_{j,i}\}_{i=1}^{N_j}$. Let $\vec{u}_j = [u_{j,1}, \ldots, u_{j,N_j}]^T$ be the vector representation of the function u_j on grid Ω_j, and define $\vec{w}_{j-1}$ similarly. Using the matrix $P_{j-1}^j : \mathbb{R}^{N_j} \times \mathbb{R}^{N_{j-1}}$, $P_{j-1}^j w_{j-1}$ then gives the vector representation of function $w_{j-1} \in \mathcal{V}_{j-1}$ in $\mathcal{V}_j$, *i.e*.

$$P_{j-1}^j \vec{w}_{j-1} = \vec{w}_j = [w_{j,1}, \ldots, w_{j,N_j}]^T \tag{4.24}$$

such that $w_j = \sum_{i=1}^{N_j} w_{j,i}\varphi_{j,i}$. The $w_{j,i}$, $i = 1, \ldots, N_j$ are simply the nodal values of w_{j-1} on grid Ω_j. For any nodes $\vec{x} \in \Omega_j \cap \Omega_{j-1}$ the prolongation operator is the identity. For any node $\vec{x} \in \Omega_j \setminus \Omega_{j-1}$ the prolongation operator gives the average of co-linear coarse grid nodes, as the functions in $\mathcal{V}_j$, $j = 1, \ldots, J$ are continuous and piecewise linear. As an example consider the set-up in Figure 4.9. The result of applying the prolongation operator at nodes marked 10 and 11 is given by

$$\begin{aligned} w_{j,10} &= w_{j-1,10} \\ w_{j,11} &= \frac{1}{2}(w_{j-1,10} + w_{j-1,12}). \end{aligned} \tag{4.25}$$

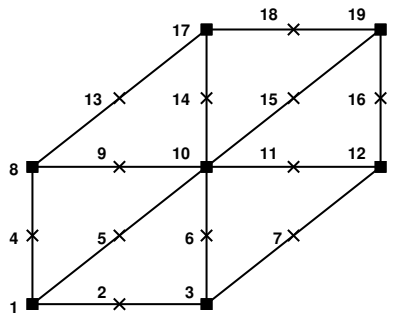


Figure 4.9: Example coarse grid elements with nodes on on fine (×) and coarse (■) grid

The restriction operator is the transpose of the prolongation, such that

$$R_j^{j-1}\vec{v}_j = \left(P_{j-1}^j\right)^T \vec{v}_j = \vec{v}_{j-1} = [v_{j-1,1}, \ldots, v_{j-1,N_{j-1}}]. \tag{4.26}$$

The result of applying the restriction operator at node labelled 10 in Figure 4.9 is given by

$$v_{j-1,10} = v_{j,10} + \frac{1}{2}(v_{j,5} + v_{j,6} + v_{j,9} + v_{j,11} + v_{j,14} + v_{j,15}). \tag{4.27}$$

The restriction operator defined in this way may be used as part of a Galerkin coarse grid operator. However, if the operator is re-discretised on a coarser grid the rows in the restriction matrix need to be normalised in order to give an appropriate transfer operator.

Note that a grid transfer operator may be easier to obtain in a finite element setting, as the inner product is used as part of the variational formulation. For finite difference approximations it is not directly obvious which grid transfer operators should be used as a basis for the underlying space is not explicitly constructed. There does exist theory which gives simple heuristics for the choice of restriction and prolongation operators (see [42, 134, 177]) which relates the accuracy of the grid transfer operators to the order of the PDE to solve.

We have now given a description of the different components of the linear multigrid iteration in a heuristic manner, without much emphasis on the theory of multigrid methods which is given in Chapter 5. Before a discussion of the theory a justification for the statement that a multigrid iteration is of optimal order ($\mathcal{O}(N_J)$) running time is given.

4.1.3 Characterisation of Running Time

In this section we present a characterisation for the running time of a linear multigrid iteration under the assumption that the number of iterations required to reach a convergence criterion is independent of the grid level. This simply shows that the computational effort required to perform a single multigrid iteration is proportional to the number of unknowns in the system. To do this we break down the computational cost of the linear multigrid algorithm in terms of constituent parts, taking advantage of the recursive definition of the algorithm. We summarise the characterisation as presented in [177, §2.4.3], assuming that a regular coarsening as introduced in Subsection 4.1.1 is used.

The cost per linear multigrid iteration is a combination of the cost of performing smoothing, grid transfers and multigrid on a coarser grid. Letting W_J denote the computational cost for performing smoothing and grid transfer operations on grid Ω_J, and let C_J be the cost of a multigrid iteration on grid Ω_J. Then the cost per multigrid iteration can be defined recursively as

$$C_j = W_j + \gamma C_{j-1}, \quad C_2 = W_2 + C_1. \tag{4.28}$$

where γ is the number of times a multigrid iteration is to be performed on each level (see Algorithm 4.1) and C_1 is the cost to perform a solve on the coarsest grid level. It is assumed that

$$W_j \approx CN_j \tag{4.29}$$

for N_j the number of nodes on grid Ω_j and C some small integer constant bounding the number of operations performed per node on the grid in the grid transfer and smoothing processes. It is clear that this should be the case for the grid transfer as this is performed as a sparse matrix-vector product where the number of non-zero entries in a row of the matrix is bounded by a small constant. For a point-wise smoothing operator such as the Jacobi or Gauss-Seidel a small number of calculations are required to solve a linear equation at each point on the grid. The same is true if a block smoother is used, so long as the size of the blocks is fixed, *i.e.* does not grow with the size of the grid.

It is also assumed that the cost of solving on the coarsest grid level is negligible, *i.e.* $C_1 \approx 0$. For a regular refinement of a two-dimensional grid we have

$$N_j \approx 4N_{j-1} \tag{4.30}$$

and so

$$W_j \approx 4W_{j-1}. \tag{4.31}$$

Using this and expanding the recursion (4.28) the cost of a linear multigrid iteration is given by

$$\begin{aligned} C_J &= W_J + \gamma C_{J-1} \\ &= W_J + \gamma \left(W_{J-1} + \gamma C_{J-1} \right) \\ &= \dots \\ &= \sum_{j=2}^{J} W_j \gamma^{J-j} + \gamma^{J-1} C_1 \\ &\approx W_J \sum_{j=2}^{J} \frac{\gamma^{J-j}}{4^{J-j}} \\ &\approx C N_J \sum_{j=2}^{J} \frac{\gamma^{J-j}}{4^{J-j}}. \end{aligned} \tag{4.32}$$

For the multigrid V-cycle ($\gamma = 1$) and the W-cycle ($\gamma = 2$) the cost per iteration is bounded by

$$C_J \leq \begin{cases} \frac{4}{3} C N_J, & \gamma = 1 \\ 2 C N_J, & \gamma = 2, \end{cases} \tag{4.33}$$

which is linear with respect to the number of nodes on a grid. Larger values of γ are not common, and for large enough γ the running time per iteration is no longer linear (see [177, pg. 51]). For the purposes of this thesis we are mainly interested in the multigrid V-cycle which is clearly of linear order running time per iteration. A similar characterisation of running time in higher spatial dimensions is a simple extension of the characterisation given here. For example, in three dimensions, we simply require to know that, for regular refinement, the relation

$$N_j \approx 8N_{j-1}$$

holds. The expansion of the recursion is left to the reader, but the result is that in d

dimensions for a regular refinement strategy the computational cost per V-cycle is given by

$$C_J \leq C N_j \frac{2^d}{2^d - 1}. \tag{4.34}$$

The result in (4.33), when combined with a grid independent convergence result, demonstrates that a multigrid iteration is of optimal cost $\mathcal{O}(N_j)$ on grid Ω_j. Grid independent convergence is discussed in more detail in Chapter 5. Before then we provide a note on a common use of multigrid methods in practice and then introduce nonlinear multigrid methods, which are the main focus of this thesis.

4.1.4 Linear Multigrid as a Preconditioner

In practice linear multigrid is often used as a preconditioner for a Krylov subspace method. The use of a linear multigrid iteration is described briefly in conjunction with the flexible GMRES iteration outlined in Algorithm 3.2. Preconditioning is performed in line 3 of Algorithm 3.2 in which the quantity of interest is the vector

$$z_j = M_j^{-1} v_j$$

using a preconditioning matrix M_j at iteration j. In the case of multigrid methods the multigrid iteration matrix is not formed explicitly. Instead a multigrid iteration is applied to the operator equation to be solved, *i.e.* to (see Algorithm 3.2)

$$A z_j = v_j.$$

The resulting vector is assigned as z_j in the FGMRES iteration. The application of multigrid to CG and GMRES is performed in a similar manner, but, in the latter case, the resulting algorithm is not as stable as when an FGMRES iteration is used (see Subsection 3.3.4).

As a preconditioner for a CG method the convergence of the multigrid iteration is typically accelerated [153, §6.11.3]. However, as part of a GMRES iteration multigrid may be used in a more interesting way. GMRES is an iterative solver applicable for non-symmetric and indefinite systems of equations, which a standard multigrid iteration may struggle to solve. It is not necessary, though, for a preconditioner to be applied to the full problem for GMRES to be an optimal algorithm [70, 124, 153]. For example, it may be

that a linear operator A may be split into a symmetric part A_{s} and a non-symmetric part A_{ns} such that

$$A = A_{\mathrm{s}} + A_{\mathrm{ns}}.$$

In this case it is possible that applying the multigrid iteration to the equation

$$A_{\mathrm{s}} z_j = v_j$$

in line 3 of Algorithm 3.2 will give a good preconditioner [63, 158].

It may also be the case that a non-convergent multigrid iteration will be suitable as a preconditioner for a GMRES iteration [153]. The use of multigrid as a preconditioner is a popular approach for solving problems of parabolic and elliptic type in both the linear (*e.g.* [4, 35, 116, 137, 145, 207]) and nonlinear case (*e.g.* [44, 63, 64, 99, 106, 142]), and is investigated as part of this thesis.

This concludes the discussion of linear multigrid methods, and provides a good basis for the discussion of nonlinear multigrid methods presented in the next section, which use the same principles introduced here.

4.2 Multigrid in a Nonlinear Setting

Nonlinear multigrid methods are based upon generalisations of the linear multigrid paradigm to a nonlinear setting. The idea that a fine scale problem can be efficiently solved by solving problems in spaces with low dimension is very appealing, especially as operations involved in the solution of nonlinear problems are often more computationally expensive than in the linear case. There are two main approaches to apply a multigrid method in a nonlinear context: Newton-Multigrid (Newton-MG) and Nonlinear Multigrid. There are various methods that can be used in both of these approaches and these will be introduced in Subsections 4.2.1 and 4.2.2.

In the next sections the solution of the discrete nonlinear equation

$$F_j(u_j) = 0, \quad j = 1, \ldots, J \tag{4.35}$$

or

$$A_j(u_j) = f_j, \quad j = 1, \ldots, J \tag{4.36}$$

is considered on a hierarchy of grids as defined in (4.2). F_j (A_j) represents the discretisation of the same arbitrary nonlinear PDE on a sequence of grids Ω_j, $j = 1, \ldots, J$. Note that (4.35) is equivalent to (4.36) setting $F_j(u_j) = f_j - A_j(u_j)$. It is assumed that a unique solution u_j^*, $j = 1, \ldots, J$, exists for the above equations in some ball $B_{u^*}(\delta)$ (see (3.63)). The residual r_j and error e_j in an approximation u_j on grid Ω_j are then defined as

$$r_j \equiv f_j - A_j(u_j), \quad e_j \equiv u_j^* - u_j. \tag{4.37}$$

4.2.1 Newton-Multigrid (Newton-MG)

The most obvious application of a multigrid method in the nonlinear setting is as part of a Newton (or more commonly an approximate Newton) iteration. The Newton iteration, as introduced in Subsection 3.4.1, requires a linear problem to be solved per Newton step. This may be done approximately with any appropriate iterative method. The term Newton-Multigrid will be used in this thesis to refer to any Newton iteration that uses a multigrid iteration as part of the linear solve. This includes the cases where linear multigrid is used as a solver, or as the preconditioner for a Krylov subspace solver for the linear system of equations. The method is presented in Algorithm 4.2 assuming that an approximation at Newton step i is given. This approach is popular in practice (*e.g.* [44,76,

Algorithm 4.2 Newton-MG

1: Set $\delta^{(i)}$ to be the approximate solution of $F'(u^{(i)})\delta^{(i)} = F(u^{(i)})$ from an iterative method using multigrid
2: $u^{(i+1)} \leftarrow u^{(i)} + \gamma\delta^{(i)}$

84,90,100,108]) where observed results show mesh-independent convergence. This is due to the mesh-independent properties of both the Newton and linear multigrid methods. For the Newton iteration it is difficult to prove the mesh-independent convergence, although it has been done for certain model problems (*e.g.* [107]). A discussion of the existing theory and the difficulties in proving convergence for general nonlinear problems is discussed in more detail in Chapter 5. Now we turn our attention to another approach to utilise multigrid principles in the nonlinear setting which leads to methods termed *nonlinear multigrid methods* in this thesis.

4.2.2 Nonlinear Multigrid Methods

This section introduces nonlinear multigrid methods applied to Equation (4.36), which use the same principles introduced for linear multigrid methods and apply them directly in a nonlinear setting. As such we seek to perform both smoothing and coarse grid correction. Typical nonlinear smoothing operators are introduced in Subsection 3.4.2 so we are left to describe the coarse grid correction step. In the linear case the error in approximation satisfies the residual Equation (4.7). This is not the case in the nonlinear setting and the nonlinear residual equation in space $\mathcal{V}_j$ reads

$$A_j(u_j^*) = A_j(u_j) + r_j \tag{4.38}$$

for the residual r_j defined as in (4.37). From the definition of the error in (4.37) we see that $u_j^* = e_j + u_j$. Since the current approximation u_j is known, if we can solve for u_j^* we can calculate the error in approximation. Using the residual equation on grid Ω_j does not change the definition of the problem. However, solving the nonlinear residual Equation (4.38) on a coarser grid gives rise to the equation

$$A_{j-1}(u_{j-1}^*) = A_{j-1}\left(R_j^{j-1}u_j\right) + R_j^{j-1}r_j \tag{4.39}$$

to solve for the approximation on grid Ω_{j-1}. The value that is solved for on the coarse grid is the approximation to the exact solution and the first variant of nonlinear multigrid - the *Full Approximation Scheme* (FAS) [32] - takes its name from this fact. FAS is presented in Algorithm 4.3. Note that in line 10 of Algorithm 4.3 the error is calculated on the coarse grid before interpolating to the fine grid. This is a necessary step to take, and in general, interpolating the approximation to the fine grid does not give a convergent iteration [177]. This is due to the fact that a smoothing iteration smooths the error in approximation, so that the error is well represented on the coarse grid, and it should therefore be the error that is transferred back to the fine grid. In Algorithm 4.3 the same restriction operator is used to restrict the approximation as well as the residual. However, it may be desirable to restrict the approximation with a higher accuracy than the residual (see [42, 177]). This is not necessary for the problems considered in this thesis, though.

To complete the description of Algorithm 4.3 the definition of the grid transfer and coarse grid operators are required. The grid transfer operators are formed in the same way as in the linear case as their definition depends only on the inner products and basis chosen for the finite-dimensional spaces. The coarse grid operator is a different matter,

Algorithm 4.3 FAS

Require: A_j, A_{j-1}, S_j, P_{j-1}^j, R_j^{j-1}
1: **function** FAS($j, u_j, f_j, \gamma, \nu_1, \nu_2$)
2: $u_j \leftarrow S_j^{\nu_1}(f_j, u_j)$ ▷ *Perform ν_1 pre-smooths*
3: $u_{j-1} \leftarrow R_j^{j-1} u_j, \quad \tilde{u}_{j-1} \leftarrow u_{j-1}$
4: $f_{j-1} \leftarrow A_{j-1}(u_{j-1}) + R_j^{j-1}(f_j - A_j(u_j))$
5: **if** $j = 2$ **then**
6: $\tilde{u}_{j-1} \leftarrow A_{j-1}^{-1}(f_{j-1})$
7: **else**
8: **for** $i = 1, \ldots, \gamma$ **do** ▷ *Approximate correction on coarse grid*
9: $\tilde{u}_{j-1} \leftarrow$ FAS($j-1, \tilde{u}_{j-1}, f_{j-1}, \gamma, \nu_1, \nu_2$)
10: $e_{j-1} \leftarrow \tilde{u}_{j-1} - u_{j-1}$
11: $u_j \leftarrow u_j + P_{j-1}^j e_{j-1}$
12: $u_j \leftarrow S_j^{\nu_2}(f_j, u_j)$ ▷ *Perform ν_2 post-smooths*
13: **return** u_j

though. This can no longer be stored as a matrix as the operator depends on the current approximation. Although a Galerkin coarse grid operator can be written mathematically it is not clear how to implement the method in practice without performing transformations to finer grids. There is limited research in which an algebraic coarse grid operator has successfully been applied for model problems in the nonlinear setting [66], but this process is not widespread and is not generally applicable. Consequently, the coarse grid operators in Algorithm 4.3 are assumed to be consistent discretisations of the differential operator on the coarse grids.

FAS was first presented in [32] and has been used effectively in research projects for the implementation of solution algorithms for complex systems of nonlinear equations (*e.g.* [74, 75, 82, 102, 125, 151, 159]). For a large number of applications grid independent convergence is observed, although a mathematical theory to support this does not exist, even for simple model problems. The convergence does not display the favourable quadratic convergence of the Newton method, and has similar linear convergence properties as the linear multigrid method. Justification for this is given in Subsection 4.2.3 as well as Chapter 5.

FAS is a common implementation of a nonlinear multigrid method and will be the focus of this thesis, as the implementation of FAS is one of the simplest of a nonlinear multigrid method and the results obtained are often optimal. A generalisation of FAS is the Nonlinear Multilevel Method (NMLM) of Hackbusch [85] given in Algorithm 4.4, where a scalar parameter s_j is introduced. This parameter is chosen depending on the current the right-hand side, as some function $\sigma(j, f_j)$. The choice of this is not of impor-

tance here and is discussed in more detail in Section 7.6. The interested reader may find a discussion of an appropriate choice of s_j in [87]. Strictly speaking Algorithm 4.4 is not

Algorithm 4.4 NMLM

Require: A_j, A_{j-1}, S_j, P_{j-1}^j, R_j^{j-1}

function NMLM($j, u_j, f_j, \gamma, \nu_1, \nu_2$)
  $u_j \leftarrow S_j^{\nu_1}(f_j, u_j)$ ▷ *Perform ν_1 pre-smooths*
  $u_{j-1} \leftarrow R_j^{j-1} u_j, \quad \tilde{u}_{j-1} \leftarrow u_{j-1}$
  $f_{j-1} \leftarrow R_j^{j-1}(f_j - A_j(u_j))$
  $s_{j-1} \leftarrow \sigma(j-1, f_{j-1})$
  $f_{j-1} \leftarrow A_{j-1}(u_{j-1}) + s_{j-1} f_{j-1}$
  if $j = 2$ **then**
    $\tilde{u}_{j-1} \leftarrow A_{j-1}^{-1}(f_{j-1})$
  else
    for $i = 1, \ldots, \gamma$ **do** ▷ *Approximate correction on coarse grid*
      $\tilde{u}_{j-1} \leftarrow$ NMLM($j-1, \tilde{u}_{j-1}, f_{j-1}, \gamma, \nu_1, \nu_2$)
  $e_{j-1} \leftarrow (\tilde{u}_{j-1} - u_{j-1}) / s_{j-1}$
  $u_j \leftarrow u_j + P_{j-1}^j e_{j-1}$
  $u_j \leftarrow S_j^{\nu_2}(f_j, u_j)$ ▷ *Perform ν_2 post-smooths*
  return u_j

the method presented by Hackbusch, which requires a sequence of approximations $\hat{u}_{j-1}$, $j = 2, \ldots, J$ and corresponding s_j which serve as an initial approximation to take on each coarse grid and an appropriate scaling parameter s_j such that an appropriate coarse grid correction is obtained. The significance of the parameter s_j will be made clear in the next section, which discusses similarities between a Newton-MG and Nonlinear Multigrid method.

4.2.3 Similarity between Nonlinear Multigrid and Newton's Method

In this section we establish a connection between nonlinear multigrid methods and Newton's method that aids in considering how a nonlinear multigrid iteration can be expected to perform. The ideas in this subsection are developed from a discussion of NMLM in [150, §3]. The discussion here focuses on the comparison between Newton and NMLM, and extends the result to also cover FAS. As far as we are aware this is the first time a discussion of this form has been given.

We consider again the solution of nonlinear Equation (4.36) with solution $u_j^* \in \mathcal{V}_j$

unique in the ball $B_{u_j^*}(\delta)$. Let

$$\begin{aligned} r_{j-1} = R_j^{j-1} r_j = R_j^{j-1}(f_j - A_j(u_j)) \\ u_{j-1} = R_j^{j-1} u_j. \end{aligned} \tag{4.40}$$

In order for the notation to be clearer let $DA(u)$ represent the Frèchet derivative of A at function u. As in the previous section the operator $S_j(f_j, u_j)$ represents the action of a nonlinear smoothing iteration.

First we consider the calculations that are made in a two-grid FAS iteration, in which the coarse grid problem is solved exactly. Let $C_j(f_j, u_j)$ represent the FAS coarse grid correction operator, which is a combination of the non-smoothing steps in Algorithm 4.3. In the case of a two-grid iteration the coarse grid correction operator is given by

$$C_j(f_j, u_j) = u_j + \left[A_{j-1}^{-1}\left(A_{j-1}(u_{j-1}) + r_{j-1}\right) - u_{j-1}\right]. \tag{4.41}$$

For the NMLM (see Algorithm 4.4) the exact coarse grid correction operator is given by

$$C_j(f_j, u_j) = u_j + \frac{1}{s_{j-1}}\left[A_{j-1}^{-1}(A_{j-1}(u_{j-1}) + s_{j-1} r_{j-1}) - u_{j-1}\right]. \tag{4.42}$$

In this form it is difficult to see a connection with a Newton iteration. To make the connection clearer the Newton iteration is presented in a form given in [150]. We note first of all that an application of the chain rule to $A^{-1}(A(u)) = u$ gives

$$(DA(u))^{-1} = DA^{-1}(A(u)). \tag{4.43}$$

Using (4.43) a single iteration of a Newton iteration reads

$$\begin{aligned} u_j^{(i+1)} &= u_j^{(i)} + \left(DA_j(u_j^{(i)})\right)^{-1}\left(f_j - A_j(u_j^{(i)})\right) \\ &= u_j^{(i)} + DA_j^{-1}(A_j(u_j^{(i)}))(r_j^{(i)}). \end{aligned} \tag{4.44}$$

To see the connection to the nonlinear multigrid methods we consider that instead of performing the linearisation in the second line of (4.44) we approximate the derivative DA_j^{-1} using a divided difference. This may be done using either a Frèchet or a Gateaux difference, as described below.

A Frèchet difference uses the definition of the Frèchet derivative (see (3.65)) to approximate

$$\begin{aligned} DA^{-1}(A(u))(r) &\approx A^{-1}(A(u)+r) - A^{-1}(A(u)) \\ &= A^{-1}(A(u)+r) - u, \end{aligned} \tag{4.45}$$

with equality in the limit as $\|r\| \to 0$. Inspection of (4.45) and (4.41) shows that the two-grid FAS method uses a correction term that approximates a Newton correction on the coarse grid. The more accurate the approximation to the nonlinear problem, and the smaller the residual, the more accurately the coarse grid correction approximates a Newton correction. This demonstrates that for approximations close to the exact solution, in a ball of guaranteed convergence of the Newton iteration, the FAS iteration will perform very well. For an approximation further from the exact solution it is not clear mathematically what effect this should have on the convergence, although there will not be an approximate linearisation of the nonlinear equation in the Newton sense. Results in Chapters 7 and 8 suggest that for an approximation far away from the exact solution the method will not perform well. It is also clear that FAS cannot gain the same quadratic convergence as a Newton iteration. This is due to the fact that an FAS iteration is a generalisation of the linear multigrid method, in the sense that algebraically the formulations are equivalent for a linear problem. Linear multigrid converges linearly so it is not possible for the nonlinear method to converge super-linearly, as what is true for the nonlinear method must be true for the linear method. The convergence rate of a multi-grid FAS iteration is bounded by the two-grid FAS convergence rate [88, 150]. For a more detailed discussion of the convergence of FAS see Chapter 5.

An alternative to approximating the derivative of the nonlinear operator using a Fréchet difference is to use a Gateaux difference. This uses the definition of the Gateaux derivative (see (3.64)) to approximate

$$\begin{aligned} DA^{-1}(A(u))(r) &\approx \frac{1}{s}\left[A^{-1}(A(u)+sr) - A^{-1}(A(u))\right] \\ &= \frac{1}{s}\left[A^{-1}(A(u)+sr) - u\right]. \end{aligned} \tag{4.46}$$

Using this approximation of the derivative in the coarse grid correction leads to the NMLM two-grid iteration (see Algorithm 4.4 and (4.42)). As can be seen, the choice of s is important in determining the accuracy with which the derivative is approximated, and depends on the value of the right-hand side on the coarse grid (which is also depen-

dent on the current approximation). Given a suitable choice for the parameter s it can be seen that the approximation to the derivative should be better for the NMLM than for the FAS iteration. This leads to increased robustness of the method as shown in [86]. As mentioned previously, though, the calculation of an appropriate scalar s is non-trivial and hence we prefer the FAS iteration. The FAS iteration is especially useful when the initial approximation is close to the exact solution, as is often the case for implicit time-stepping applied to time-dependent problems. In fact an improved convergence behaviour of the FAS iteration is observed for time-dependent problems when smaller time-steps are used compared to larger time-steps.

In this chapter we have given an introduction to the standard linear and nonlinear multigrid methods. There are many variants of both linear and non-linear multigrid methods which take advantage of properties for specific problems (*e.g.* the hierarchical basis method [9]; monotone multigrid [80, 109, 110]; p-multigrid [5]; cascadic multigrid [20]; and algebraic multigrid, and variants thereof [98, 117, 136, 180], in the linear case, and non-smooth Newton-MG [81] and algebraic nonlinear multigrid [66] in the nonlinear case). These methods all use coarser spaces (not necessarily geometric spaces) to capture information not approximated well in fine spaces. Whilst these variants are useful, their application tends to be for problems for which the standard multigrid algorithms presented in this section are unsatisfactory. As previously noted the multigrid methods outlined here provide an active research topic and give optimal results for the solution of many PDEs of elliptic and parabolic type. This thesis will use problems for which the standard multigrid algorithms are appropriate, so a discussion of the wide range of variants will not be given. In the next chapter an outline of the convergence theory for standard linear and nonlinear multigrid iterations is presented. A comparison of the theory for the nonlinear multigrid methods is made, and in Chapter 6 a more detailed, and novel, comparison of the implementations of Newton-MG and FAS is given before demonstrating results applied to a sample of nonlinear problems in Chapters 7 and 8.

Chapter 5

Convergence of Multigrid Iterations

Since the inception of multigrid methods, starting with geometric multigrid due to Fedorenko [72] and Bakhvalov [7], multigrid methods have enjoyed a lot of success in research. The basic concept is powerful, and many variants of multigrid methods have been developed that take advantage of being able to solve a problem in a space with (much) reduced dimension. Often a multigrid method will have an observed rate of convergence which is independent of any mesh parameters. Convergence proofs should confirm this behaviour, which requires demonstrating that the contraction number

$$\gamma_J \equiv \|M_J\| \tag{5.1}$$

or the spectral radius $\rho(M_J)$ for the multigrid error propagation operator M_J on grid Ω_J is uniformly bounded away from one as $h_J \to 0$ in a suitable norm, where

$$h_J = \max_{T \in \mathcal{T}_J} h(T),$$

as defined in Section 2.1. Then grid independent convergence is characterised by

$$\lim_{h_J \to 0} \gamma_J \leq C < 1 \qquad \text{or} \qquad \lim_{h_J \to 0} \rho(M_J) \leq C < 1. \tag{5.2}$$

Here, and in the following, C will denote a generic positive constant independent of mesh parameters, which is not necessarily the same each time it is used. Convergence proofs that try to approximate the quantity γ_J or $\rho(M_J)$ are called *quantitative*, and proofs that focus on characterising the conditions under which (5.2) holds are called *qualitative*. Results from both methods of proof give valuable information for a multigrid practitioner. A quantitative analysis is valuable from an implementation point of view, where it can be observed what effect changing a component in the multigrid iteration will have on the speed of convergence of the overall method. Qualitative analyses indicate which problems are appropriate to be solved using a multigrid iteration, but importantly also characterise when a particular multigrid solver is unsuitable for a problem. In this case proofs may indicate which multigrid components need to be changed in order to obtain a convergent iteration.

In this chapter an outline of the theory for the convergence of linear and nonlinear geometric multigrid algorithms is given. We are most interested in qualitative convergence proofs, although a brief discussion of quantitative analyses will be given. Qualitative proofs are of more interest as these results are more general, and give an idea in which situations a multigrid iteration can be expected to converge, which is more useful for this study than an estimation of the convergence factors. The convergence factors for Newton-MG and FAS are compared in numerical experiment in Chapters 7 and 8. The first sections in this chapter outline the convergence theory for linear multigrid methods, which is much better understood than the nonlinear case. The linear theory presented may be re-used in the proofs of convergence of Newton-MG methods. The chapter concludes with a presentation of convergence theory for nonlinear multigrid methods and a comparison of the theory for nonlinear and Newton-MG methods.

5.1 Linear Multigrid

There exists a wealth of literature for the convergence of linear geometric multigrid methods (*e.g.* [28, 30, 32, 33, 42, 133, 134, 160, 177, 196], amongst many others). In this section we do not seek to present a detailed description of all of this literature. Instead key results from selected papers are presented. In practice when a multigrid method works well it is usually sufficient to use a V-cycle to gain convergence, and it is often the most computationally efficient algorithm to use. Therefore the results presented will mostly concern the convergence of the V-cycle. We also concentrate on the case of a finite element discretisation, as this allows for the use of powerful theoretical tools from Hilbert space theory. Where appropriate it will be stated where the theory also applies to different discretisation

methods.

Unless otherwise stated the convergence theory presented here is for symmetric positive definite operators arising from the discretisation of (typically) elliptic second order PDEs. In d dimensions these generally take the weak form

$$\begin{gathered}\int_\Omega \left[\sum_{i,j=1}^{d} a_{ij}(\vec{x})\frac{\partial u}{\partial x_i}\frac{\partial v}{\partial x_j} + b(\vec{x})uv\right] \mathrm{d}\vec{x} = \int_\Omega fv\,\mathrm{d}\vec{x}, \quad \vec{x}\in\Omega \\ u = 0, \quad \vec{x}\in\partial\Omega\end{gathered} \tag{5.3}$$

where it is assumed $a_{ij}, b \in C^1(\Omega)$, $a_{ij} = a_{ji}$ and $b \geq 0$. The tensor $A = [a_{ij}]_{i,j=1}^d$ is called the *coefficient* function and is assumed to be symmetric positive definite. The analysis is much simplified in this case as the energy inner product

$$a(u,v) \equiv \int_\Omega \left[\nabla u^T A(x)\nabla v + b(x)uv\right]\,\mathrm{d}x \tag{5.4}$$

induces a norm. The case of homogeneous Dirichlet boundary conditions is investigated for simplicity and the extension to non-homogeneous conditions is simple. Neumann conditions may also be considered, but some Dirichlet condition is required for a solution to be unique. Under the given assumptions the Lax-Milgram theorem [39, §2.7] guarantees that a unique solution u^* exists. Letting

$$F(v) = \int_\Omega fv\,\mathrm{d}x \tag{5.5}$$

the solution is in the space $H_0^1 \cap H^{1+\alpha}$ when $F \in H^{-1+\alpha}$ for $\alpha \in (1/2, 1]$ [38]. Early multigrid proofs of the V-cycle needed to assume *full elliptic regularity*, *i.e.* $\alpha = 1$. Geometrically this confines the domain to being convex [38]. These regularity assumptions were dropped due to the work by Oswald, Bramble, Pasciak, Wang and Xu [30, 140, 141, 196], which develops the fundamental theory for the convergence of multigrid methods. The review by Yserentant [202] is a good overview of this fundamental theory. A brief discussion of this fundamental theory is given in Subsection 5.1.3 which includes some more recent technical advances. Before that a discussion of other methods of proof of convergence for multigrid methods is presented in Subsections 5.1.1 and 5.1.2 below.

5.1.1 Local Fourier Analysis (LFA)

Local Fourier analysis (LFA), also called local mode analysis, was the first method developed to perform the analysis of the convergence of linear multigrid methods. It was first introduced by Brandt [32] and remains popular (*e.g.* [94, 118, 133, 134]) due to its quantitative estimation properties. Local Fourier analysis is a tool that is used to aid in the investigation of multigrid methods, rather than a rigorous mathematical framework [177, 190]. It arose from rigorous Fourier analysis, in which it is found that Fourier modes forming a basis for functions defined on a grid coincide with the eigenfunctions of certain discrete operators [177, §3]. With a suitable choice of smoothing and grid transfer operators (see [33, 34, 42, 177]) low dimensional subspaces of basis functions are found to be invariant under the multigrid operator. Functions in a single subspace are known as *harmonics* [177, pg. 86] and correspond to functions that are indistinguishable (up to a change of sign) on a coarse grid (for an example see Figure 4.3).

Using the spaces of harmonics a multigrid operator can be represented in block diagonal form [34, 177]. Therefore, an investigation of the spectral properties of the multigrid cycle is performed by analysing low dimensional spaces (in the case of a two-grid cycle), see [177, §3]. In the rigorous analysis the Fourier basis must conform to the boundary conditions, which only occurs for very restricted cases. For example, mixed boundary conditions, non-constant coefficient operators and non-rectangular domains cannot be treated in general by the rigorous analysis. LFA allows Fourier analysis to be applied in more general cases. An assumption is made that an operator is linear, constant-coefficient and defined on an infinite domain (*i.e.* boundary effects are ignored). Ignoring boundary effects does not have such a large impact on the accuracy of the analysis, and can be considered as an assumption that boundary conditions have been captured sufficiently accurately by a solution method. LFA then gives the asymptotic convergence factor for the two-grid iteration [177, §4]. Convergence of the W-cycle follows immediately from this using a simple recursion formula [177, pg. 78], whereas it is not necessarily true that the V-cycle will converge ([130] gives an example), although situations in which V-cycle convergence will occur can be characterised [133]. To consider multigrid convergence it is possible to perform more than a two-grid analysis. However, the amount of work required to perform the analysis grows quickly, and the complexity of the problem becomes impractical, although some analyses have been performed in this case [191].

In the constant coefficient linear case the convergence estimates may be sharp [123, 134] and in the variable coefficient and non-linear case repeated LFA for multiple constant coefficient cases can give an approximation to the convergence rate [177, 190]. Systems of equations may also be analysed [118]. However, it is suggested that LFA be used

by practitioners implementing multigrid rather than mathematicians analysing multigrid [177, 190]. This is in particular because LFA is not a rigorous theory, but it also requires elliptic regularity of the solutions, which is not the case for other qualitative methods of proof. These are discussed in the next sections.

5.1.2 Hackbusch's Method

Hackbusch has been an influential member of the multigrid community since the early 1980's. His 1985 manuscript [85] remains one of the comprehensive texts on multigrid methods, in which linear and nonlinear iterations are discussed. The convergence theory introduced in [85] involves proving two properties - the *smoothing property* and the *approximation property* [85, §6.1.3]. This method of proof is often called the general multigrid theory, but is called Hackbusch's method in this thesis without confusion. Hackbusch's method has proved useful and many proofs use the theoretical tools presented, although the exact definition of the smoothing and approximation properties varies depending on the proof performed, (see [24, 36–38, 85, 123, 202]). However, every definition of the smoothing property formalises the requirement that the smoother should be effective at removing high frequency error components, and every definition of the coarse grid correction property formalises the requirement for the correction on the coarse grid error to approximate the fine grid error. The quality of grid transfer and coarse grid operators are implicitly contained in this property. The theory is presented in a general Banach space setting [85, §§6-9] and applies equally to linear [85, §§6-7] and nonlinear [85, §9] problems. However, the assumptions to be satisfied are difficult to prove for individual problems if the function spaces considered are not Hilbert spaces.

Results for the convergence of the W-cycle for general symmetric positive definite operators and the V-cycle for problems with full elliptic regularity have existed in this framework since the early '80s (see [24, 201] and the closely related [10]). Convergence results for the V-cycle without any regularity assumptions were gained in [36, 37]. All the proofs include the case of mildly varying diffusion coefficients on the domain. To the best of our knowledge the case of highly varying and discontinuous coefficients has not been properly studied within this framework. However, this generic method of proof is applicable to problems where the function spaces are *not* nested and the coarse grid operators are not Galerkin operators (see (4.20)) [27]. Although the proofs using Hackbusch's method are mostly qualitative, convergence bounds derived in this framework can be directly used also in LFA and it can be shown that the bounds cannot be improved upon [123, 134].

5.1.3 Subspace Correction Theory

Subspace correction and domain decomposition methods (see [165, 176]) began to gain in popularity during the 1990's. Due to the work by Bramble, Pasciak, Wang and Xu [30, 31] it was demonstrated that multigrid methods are part of the wider class of domain decomposition and subspace correction methods. From this it was possible to analyse the multigrid iteration in the same abstract framework as other subspace correction methods (including the Jacobi and Gauss-Seidel iterations, and variations thereof). The review by Xu [196] is the standard reference on the matter, although [202] gives a description of the theory applied to multigrid methods only.

The analysis requires that the operator be symmetric positive definite (spd), or be dominated by the symmetric part of the operator. This is in contrast to the previous methods of analysis which can theoretically deal with non-symmetric and indefinite operators, although these are difficult to analyse. LFA allows for quantitative estimates to be gained for non-symmetric and indefinite forms without a qualitative insight into why the methods should converge.

The subspace correction theory is that which is of most interest in this thesis. We point the reader to proofs of the main results, while giving these without the technical theory since the implications of the proofs are of greater interest within the scope of this work. The most important results in this field are those by Bramble, Pasciak, Wang and Xu [30, 31, 196]. These were the first to demonstrate grid-independent convergence of the multigrid V-cycle without full elliptic regularity for operators of the form (5.3) with constant or mildly varying coefficients [202, Theorem 5.1], [196, Theorem 6.10]. The grids in the hierarchy are assumed quasi-regular with respect to the triangulations and the function spaces are nested. Implicit in the proofs are that some Sobolev type inequalities [196, §5] hold on the domain, and the operators and functions involved in the multigrid iteration are required to satisfy *strengthened Cauchy-Schwarz* inequalities [202, Equations (5.3), (5.4)], which give an important relationship between the smoothers and projection operators on each grid level (see [196, §6.1]). These assumptions are satisfied for standard piecewise linear nodal basis finite element functions, and a large class of other discretisations. The Cauchy-Schwarz estimates are satisfied in the case that an spd smoother is used. Optimal convergence results can be extended to the convergence of indefinite and non-symmetric operators so long as the symmetric positive definite part of a problem dominates [26, 49, 51]. To the best of our knowledge further advancements in the convergence theory for indefinite and non-symmetric operators have not been made within the subspace correction framework. However, the result of [26] using a *compact perturbation* is useful and one that will be used in Chapter 7. The proof is not recreated

here, but the reader is directed to [26, Theorem 5.2].

In the years since [196] the convergence theory has been advanced to include highly varying coefficients and different families of elements, as well as mesh adaptation. Mesh independent convergence can be gained in the case of adaptively refined meshes quite easily [29, 30], although development of optimal order algorithms is a lot more technical see [193, 199]. Mesh adaptation is not considered in this thesis as this is a problem dependent, not solver dependent, enhancement. Thus a comparison of different solution algorithms, as presented in this thesis, is simplified by not considering this enhancement. What is of interest is the case of a highly varying, or discontinuous, coefficient on a domain which we present in some detail below. We return to this in a discussion of results in Chapter 7. For ease of presentation we consider only the case of a distribution of piecewise constant functions on a domain, although the results are trivially extended to piecewise smooth functions satisfying a necessary assumption, called *quasi-monotonicity* [143, 144, 156], which is discussed below.

We consider the application of multigrid to the operator equation

$$\int_{\Omega_J} \alpha(\vec{x}) \nabla u \nabla v \, \mathrm{d}\vec{x} = \int_{\Omega_J} f v \, \mathrm{d}\vec{x} \tag{5.6}$$

where Ω_J is a discrete domain split into a triangulation $\mathcal{T}$, and $\alpha(\vec{x}) \geq C > 0$ is piecewise constant, with finite number s of different coefficient values. Let γ_i, $i = 1, \ldots, s$ be the regions of different coefficient values on the domain. Assume that these regions are polygonal, and that any discontinuities between two sub-domains are aligned with the triangulation $\mathcal{T}$. It is assumed that $\gamma_i \cap \gamma_j = \emptyset$ for $i \neq j$ and that $\overline{\Omega} = \bigcup_i \overline{\gamma}_i$. An example of this is shown in Figure 5.1 for $s = 5$. Practical experiments demonstrate optimal multigrid

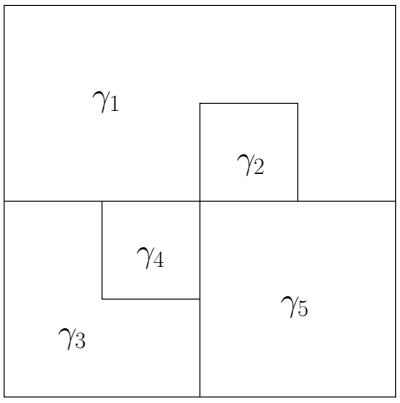


Figure 5.1: Square domain separated into polygonal sub-domains γ_i, $i = 1 \ldots 5$, of different coefficient value.

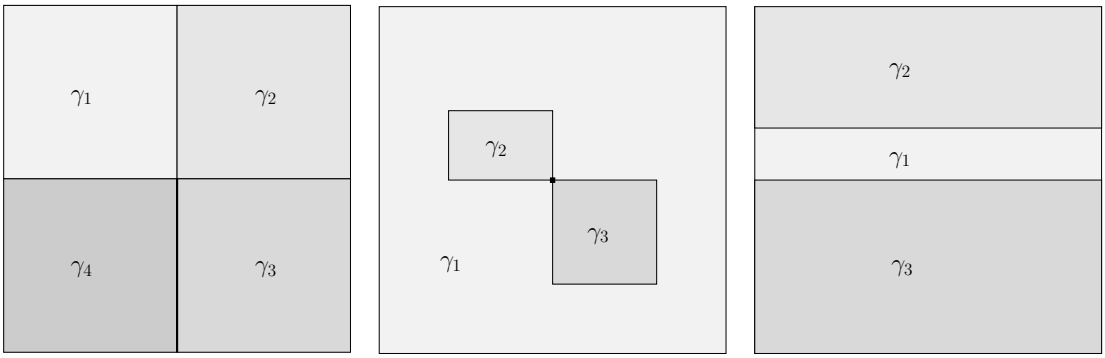
convergence for certain arrangements of coefficients [156]. Convergence proofs for this case have received recent interest - see [197, 206, 207] in which quasi-optimal results are proved. Here quasi-optimal refers to the fact that there is a mild dependence on mesh parameters and an $\mathcal{O}(N\log(N))$ running time. An algebraic treatment is given in [181], which can be useful in the cases where a geometric proof fails. In the following we summarise the geometric result proved in [156], which will be used in the discussion of results in Chapter 7. It is assumed that standard conforming linear finite element spaces are used. The finest space must resolve all discontinuities in α for the discrete solution to appropriately approximate the continuous solution, although coarse grids need not be aligned with discontinuities. The technical details of the proof are left to [156] and we give the conditions that need to be satisfied in order to prove optimal and quasi-optimal convergence bounds. The most important condition to be satisfied is quasi-monotonicity, which is defined below after some introductory notation. Let $\{\varphi_k\}$, $k = 1, \ldots, N$ for $N = \#\mathcal{N}$ (with $\mathcal{N}$ the set of nodes on grid Ω_J) be the standard continuous piecewise linear nodal basis on grid Ω_J. We define

$$\omega_k \equiv \text{supp}(\varphi_k) \qquad \text{and} \qquad \omega_T \equiv \bigcup_{\{k:\omega_k \cap T \neq \emptyset\}} \omega_k \tag{5.7}$$

for $T \in \mathcal{T}$ an element on grid Ω_J. Let ω_T be partitioned into regions $(\gamma_1^T, \ldots, \gamma_s^T)$ of different constant coefficient such that $\alpha|_{\gamma_i^T} \leq \alpha|_{\gamma_j^T}$ for $i < j$. Consider a directed graph $\mathcal{G}^T = (\mathcal{P}^T, \mathcal{E}^T)$ for vertex set $\mathcal{P}^T = \{\gamma_1^T, \ldots, \gamma_s^T\}$ and edge set $\mathcal{E}^T = \{(\gamma_i^T, \gamma_j^T) \mid \gamma_i^T \leq \gamma_j^T \text{ and } \gamma_i^T \cap \gamma_j^T \neq \emptyset\}$. Then we have the following definition of quasi-monotonicity:

Defintion 5.1. *The coefficient α is type m ($m = 0, 1$) quasi-monotone on ω_T if for every $\gamma_i^T \neq \gamma_s^T \in \mathcal{P}^T$ there exists a path $p \in \mathcal{G}^T$ from γ_i^T to γ_s^T such that the interface $\gamma_i^T \cap \gamma_j^T$ for each edge $(\gamma_i^T, \gamma_j^T) \in p$ is an m-dimensional manifold.*

This definition simply states that there exists a path from every region of constant coefficient in ω_T to the region of highest coefficient γ_s^T which passes through regions of increasing coefficient. To aid in the understanding of the definition examples of quasi-monotone coefficient in two dimensions on an entire domain (rather than restricted to ω_T) are given in Figure 5.2. Letting Γ denote the Dirichlet boundary of Ω then the related concept of Γ-quasi-monotonicity is satisfied if there exists a path from any γ_i^T to the Dirichlet boundary whenever $\bar{\omega}_T \cap \Gamma \neq \emptyset$. The following result (under standard assumptions) is proved in [156, Theorem 6.1] and will be referred to in the discussion of the convergence of nonlinear multigrid methods in Chapter 7.



(a) Type 1 quasi-monotonicity conditions satisfied. All paths to γ_4 pass through interfaces of dimension 1

(b) Type 0 quasi-monotonicity condition satisfied. The interface between γ_2 and γ_3 is a zero-dimensional manifold

(c) Quasi-monotonicity condition not satisfied

Figure 5.2: Examples of domains which do or do not satisfy a quasi-monotonicity condition. m-dimensional manifolds are shown in bold.

Theorem 5.1. *Let Equation* (5.6) *be defined on a hierarchy of grids* $\Omega_1, \ldots, \Omega_J$*. The energy norm of the multigrid error propagation operator* M_J *is bounded by*

$$\|M_J\| \leq 1 - \frac{1}{Jc} \tag{5.8}$$

where c *is: a) constant when the support of the basis functions on every level satisfy either a type 1 quasi-monotone or* Γ*-quasi-monotone condition; b)* $\mathcal{O}(1 + \log(H_T/h_T))$ *for* $H_T = \operatorname{diam}(\omega_T)$ *and* $h_T = \max_{T \in \omega_T} \operatorname{diam}(T)$ *when a type 0 quasi-monotone or* Γ*-quasi-monotone condition holds; c) dependent on* $\max_T \max_{x,y \in \omega_T} \alpha(x)/\alpha(y)$ *when no quasi-monotonicity is satisfied, where* $\max_T$ *is taken over all elements in the grid hierarchy.*

Proof. See proof of [156, Theorem 6.1] □

This theorem simply states in which situations a multigrid iteration can be expected to converge independently of the size of the jumps in coefficient as well as of mesh parameters, and should be used as a tool for practitioners to be able to justify why a multigrid iteration may or may not perform well for varying arrangements of coefficient. Note that, although Figure 5.2 shows arrangements of coefficient in which the entire domain is not quasi-monotone, Theorem 5.1 only requires that quasi-monotonicity holds *locally* over an extended support of each basis function (*i.e.* ω_T).

The material covered here is enough to read the rest of this thesis, although many other technical results do exist. We now give a background of convergence theory for the

nonlinear iterations in the next section before moving on to a detailed comparison of the algorithmic properties of Newton-MG and FAS in Chapter 6.

5.2 Nonlinear Problems

In this thesis we are particularly interested in the convergence properties of nonlinear methods related to the general elliptic problem in weak form given by

$$\begin{gathered}\int_\Omega f(\vec{x},u,\nabla u)\nabla u\nabla v\,\mathrm{d}\vec{x}=\int_\Omega gv\,\mathrm{d}\vec{x},\quad \vec{x}\in\Omega\\ u=0,\quad \vec{x}\in\partial\Omega\end{gathered}\tag{5.9}$$

and the linearisation

$$F(u)(v,w)=\int_\Omega Df(\vec{x},u,\nabla u)(w)\nabla u\nabla v+f(\vec{x},u,\nabla u)\nabla w\nabla v\,\mathrm{d}\vec{x}\tag{5.10}$$

where Df is the derivative of operator f with respect to u, and ∇u is taken as dependent on u. We assume that f is bounded in terms of $\vec{x}$, and is C^1 in u. In the case that f is a function solely of ∇u and is monotone increasing $F(u)$ gives a symmetric positive definite bi-linear form. Otherwise the operator will contain some component that acts like a convective term. For some examples of model problems see Chapters 7 and 8.

To the best of our knowledge there exists no valid theory that demonstrates mesh independent convergence for either Newton-MG methods or for nonlinear multigrid variants for problem (5.9). This is because the natural setting for the problem (5.9) is in the space $W^{1,p}$, $p>2$ [39, pg. 235], [107], which is not a Hilbert space. $W^{1,p}$ is the usual Sobolev space of functions with first derivative bounded in L^p norm. In the context of Newton iterations this is a problem as the convergence of linear iterations, required to demonstrate that the linear system of equations arising from the linearisation is solved accurately enough, is performed in a Hilbert space setting. In [107] a novel iterative method is designed to converge in the $W^{1,p}$, $p>2$ norm to establish a mesh independent approximate Newton iteration. As far as we know this is the only paper demonstrating a mesh independent convergence of a problem with high order nonlinearity, but it is not applicable to Newton-MG. Recently Karátson [104] demonstrated that problems with a high order nonlinearity *cannot* satisfy assumptions that are necessary for current theory to demonstrate mesh independent quadratic convergence (see [104, Remark 4.4]). Therefore the convergence analysis is not currently possible for a Newton-MG method for (5.9), al-

though results in Chapters 7 and 8 show that at least linear mesh independent convergence of Newton-MG (and FAS) iterations is possible. Note that a *necessary*, but not *sufficient*, condition for the convergence of a Newton iteration is that the linear iteration should converge. This allows for predictions to be made when a nonlinear iteration will not converge as we can characterise the situations in which the linear inner iteration will not converge, given the solution method to be used.

There are no valid convergence proofs for the case of a nonlinear multigrid iteration for (5.9). Results for the convergence in the nonlinear case are restricted to semi-linear problems of the form

$$\int_{\Omega} f(\vec{x})\nabla u\nabla v + b(\vec{x},u)uv \,\mathrm{d}\vec{x} = \int_{\Omega} gv \,\mathrm{d}\vec{x} \tag{5.11}$$

with many assumptions not applicable to most application areas [88, 149, 150, 195]. The most sophisticated results are from 1989 due to Hackbusch and Reusken [88]. The method of proof is technical and it is difficult to see how the method of proof can be applied in a more general situation. In fact, there appears to have been no advance in the convergence theory since this time. Results for the convergence of Newton methods for (5.11) are much stronger than those for nonlinear multigrid methods (*e.g.* [1, 45, 58, 189]) and have the advantage that the analysis of the convergence of the inner iteration is completely separate from the analysis of the convergence of the outer iteration under the assumption that the inner iteration is solved 'accurately enough'. This allows for any suitable linear iteration to be used as part of a Newton iteration without the nonlinear analytical tools needing to be changed.

Note that an implicit assumption that has been made here, which is standard, in that we assume that some initial guess u^0 to the solution of (5.9) has been given which is 'close enough' to the exact solution. This is an essential part of any proof of convergence for nonlinear problems, although the choice of a suitable u^0 can be a non-trivial problem for an implementation of a nonlinear solution algorithm.

Using the material presented so far in this thesis we are ready to begin a direct comparison of Newton-MG and FAS iterations applied to second order nonlinear differential operators. Note that much of the discussion presented is independent of the specific problem being solved. Chapter 6 introduces a theoretical bound on the computational effort per multigrid V-cycle for each nonlinear iteration, which is verified to be sharp using numerical experiments. The performance in terms of robustness and efficiency of the multigrid iterations is demonstrated for a model nonlinear scalar equation in Chapter 7, and is extended to more complex time dependent nonlinear equations in Chapter 8.

Chapter 6

Running Time: FAS vs Newton-MG

In this chapter a new theoretical bound on the running time of an implementation of a Newton-MG and FAS iteration is presented which allows for a direct comparison of the time complexity of a single V-cycle iteration. The framework in which the comparison is performed is novel and aids in highlighting the computational benefits in solving linear, instead of nonlinear, problems as part of an iterative solution method. The focus will be on a finite element discretisation scheme, although the framework is independent of the discretisation method used. The discussion in this chapter considers the implementation of each algorithm in order to derive bounds of the computational complexity. Care has been taken to ensure that the implementation is efficient, and where appropriate sufficient information is given for the reader to recreate an implementation that achieves the same results as presented in this thesis. In this respect it should be noted that all grids used are assumed simplicial. In order for a reader to compare their own implementations we also specify the hardware on which numerical experiments were performed. All experiments were run in a single thread on a CPU, and were all on the same desktop machine using a 64-bit architecture. The hardware used was the following: Intel Xeon E3-1270 CPU (3.4GHz, 8M Cache, 4 core, 8 thread); 8GB RAM (1333MHz); 1TB SATA hard drive (Seagate Barracuda, 7200RPM). No GPUs were used.

There are several papers in which FAS multigrid is compared against Newton-MG for individual application areas with a particular discretisation [86, 120, 121, 127, 167]. Most of these compare the running times as well as the convergence factors of the iterations, and come to the same conclusion - that the execution time for Newton-MG methods is less

than for FAS multigrid [120, 121, 127, 167]. For a finite difference discretisation the running times of the algorithms are shown to be much closer [91], which is also highlighted in Section 6.4. Previous comparisons have considered problems that are quite complex, including systems of equations, within a single finite element discretisation. Results in Chapter 8 are similar to those in previous papers, but the more useful results are obtained for model problems, which allow for an upper bound on the running times to be gained and give a *quantitative* estimate of the minimum increase in performance to be gained from an *efficient* implementation of Newton-MG over FAS. In this chapter we introduce the framework for comparing the running times without giving any quantitative results, which are left for Chapters 7 and 8. The framework presented is independent of the discretisation scheme used and the number of dimensions in which an operator is set. In the next section a characterisation of the running time for each algorithm is given, followed by a comparison of the bounds for the case of the standard piecewise linear finite element nodal basis. These bounds hold directly in the case of lowest order Raviart-Thomas elements, and extensions to other finite elements is possible. The estimates obtained in this chapter are demonstrated to be sharp using model problems in Chapter 7. To finish the chapter it is shown how the framework extends to other discretisation schemes by considering a model problem discretised using a finite difference method. The bounds gained for the finite difference discretisation are not tight and no results are presented, although it should be clear that the theory will still be valid for this case.

6.1 Characterisation of Running Time

In this section we consider the Newton-MG and FAS iterations applied to a general nonlinear problem. A theoretical bound on the running time is developed that allows a direct comparison between the two methods and makes it clear why Newton-MG methods execute faster than FAS multigrid methods. A note to make is that [86] finds that the Nonlinear Multilevel Method (NMLM) of Hackbusch performs better than a standard Newton-MG iteration due to an increased stability of the method. This is to be expected, though, as NMLM is a global algorithm. However, the NMLM is not a practical algorithm, and this is discussed in more detail in Section 7.6.

The amount of computational effort required to perform each of the algorithms is characterised in terms of a *work unit*. One work unit (denoted W_j) is the amount of time required to calculate the nonlinear residual on grid Ω_j. The amount of work required depends on the number of dimensions in which we are working, as well as the discretisation and choice of smoothers. In this section we derive a formula for calculating the amount

of computational time required for a single linear or nonlinear multigrid γ-cycle, $\gamma \geq 1$, when using a generic finite element discretisation. Here $\gamma = 1$ gives a V-cycle, $\gamma = 2$ a W-cycle, *etc*. We conclude the chapter by giving a quantitative estimate on the running time of the algorithms for a two-dimensional piecewise linear nodal finite element basis.

Consider that a nonlinear problem has been discretised on a hierarchy of grids (see (4.2)) $\Omega_j \subset \mathbb{R}^d$, $j = 1, \ldots, J$, $d = 1, 2, 3$. We consider a finite element discretisation with basis $\{\varphi_i^j\}_{i=1}^{N_j}$ such that $\mathrm{span}\{\varphi_i^j\}_{i=1}^{N_j} = \mathcal{V}_j$, $j = 1, \ldots, J$ on a quasi-regular triangulation $\mathcal{T}_j$ on grid Ω_j. $\mathcal{V}_j$ is the finite element subspace associated with grid Ω_j. The superscript j for the basis functions is dropped when it is apparent from the context which grid is referred to. N_j now refers to the number of unknowns on the grid, rather than the number of non-Dirichlet nodes on the grid, as was previously introduced. The two numbers coincide when a piecewise linear nodal basis is used. We do not assume that a regular refinement strategy is used, but do assume that the refinement is such that there is a geometric progression for the number of nodes on each grid, *i.e*.

$$N_{j+1} \approx sN_j, \tag{6.1}$$

for scalar $s > 1$. We assume that a discrete nonlinear problem, given in weak form as

$$F_j(u_j, \varphi_i) = 0, \quad i = 1, \ldots, N_j \tag{6.2}$$

on grid Ω_j, is discretised using finite elements. We let N_T be the number of unknowns per element $T \in \mathcal{T}_j$, which is assumed equal for all T. The application of F_j can be calculated as a sum over the elements $T \in \mathcal{T}_j$ as follows:

$$F_j(u_j, \varphi_i) = \sum_{T \in \mathrm{supp}(\varphi_i)} F_j^{(T)}(u_j, \varphi_i), \tag{6.3}$$

where $F_j^{(T)}$ is the discrete operator restricted to element $T \in \mathcal{T}_j$. Let the Frèchet derivative of $F_j(u, \varphi)$ at u be denoted by

$$F_{u,j}(\psi, \varphi) = D\left[F_j(u, \varphi)\right](\psi) \tag{6.4}$$

and assume that this exists and is invertible for all $u \in \mathcal{V}_j$, $j = 1, \ldots, J$. Then the

Jacobian matrix

$$\left[F_{u,j}(\varphi_k, \varphi_i)\right]_{i,k=1}^{N_j} \tag{6.5}$$

exists and is invertible. For practical purposes we assume that the entries in the Jacobian matrix are calculated using numerical differentiation, as for complex problems this requires less computational time to calculate than using exact formulae for the derivatives[1]. The entry in row i, column k of the Jacobian matrix is constructed as follows:

$$F_{u,j}(\varphi_k, \varphi_i) \approx \sum_{T \in \text{supp}(\varphi_i) \cap \text{supp}(\varphi_k)} \frac{F_j^{(T)}(u_j + \epsilon\varphi_k, \varphi_i) - F_j^{(T)}(u_j, \varphi_i)}{\epsilon}. \tag{6.6}$$

The value $F_j^{(T)}(u_j, \varphi_i)$ is used in the calculation of the nonlinear residual, and can be re-used when calculating the entries in the Jacobian matrix. Hence, per simplex, when calculating the residual we can perform an extra local function evaluation at each of the N_T unknowns to obtain the element-wise contribution to the Jacobian matrix. For example, in d dimensions with a piecewise linear nodal finite element basis we require one local function evaluation to calculate the residual and an extra $d+1$ local function evaluations (per simplex) to calculate the contribution to the Jacobian matrix. Therefore the cost of calculating the Jacobian matrix and the residual is approximately $(d+2)W_j$. Using similar reasoning we see that calculating the diagonal entries of the Jacobian matrix requires just one extra local function evaluation per node on each element. Hence the cost to calculate the nonlinear residual and the diagonals of the Jacobian matrix is approximately $2W_j$. The diagonals of the Jacobian matrix are used in the nonlinear smoothing operator. In the general case, assuming that a forward or backward difference is used to approximate the numerical derivative, as in (6.6), the cost of calculating the Jacobian matrix and nonlinear residual is approximated by $(N_T+1)W_j$, whereas the cost of calculating the nonlinear residual and the diagonals of the Jacobian matrix remains at $2W_j$.

In the case that a central difference formula is used two extra local function evaluations are required per unknown on an element, and so the cost of calculating the residual and the Jacobian is approximated by $(2N_T+1)W_j$ and the cost of calculating the nonlinear residual and the diagonals of the Jacobian matrix is approximated by $3W_j$. The case of using a forward difference is used in the following discussion, although the extension to the central difference case simply requires a substitution of appropriate timing estimates.

[1]Whilst using a numerical derivative is common in practice issues may arise if large relative errors are introduced in the approximation of the derivative. This is investigated in more detail in Chapter 7.

We finally note that using (6.1) we can characterise the cost of calculating the nonlinear residual on grid Ω_{j-1} as

$$W_{j-1} \approx \frac{1}{s} W_j, \tag{6.7}$$

assuming that the amount of work is proportional to the number of nodes. This simply states that per node a fixed number of operations is performed, which is generally the case. Using the information presented so far we obtain a bound for the computational effort per *V-cycle* in the next section. From this, bounds on the other types of multigrid iteration (*e.g.* W-cycle) can be obtained.

6.1.1 Computational Cost per V-Cycle

In this section we present a characterisation of the computational effort required to perform a single linear or nonlinear V-cycle in a Newton-MG or FAS iteration. Let $C^{(j)}_{\text{NMG}}$ be the cost of performing a single Newton iteration on grid Ω_j where the coarse grid operators are re-discretised; let $\bar{C}^{(j)}_{\text{NMG}}$ be the cost for a single Newton iteration where Galerkin coarse grid operators are used; and let $C^{(j)}_{\text{FAS}}$ be the cost of performing a single FAS V-cycle. In the characterisation of the running times we will make the assumption that (excepting the execution time of the linear multigrid iteration) nonlinear operations dominate the running time of an iteration. Therefore, by inspection of Algorithms 4.2 and 4.3 it can be seen that

$$C^{(j)}_{\text{NMG}} \approx C^{(j)}_{\text{RJ}} + \sum_{i=1}^{j-1} C^{(i)}_{\text{J}} + pC^{(j)}_{\text{LMG}} \tag{6.8}$$

$$\bar{C}^{(j)}_{\text{NMG}} \approx C^{(j)}_{\text{RJ}} + C^{(j)}_{\text{G}} + pC^{(j)}_{\text{LMG}} \tag{6.9}$$

$$C^{(j)}_{\text{FAS}} \approx C^{(j)}_{\text{S}}(\nu_1 + \nu_2) + C^{(j)}_{\text{RHS}} + C^{(j-1)}_{\text{FAS}} \tag{6.10}$$

for $C^{(j)}_{\text{RJ}}$ the cost of calculating the nonlinear residual and Jacobian; $C^{(j)}_{\text{J}}$ the cost of calculating the Jacobian; p the number of linear V-cycles to perform per Newton iteration; $C^{(j)}_{\text{LMG}}$ the cost of performing a linear V-cycle without the calculation of the coarse grid operators included; $C^{(j)}_{\text{G}}$ the cost of calculating the Galerkin coarse grid operators; $C^{(j)}_{\text{S}}$ the cost of a nonlinear smoothing operation; ν_1 and ν_2 the number of pre- and post-smoothing iterations, respectively; and $C^{(j)}_{\text{RHS}}$ the cost of calculating the perturbed right-hand side for FAS. The assumption that the nonlinear operations dominate the execution time is a fair assumption, as demonstrated by the sharpness of the results in Chapter 7. In fact, the

more complicated an operator becomes the more expensive the calculation of the nonlinear terms becomes in comparison to the linear operations, such as grid transfer operations. Results confirming this behaviour are presented in Chapter 7.

In (6.8) the cost of the Newton iteration is given as the cost of calculating the residual and Jacobian matrix on the current grid level plus the cost of performing the linear multigrid iterations. The cost of calculating the Jacobian matrix on the coarser grid levels is included under the assumption that the Jacobian is re-discretised on each grid level. In the case that a Galerkin coarse grid operator is used we need to take into consideration whether the number of non-zero entries per row in the coarse operators grows. Under the assumption that the number of non-zeros per row does not grow (as is the case for the piecewise linear basis functions), or that the rate of growth is controlled such that the calculation of the coarse grid matrices has linear order running time, then the cost of the calculation of the coarse grid operators is linear. It is typical in an implementation to ensure that the coarse grid matrices do not become too dense, as otherwise multigrid is no longer an optimal iteration. In (6.10) the cost of the FAS iteration is given as the cost of smoothing on the current grid, calculating the perturbed right-hand side for the next coarsest grid, and performing an FAS V-cycle on the next coarsest grid.

Before approximating the running times we require estimates for the computational effort to calculate the Jacobian matrix, perform a nonlinear smooth and to calculate the perturbed right hand side for FAS. The cost of calculating the Jacobian matrix is the same as the cost of calculating the residual and the Jacobian, as the residual is calculated as part of the Jacobian calculation. Hence we get

$$C_{\text{RJ}}^{(j)} = C_{\text{J}}^{(j)} = (N_T + 1)W_j. \tag{6.11}$$

To calculate the cost of the nonlinear smoother we need to know which nonlinear smoother we are using. Assuming that we are performing a point-wise nonlinear smooth, such as nonlinear Jacobi (see [139]) we have to calculate the nonlinear residual in each iteration, as well as the diagonals of the current Jacobian matrix. Using previous discussion the cost of this is

$$C_{\text{S}}^{(j)} = 2W_j. \tag{6.12}$$

This cost is accurate if we are performing a point-wise nonlinear Jacobi iteration. A point-wise Gauss-Seidel iteration requires the recalculation of the operator over the support of a basis function every time that a point-wise value is updated [167]. Hence a full point-wise

nonlinear Gauss-Seidel iteration is considerably more expensive than the Jacobi iteration. If some block smooth is performed we need to calculate at least some off-diagonal entries in the Jacobian matrix. This means that the cost of a block smooth will also be considerably higher than the cost of the point-wise Jacobi smoother. In this investigation we consider the case of a pointwise Jacobi smoother, as this gives the smallest cost per iteration whilst the algorithm remains robust.

Finally the cost of calculating the perturbed right-hand side for the next coarsest grid level is estimated. From Algorithm 4.3 we see that to calculate the perturbed right-hand side we calculate the residual on the current grid level, and apply the nonlinear operator to the restricted approximation on the coarser grid. The cost of applying the nonlinear operator is approximately the cost of calculating the residual on a given grid, *i.e.* the cost is approximated by

$$C_{RHS}^{(j)} = W_j + W_{j-1} = (1 + s^{-1})W_j. \tag{6.13}$$

Using the values in (6.11), (6.12) and (6.13) we are able to estimate the running times (6.8), (6.9) and (6.10). We first consider Newton-MG, leaving the estimation of the computational effort for the linear multigrid iteration and the calculation of the Galerkin coarse grid operators until later. We find that

$$\begin{aligned} C_{\text{NMG}}^{(j)} &= (N_T + 1)W_j + \sum_{i=1}^{j-1}(N_T + 1)W_i + pC_{\text{LMG}}^{(j)} \\ &= (N_T + 1)\sum_{i=1}^{j}\frac{1}{s^{j-i}}W_j + pC_{\text{LMG}}^{(j)} \\ &= (N_T + 1)\sum_{i=0}^{j-1}\frac{1}{s^{i}}W_j + pC_{\text{LMG}}^{(j)} \\ &\leq (N_T + 1)\frac{s}{s-1}W_j + pC_{\text{LMG}}^{(j)} \end{aligned} \tag{6.14}$$

in the case that the operator is re-discretised on each grid level, and

$$\bar{C}_{\text{NMG}}^{(j)} = (N_T + 1)W_j + C_{\text{G}}^{(j)} + pC_{\text{LMG}}^{(j)} \tag{6.15}$$

in the case that the Galerkin operator is used. The Galerkin coarse grid matrix is calculated using sparse matrix multiplication of the representations of the restriction, prolongation and Jacobian operator, utilising the method given in Bank and Douglas [8], for example.

In order to perform a comparison of relative efficiency with the FAS iteration we are interested in the cost per V-cycle iteration. Hence we introduce the scaled variables

$$\tilde{C}^{(j)}_{\text{NMG}} \equiv \frac{C^{(j)}_{\text{NMG}}}{p}, \qquad \hat{C}^{(j)}_{\text{NMG}} \equiv \frac{\bar{C}^{(j)}_{\text{NMG}}}{p} \tag{6.16}$$

to be the cost per linear V-cycle of the Newton-MG algorithm.

Now consider the FAS multigrid iteration, where the cost of an FAS V-cycle is approximated as

$$\begin{aligned} C^{(j)}_{\text{FAS}} &= 2W_j(\nu_1+\nu_2) + (1+s^{-1})W_j + C^{(j-1)}_{\text{FAS}} \\ &= \left(2(\nu_1+\nu_2) + 1 + s^{-1}\right) W_j + C^{(j-1)}_{\text{FAS}} \\ &= \left(2(\nu_1+\nu_2) + 1 + s^{-1}\right) \left(W_j + \frac{W_j}{s}\right) + C^{(j-2)}_{\text{FAS}} \\ &= \cdots \\ &= \left(2(\nu_1+\nu_2) + 1 + s^{-1}\right) \left(\sum_{i=1}^{j-1} \frac{W_j}{s^{(i-1)}}\right) + C^{(1)}_{\text{FAS}} \\ &\leq \left(2(\nu_1+\nu_2) + 1 + s^{-1}\right) \frac{s}{s-1} W_j, \end{aligned} \tag{6.17}$$

where, in the last step, we have assumed that the time taken to solve on the coarsest grid level $C^{(1)}_{\text{FAS}}$ is negligible compared to W_j. This is usually the case, but can lead to a potential conflict since FAS requires that the solution of the nonlinear equation is well approximated on the coarsest grid. If this is not the case then higher dimensional coarse spaces are required and the cost of the coarsest grid solve may start to have an adverse effect on the overall execution time.

Before a quantitative estimate can be gained from approximations (6.14) and (6.15) the costs of the linear multigrid iteration and calculation of the Galerkin operators need to be approximated in terms of a work unit. An accurate estimation of these values depends on the dimension of the problem as well as the complexity of the nonlinear problem to be solved. In Section 6.2 an upper bound for the two-dimensional case is presented (see Equation (6.24)), after which the theoretical estimates are compared. Results indicate that the upper bound is sharp for simple model problems and is pessimistic in the case of more complex problems. Hence, the theory introduced here describes the *worst case* running times for the Newton-MG iteration. We also note that we have restricted the smoothing iteration in the FAS iteration to be a pointwise Jacobi iteration. It may be necessary to use a different smoother in practice, in which case the execution of the algorithm will take

considerably longer. Hence the results presented here are for the *best case* running time for an FAS iteration. In the next section we introduce the extension of the estimate of the running times to the case that a non-V-Cycle iteration is used, before gaining some quantitative estimates of execution times for a more specific problem setting.

6.1.2 Estimates for W-Cycles and Higher

In the previous section a bound was gained for the case that a V-cycle iteration was performed as part of a Newton-MG or FAS iteration. These estimates can be used directly to approximate the running times of other types of multigrid cycles. Consider first the FAS iteration. Performing γ multigrid iterations on each coarse grid level instead of the recursion (6.10) we have

$$\begin{aligned} C^{(j)}_{\text{FAS},\gamma} &= \left(2(\nu_1+\nu_2)+1+s^{-1}\right)\left(\sum_{i=1}^{j-1}\left(\frac{\gamma}{s}\right)^{(i-1)}\right)W_j + C^{(1)}_{\text{FAS}} \\ &\leq \left(2(\nu_1+\nu_2)+1+s^{-1}\right)\frac{s}{s-\gamma}W_j. \end{aligned} \tag{6.18}$$

Here $\gamma = 1$ gives a V-cycle, $\gamma = 2$ a W-cycle, and so on. The assumption made in (6.18) is that $\gamma < s$. In this case the running time of the iteration is linear, which is the case in which we are interested. If regular refinement (see Subsection 4.1.1) is used in a d-dimensional setting we have that $s = 2^d$. Hence in two dimensions we are restricted to a cycle where $\gamma < 4$, which is typical in implementations anyway. By definition $C^{(j)}_{\text{FAS}} = C^{(j)}_{\text{FAS},1}$. It is possible to use the estimate $C^{(j)}_{\text{FAS}}$ to obtain estimates for $C^{(j)}_{\text{FAS},\gamma}$ by noting that

$$\frac{C^{(j)}_{\text{FAS},\gamma}}{C^{(j)}_{\text{FAS}}} \approx \frac{s-1}{s-\gamma} \equiv r_\gamma. \tag{6.19}$$

The same ratio of running times holds in the linear case as well. Letting $C^{(j)}_{\text{LMG},\gamma}$ denote the cost of performing a γ-cycle linear multigrid iteration, and letting $C^{(j)}_{\text{LMG}} = C^{(j)}_{\text{LMG},1}$ the ratio

$$\frac{C^{(j)}_{\text{LMG},\gamma}}{C^{(j)}_{\text{LMG}}} \approx \frac{s-1}{s-\gamma} = r_\gamma \tag{6.20}$$

holds. This gives the more general estimates

$$C^{(j)}_{\text{NMG},\gamma} \leq (N_T+1)\frac{s}{s-1}W_j + pr_\gamma C^{(j)}_{\text{LMG}} \tag{6.21}$$

$$\bar{C}^{(j)}_{\text{NMG},\gamma} \approx (N_T+1)W_j + C^{(j)}_{\text{G}} + pr_\gamma C^{(j)}_{\text{LMG}} \tag{6.22}$$

and

$$C^{(j)}_{\text{FAS},\gamma} \approx r_\gamma C^{(j)}_{\text{FAS}}, \tag{6.23}$$

where the inequality in (6.21) and (6.18) can be replaced by an approximately equal to under the assumption that there are sufficiently many levels in the multigrid hierarchy and $\gamma < s$. In the next section we consider these bounds for a piecewise linear nodal finite element basis in two dimensions, and discuss the values obtained. A brief discussion is also given with regards to the use of other bases.

6.2 Comparison of Theoretical Bounds

In this section the cost of a single nonlinear γ-cycle is compared to the cost of a linear γ-cycle for a general *two-dimensional* problem discretised using a *piecewise linear nodal* finite element basis. As we have assumed a simplicial grid, the number of unknowns per element is given by the number of vertices, *i.e.* $d+1 = 3$ in two dimensions. The integer γ is as introduced in the previous section. Before a quantitative estimate for a Newton-MG iteration can be obtained we require an estimate of the computational costs of a linear multigrid V-cycle and the calculation of Galerkin coarse grid operators in terms of a work unit. From empirical experiment we have found that an upper limit for a single V-cycle and the amount of time required to calculate the Galerkin coarse grid operators is given by

$$C^{(j)}_{\text{LMG}} \leq \frac{3}{2}W_j, \qquad C^{(j)}_{\text{G}} \leq W_j. \tag{6.24}$$

These upper bounds are very good estimates for a simple model problem (see discussion of the p-Laplacian in Chapter 7), and are more pessimistic the more complex the nonlinear operator becomes. This is because the cost of the linear operations remains constant as the complexity of the nonlinear operator increases. Hence, the more complicated the

problem the less time a linear V-cycle takes compared to the cost of calculating the nonlinear residual and the results presented here are for a worst-case scenario for Newton-MG. These estimates will also be pessimistic if a finite element basis with an increased number of unknowns per simplex is used, as the time taken to calculate the nonlinear residual will increase more than the time taken for the multigrid iteration on a more dense matrix. The estimates for the FAS iteration are more likely to be sharp for more complex problems, as all calculations counted are related with the nonlinear operator. However, in an implementation the bounds for the FAS iteration are also likely to become pessimistic as the complexity of the nonlinear operator increases. The reasons for this are discussed in Chapters 7 and 8 alongside the results supporting these claims.

In the following we consider the case that equal numbers of pre- and post-smoothing iterations are performed. This is common in practice and reduces the size of the parameter space to consider. The number of pre- and post-smoothing iterations per linear multigrid iteration is kept constant at three iterations throughout. This number may be reduced, but our empirical evidence suggests that this number strikes a good balance between stability and efficiency of the linear multigrid iteration. To simplify the notation in the nonlinear case let $\nu_1 = \nu_2 = \nu$ in the FAS iteration. In a two-dimensional setting the running times of the FAS and Newton-MG cycles can then be approximated as

$$\begin{aligned}\tilde{C}^{(j)}_{\text{NMG},\gamma} = \frac{1}{p} C_{\text{NMG},\gamma} &\approx \frac{1}{p}(N_T+1)\frac{s}{s-1} W_j + r_\gamma C^{(j)}_{\text{LMG}} \\ &\leq \left(\frac{1}{p}(3+1)\frac{4}{3} + r_\gamma \frac{3}{2}\right) W_j \\ &= \left(\frac{16}{3p} + \frac{3}{2} r_\gamma\right) W_j,\end{aligned} \tag{6.25}$$

when the Jacobian is re-discretised on each grid in a Newton-MG iteration;

$$\begin{aligned}\hat{C}^{(j)}_{\text{NMG},\gamma} = \frac{1}{p} \bar{C}_{\text{NMG},\gamma} &\approx \frac{1}{p}(N_T+1) W_j + \frac{1}{p} C^{(j)}_{\text{G}} + r_\gamma C^{(j)}_{\text{LMG}} \\ &\leq \left(\frac{1}{p}(3+1) + \frac{1}{p} + r_\gamma \frac{3}{2}\right) W_j \\ &= \left(\frac{5}{p} + r_\gamma \frac{3}{2}\right) W_j,\end{aligned} \tag{6.26}$$

when Galerkin coarse grid operators are used in a Newton-MG iteration; and

$$\begin{aligned} C^{(j)}_{\text{FAS},\gamma} = r_\gamma C^{(j)}_{\text{FAS}} &\leq r_\gamma \left(2(\nu_1+\nu_2)+1+s^{-1}\right)\frac{s}{s-1}W_j \\ &= \left(4\nu+\frac{5}{4}\right)\frac{4}{3}r_\gamma W_j, \end{aligned} \tag{6.27}$$

for an FAS iteration. Tables 6.1a to 6.1c present these approximations for varying γ, varying smoothing and varying linear multigrid iterations. These approximations show that

γ	$\mathbf{p=1}$	$\mathbf{p=2}$	$\mathbf{p=3}$	$\mathbf{p=4}$
1	$\frac{246}{36}$	$\frac{150}{36}$	$\frac{118}{36}$	$\frac{102}{36}$
2	$\frac{273}{36}$	$\frac{177}{36}$	$\frac{145}{36}$	$\frac{129}{36}$
3	$\frac{354}{36}$	$\frac{258}{36}$	$\frac{226}{36}$	$\frac{210}{36}$

(a) Estimated running times $\tilde{C}^{(j)}_{\text{NMG},\gamma}$ in multiples of work units W_j

γ	$\mathbf{p=1}$	$\mathbf{p=2}$	$\mathbf{p=3}$	$\mathbf{p=4}$
1	$\frac{234}{36}$	$\frac{144}{36}$	$\frac{114}{36}$	$\frac{99}{36}$
2	$\frac{261}{36}$	$\frac{171}{36}$	$\frac{141}{36}$	$\frac{126}{36}$
3	$\frac{342}{36}$	$\frac{252}{36}$	$\frac{222}{36}$	$\frac{207}{36}$

(b) Estimated running times $\hat{C}^{(j)}_{\text{NMG},\gamma}$ in multiples of work units W_j

γ	$\boldsymbol{\nu=1}$	$\boldsymbol{\nu=2}$	$\boldsymbol{\nu=3}$	$\boldsymbol{\nu=4}$
1	$\frac{252}{36}$	$\frac{444}{36}$	$\frac{636}{36}$	$\frac{828}{36}$
2	$\frac{378}{36}$	$\frac{666}{36}$	$\frac{954}{36}$	$\frac{1242}{36}$
3	$\frac{756}{36}$	$\frac{1332}{36}$	$\frac{1908}{36}$	$\frac{2484}{36}$

(c) Estimated running times $C^{(j)}_{\text{FAS},\gamma}$ in multiples of work units W_j

Table 6.1: Estimated running times for nonlinear multigrid iterations.

the most expensive cost per V-cycle for a Newton-MG iteration is almost equal to the least expensive FAS iteration, when only a single non-linear pre- and post-smoothing iteration is performed. Empirical experiments suggest that performing more than a single pre- and post-smoothing iteration gives a more efficient solution algorithm as the number of cycles performed overall is reduced, even though the cost per FAS cycle is increased. The robustness of the method also increases with increasing numbers of pre- and post-smoothing iterations. We have found that performing three pre- and post-smoothing iterations is often a close to optimal number in order to strike a balance between efficiency and robustness of an FAS iteration (in the case of a V-cycle). Although for each problem it is possible to find an optimal value the comparison here will not be performed for an optimal FAS and Newton-MG iteration, as we consider the iterations applied to time-dependent problems.

In this case optimal parameters may change at every time-step and trying to determine optimal parameters dynamically would take extra computational considerations, which may be ignored if a generally close to optimal method is used. We have found that a balance between robustness and efficiency of the method occurs for FAS when $\gamma = 1$ and $\nu = 3$, and for Newton-MG when $\gamma = 1$ and $p = 3$. If we compare the running time per V-cycle at these values it is predicted that a single V-cycle as part of a Newton-MG iteration will be a factor of

$$\frac{C^{(j)}_{\text{FAS}}}{\tilde{C}^{(j)}_{\text{NMG}}} = \frac{318}{59} \approx 5.4 \tag{6.28}$$

faster to execute. This does not necessarily mean that the overall FAS algorithm will be slower, but as long as less than 5.4 times as many linear V-cycles are used than nonlinear V-cycles the Newton-MG will be the faster iteration. Results in Chapters 7 and 8 demonstrate that Newton-MG is always likely to be the more computationally efficient, often by a large margin.

Inspection of Tables 6.1a and 6.1b show that the use of the Galerkin coarse grid operator gives a slight improvement in terms of the running time. Furthermore, as the process of calculating this operator is independent of the nonlinear operator, the upper bound given here on the cost of calculation of the coarse grid operators will get more pessimistic as the nonlinear operator gets more complex. Hence for complex problems the Galerkin coarse grid operator will be preferable, which is clearly demonstrated in the results in Chapter 8. The approximations in Tables 6.1a and 6.1b also highlight the advantages of performing increased numbers of linear iterations as part of the nonlinear Newton iteration. This is well known, and heuristics for deciding on the number of multigrid iterations to perform per Newton iteration are given in [177, §5.3.3]. Simply speaking the closer an approximation to the exact solution, the more effort should be invested in solving the linear system of equations exactly. This allows the quadratic convergence of the Newton method to be recovered. An acceptable alternative is to use a constant number of linear iterations per Newton step, in which case only a linear convergence rate per Newton iteration can be expected. In this thesis the computational efficiency of dynamically choosing the number of linear iterations to perform is not considered. This is because the efficiency of the resulting algorithm will depend on the criteria used to determine the number of V-cycles to perform, as well as an initial estimate and problem dependent parameters. The discussion of this would distract from the comparison of the different methods, although it is noted that a Newton-MG algorithm may be made more efficient if appropriate criteria for

determining the number of linear iterations are chosen.

An enhancement to the Newton-MG iteration is considered in this thesis by using a multigrid preconditioned GMRES as the inner iteration for the Newton method. The execution time of this iteration will necessarily be longer than for the multigrid iteration alone. However the extra work required is a matrix-vector product and a small number of vector inner products (this can be read directly from Algorithm 3.2). In this section we have assumed that linear operations of this form are of negligible cost for an algorithm, and so the estimate of the running time per multigrid preconditioned GMRES iteration can be approximated as the same as the non-preconditioned version. It may even be the case that a multigrid preconditioned GMRES iteration can be made more efficient than just the multigrid iteration, as when multigrid is applied as a preconditioner it does not need to solve the system of linear equations. As such computational expense may be spared by performing very few smoothing iterations as part of the multigrid iteration. Results in Chapter 8 support this claim.

Before concluding the chapter we give a discussion of the effect on the execution time of using different bases and higher dimensions for each iteration, as well as a brief note on the extension and implication of the framework for different discretisation schemes.

6.3 Higher Dimensions and Different Bases

This subsection gives a brief discussion about what can be said regarding the execution times of the FAS and Newton-MG iterations applied to finite element problems discretised using higher than linear order bases and higher spatial dimensions. Only the V-cycle is considered because the theory introduced in the previous section can be used to obtain estimates for other cycles. We start with a discussion of the performance of multigrid in higher dimensions. Let $d = 3$, and assume that the estimates (6.24) hold. These are likely to be pessimistic for the three-dimensional case but we can be confident that these bounds will hold since they hold in the less computationally expensive two-dimensional case. This is under the assumption that the nonlinear operations are more expensive (per unknown) than the linear operations. Therefore, the increase in the number of unknowns going to three dimensions will cause a larger increase in the time required to perform the nonlinear operations than the linear ones.

The current discussion aims to give only a qualitative insight into the running time of the algorithms. We again consider the case of a piecewise linear nodal finite element basis and regular refinement. Therefore we have that $s = 2^d = 8$ and $N_T = d + 1 = 4$. Using these values in estimates (6.16) and (6.17) the estimates of the execution time per V-cycle

are given by

$$\begin{aligned}\tilde{C}^{(j)}_{\text{NMG}} &= \frac{1}{p}C^{(j)}_{\text{NMG}} \approx \frac{1}{p}(N_T+1)\frac{s}{s-1}W_j + C^{(j)}_{\text{LMG}} \\ &\leq \left(\frac{1}{p}(4+1)\frac{8}{7}+\frac{3}{2}\right)W_j \\ &= \left(\frac{40}{7p}+\frac{3}{2}\right)W_j\end{aligned} \tag{6.29}$$

for the Newton iteration when the coarse operators are re-discretised and

$$\begin{aligned}C^{(j)}_{\text{FAS}} &\leq \left(2(\nu_1+\nu_2)+1+s^{-1}\right)\frac{s}{s-1}W_j \\ &= \left(4\nu+\frac{9}{8}\right)\frac{8}{7}W_j\end{aligned} \tag{6.30}$$

for the FAS iteration. Now comparing the execution times when 3 pre- and post-smoothing iterations are performed for the FAS iteration and 3 linear multigrid iterations are performed as part of the Newton iteration the ratio is approximated as

$$\frac{C^{(j)}_{\text{FAS}}}{\tilde{C}^{(j)}_{\text{NMG}}} = \frac{107}{7}\times\left(\frac{143}{42}\right)^{-1} \approx 4.5 \tag{6.31}$$

which is significantly lower than the two-dimensional estimate (6.28). This helps to highlight that the FAS iteration becomes *less* expensive (relative to the calculation of the residual) as the dimension increases, whereas the Newton iteration becomes *more* expensive. However, the approximation (6.31) is likely to be at least a little pessimistic, and still represents a significant improvement in running time for Newton-MG. Also, the estimate of the FAS running time (6.17) is independent of the number of unknowns on an element only because we restrict the smoother to a nonlinear Jacobi iteration. As previously mentioned, this may not be a favourable smoother to use, and using a different smoother will have a significant negative impact on the running time. Hence the estimate (6.31) is for a *worst case* scenario, if we are considering the estimate from the point of view of a Newton iteration.

The discussion of the execution time for higher order elements follows similarly to the discussion above, as a higher order element introduces a larger number of unknowns per element. For increasing order of approximation the relative execution time of the FAS iteration remains constant (for a fixed dimension), while the relative execution time of a

Newton-MG iteration is increased. Theoretically, therefore, for a sufficiently high accuracy approximation in three dimensions the FAS iteration may outperform a Newton-MG iteration on a per V-cycle basis, but this is unlikely to occur in a practical implementation as at least a fourth order basis should be used for this to be the case.

6.4 Application to Other Discretisation Methods

The framework introduced in this chapter may be used to evaluate the execution time for discretisation methods other than finite elements. All that is required is an estimate of the cost of the different parts of the algorithms related to the cost of calculating the nonlinear residual, which is denoted by W_j. As an illustrative example a finite difference discretisation is considered, although the extension to a finite volume discretisation is performed similarly.

To evaluate the cost of the nonlinear iterations it is assumed that a difference stencil is applied, and, for simplicity, boundary effects are ignored. The number of points used in the stencil is denoted by N_{S}. The case of re-discretising the coarse grid operators as part of the inner iteration for Newton-MG is now compared to the FAS iteration.

Recall that the running times of Newton-MG and FAS are approximated in (6.8) and (6.10) as

$$\tilde{C}^{(j)}_{\mathrm{NMG}} \approx \frac{1}{p}\left(C^{(j)}_{\mathrm{RJ}} + \sum_{i=1}^{j-1} C^{(j)}_{\mathrm{J}}\right) + C^{(j)}_{\mathrm{LMG}}$$

and

$$C^{(j)}_{\mathrm{FAS}} \approx C^{(j)}_{\mathrm{S}}(\nu_1+\nu_2) + C^{(j)}_{\mathrm{RHS}} + C^{(j-1)}_{\mathrm{FAS}}.$$

The assumption is again made that the derivative is approximated using a divided difference. The cost of performing an FAS iteration for a finite difference approximation is the same, relative to the residual calculation, as the cost for a finite element discretisation, *i.e.*

$$C^{(j)}_{\mathrm{FAS}} \approx \left(2(\nu_1+\nu_2) + 1 + s^{-1}\right)\frac{s}{s-1}W_j. \tag{6.32}$$

The estimate for Newton-MG iteration for a finite difference discretisation varies slightly from that for a finite element discretisation. In order to calculate the entries in

the Jacobian matrix the derivative is approximated at every point in the difference stencil, meaning that $(N_{\mathrm{S}}+1)$ local function evaluations are required at each grid node in order to calculate the Jacobian matrix. Hence

$$C_{\mathrm{RJ}}^{(j)} = C_{\mathrm{J}}^{(j)} \approx (N_{\mathrm{S}}+1)W_j,$$

The cost of re-discretising the coarse grid operators has the same cost relative to the calculation of the residual as in the previous sections, but now the relative cost of the linear multigrid iteration $C_{\mathrm{NMG}}^{(j)}$ will be considerably higher. This is because the calculation of the residual does not require contributions to an integral to be calculated over each element, and only a pointwise calculation needs to be performed. Assume that it takes a factor μ times longer to calculate the residual in a finite element discretisation than it does in a finite difference discretisation. We are not interested in an exact value in this thesis, and simply highlight how the FAS iteration becomes more competitive in terms of computational efficiency in the case of a finite difference discretisation. Then we have the estimate (see (6.24))

$$C_{\mathrm{LMG}}^{(j)} \leq \frac{3\mu}{2}W_j$$

so that overall

$$C_{\mathrm{NMG}}^{(j)} \approx \left(\frac{1}{p}(N_{\mathrm{S}}+1)\frac{s}{s-1} + \frac{3\mu}{2}\right)W_j. \tag{6.33}$$

In two dimensions consider a regular refinement of the grid such that $s=4$, and consider a 5-point stencil such that for 3 pre- and post-smooths the computational cost of an FAS iteration is

$$C_{\mathrm{FAS}}^{(j)} \approx \left(2\times 6 + 1 + \frac{1}{4}\right)\frac{4}{3}W_j = \frac{53}{3}W_j, \tag{6.34}$$

and for 3 linear multigrid iterations per Newton step the computational cost of the Newton-MG iteration per linear V-cycle is

$$\tilde{C}_{\mathrm{NMG}}^{(j)} \approx \left(\frac{1}{3}(5+1)\frac{4}{3} + \frac{3\mu}{2}\right)W_j = \frac{16+9\mu}{6}W_j. \tag{6.35}$$

Using this approximation suggests that if $\mu > 10$, the computational cost per V-cycle would be higher for the Newton-MG than for FAS. Whilst it is unlikely that the cost of calculating the residual for a finite element discretisation will be more than 10 times the cost for a finite difference discretisation, we do expect to see a large increase in the time. For example, if the residual calculation were to cost 5 times more in the finite element setting (*i.e.* $\mu = 5$) we have that a single V-cycle would take only twice as long for FAS as for Newton-MG, which is a large improvement over the factor 5.4 estimated in Section 6.2.

The advantage of using FAS becomes even more apparent if we consider the three dimensional case. In three dimensions the standard 7-point stencil is often insufficient to give good convergence, and an improvement is observed if a 19-point stencil is used. In this case we have that $N_{\mathrm{S}} = 19$. Therefore the execution time of an FAS iteration will be approximated as

$$C_{\mathrm{FAS}}^{(j)} \approx \left(2 \times 6 + 1 + \frac{1}{8}\right) \frac{8}{7} W_j = \frac{105}{7} W_j, \tag{6.36}$$

and the execution time of a Newton-MG iteration as

$$C_{\mathrm{NMG}}^{(j)} \approx \left(\frac{1}{3}(19+1)\frac{8}{7} + \frac{3\mu}{2}\right) W_j = \frac{320 + 63\mu}{42} W_j. \tag{6.37}$$

From these estimates we see that if $\mu \approx 5$ or greater that the cost of an FAS V-cycle will actually be less than a V-cycle as part of a Newton iteration. In this case, if similar numbers of V-cycles are performed the FAS iteration will be the more computationally efficient.

From (6.32) it can be seen that the execution time of FAS, relative to W_j, is independent of the number of non-zero entries in the stencil, whereas (6.33) shows that a Newton-MG iteration becomes relatively more expensive the more non-zero entries there are in the stencil. The number of entries in the stencil increases when a higher accuracy is sought from the solution, and also by going to higher dimensions. Therefore, in a three-dimensional setting FAS is much more competitive in terms of execution time compared to the Newton-MG iteration for a finite difference discretisation than for a finite element discretisation.

Although it is more likely that an FAS iteration will be more competitive in terms of execution time per V-cycle in the finite difference setting, there are many shortcomings of using a finite difference discretisation – as discussed in Section 2.1 – especially the fact

that a regular grid is required for an efficient implementation. However, if a finite difference discretisation is applicable then the FAS is likely to be a competitive algorithm. A more detailed investigation of nonlinear multigrid methods for finite difference discretisation methods would be required before a conclusions could be drawn as to which is the most suitable, as convergence behaviours of the methods for realistic problems need to be taken into account in a thorough comparison. The discussion for the rest of the thesis returns to a finite element discretisation, in which it was shown in Section 6.2 that Newton-MG is likely to be a more efficient algorithm. A thorough investigation of the convergence and algorithmic properties of the algorithms for a finite element discretisation demonstrates the superiority of a Newton-MG iteration.

6.5 Summary

In this chapter we have developed and presented a framework for estimating the computational effort involved in a single multigrid cycle for both Newton-MG and FAS. In the next chapters we present results mainly for the V-cycle. Results for simple model problems are presented for the W-cycle and higher, but these are presented to demonstrate that, qualitatively, the running time of the methods can be determined by considering only the V-cycle iteration. Hence the investigation concentrates primarily on the V-cycle. In Chapter 7 a stationary model problem, known as the p-Laplacian is introduced, and the application of Newton-MG and FAS to this problem is discussed. This problem is chosen as it is simpler to highlight different convergence behaviours of the methods, which may be more difficult to isolate, or are less apparent, in more complex applied problems. Chapter 8 considers an application of the methods to time-dependent problems which are used to model some complex physical phenomena. The results in Chapter 8 demonstrate the advantages of Newton-MG over FAS as a flexible and robust method for solving practical PDEs.

Chapter 7

Application to the p-Laplacian

In this chapter, and the next, results are presented backing up the theory introduced in Chapter 6 for two-dimensional model problems discretised using a piecewise linear nodal finite element basis. The grids on which the problems are discretised are regular simplicial grids and a regular refinement strategy (see §4.1.1) is employed. For the study at hand, where we are interested in the computational effort in performing a V-cycle iteration, the extra complexity of solving on a domain with complex geometry, or performing some adaptive mesh refinement or other adaptive meshing would only serve to obscure the results, and so are not considered here. Some more technical results are discussed, especially the convergence of the nonlinear iterations in the case that a coefficient function is highly varying on the domain, for which the regular grids are adequate to highlight the behaviour of the methods.

The model problem considered in this chapter – the p-Laplacian – is a stationary scalar equation, and is a standard nonlinear model problem. The p-Laplacian, which is a generalisation of the well-known linear Laplacian problem, is introduced in Section 7.1. This is followed by a detailed investigation of how Newton-MG and FAS perform for this method. Results are presented demonstrating the sharpness of the bounds gained on the running time of the algorithms, as outlined in Chapter 6. We also present results demonstrating the qualitative behaviour of each of the algorithms and draw parallels with the well-known linear iterations, as the lack of theory prevents any definitive mathematical statements being made regarding the nonlinear iterations.

The discussion in this chapter serves to show that the Newton-MG iteration is superior

to FAS in almost every respect for the p-Laplacian, and highlights implementation issues required to gain an efficient realisation of the Newton-MG iteration. This behaviour is verified for applications to time-dependent problems, which are also more complex than the p-Laplacian, in Chapter 8. It is also demonstrated that the qualitative behaviour observed for the p-Laplacian holds in the case of more complex nonlinear operators too, and heuristics are given, linked to linear multigrid methods, that describe the qualitative convergence behaviour of nonlinear multigrid methods.

7.1 The p-Laplacian

In this section the problem statement, as well as the weak formulation, is given for the p-Laplacian. The domain on which the equation is to be solved is the unit square

$$\Omega \equiv (0,1)^2 \tag{7.1}$$

and the solution is assumed to exist and be unique in the space $\mathcal{V}$. For this chapter the basis functions are given by the piecewise linear nodal finite element basis functions of space S_0 (see (2.14)) and we assume that $S_0 \subset \mathcal{V}$. A hierarchy of grids

$$\Omega_1 \subset \Omega_2 \ldots \subset \Omega_J \subset \Omega \tag{7.2}$$

is given. With each Ω_j, $j = 1, \ldots, J$ we associate a node set $\mathcal{N}_j$; regular triangulation $\mathcal{T}_j$; characteristic grid spacing h_j (see (2.1)); and finite element space $\mathcal{V}_j \subset S_0(\Omega_j)$. Since a regular refinement is used $h_j = 2h_{j+1}$. Again, $N_j = \#\mathcal{N}_j$ is the dimension of $\mathcal{V}_j$.

The p-Laplacian is the generalisation of the well-known linear Laplacian operator equation and it is given by

$$\begin{gathered} -\nabla \cdot \left(|\nabla u|^{p-2} \nabla u \right) - f = 0, \quad \vec{x} \in \Omega, \\ u = 0, \quad \vec{x} \in \partial\Omega \end{gathered} \tag{7.3}$$

where

$$|\nabla u|^2 = \sum_{i=1}^{d} \left(\frac{\partial}{\partial x_i} u \right)^2 . \tag{7.4}$$

For $p \neq 2$ this equation is one of the simplest examples of a nonlinear second order

operator, and is of the form

$$-\nabla \cdot (g(\nabla u)\nabla u) = f \tag{7.5}$$

with $g \equiv |\nabla u|^{p-2}$ in this case. Note that $p = 2$ reduces to a linear Laplace's equation. The form given in (7.5) is of particular interest because, for g strictly positive, the linearisation is a symmetric positive definite bilinear form when there exists a set $\Gamma_{\mathrm{D}} \subseteq \partial\Omega$ such that $u|_{\Gamma_{\mathrm{D}}} = 0$. This means that the weak form

$$\int_{\Omega} g(\nabla u)\nabla u \nabla v \,\mathrm{d}\vec{x} = \int_{\Omega} fv \,\mathrm{d}\vec{x} + \int_{\partial\Omega\setminus\Gamma_{\mathrm{D}}} g(\nabla u)\nabla u \cdot \nu v \,\mathrm{d}\vec{x} \tag{7.6}$$

corresponds to an Euler-Lagrange equation on Ω, and solving it is therefore equivalent to minimising a nonlinear functional. In the case of (7.3) with homogeneous Dirichlet boundary the functional to be minimised is given by

$$\mathcal{J}_v = \frac{1}{p}\int_{\Omega} |\nabla v|^p \,\mathrm{d}\vec{x} - \int_{\Omega} fv \,\mathrm{d}\vec{x}. \tag{7.7}$$

Existence and uniqueness are well known for $p \in (1, \infty)$ in the continuous case [12], and also for a piecewise linear nodal finite element discretisation (see [52, Theorem 5.3.3], [12] and, in a much more general setting, [61]) in the case that $u \in W_0^{1,p}(\Omega)$ and $f \in L^{p'}(\Omega)$ with $\frac{1}{p} + \frac{1}{p'} = 1$. In the case that $p \neq 2$ (*i.e.* the nonlinear case) the operator is *degenerate* [12], and care must be taken in how a solution algorithm is implemented, especially in the case $p < 2$. In this thesis data has been chosen such that a convergent iteration is gained with a basic implementation of a Newton-MG and FAS iteration. For this we let $f \in L^{\infty}(\Omega)$, $p = 4$ and, since Ω is the unit square, the domain is convex with Lipschitz boundary. In order to highlight some of the characteristics of the nonlinear multigrid methods under investigation we consider the following 4-Laplacian:

$$-\nabla \cdot \left(\alpha(\vec{x}) \left|\nabla u\right|^2 \nabla u\right) - f = 0, \tag{7.8}$$

with homogeneous Dirichlet boundary conditions and $\alpha > 0$ a piecewise constant function on Ω. In the next section we give the weak form and discretisation of the problem, and discuss how a Newton-MG and an FAS method can be expected to perform as iterative solvers for the 4-Laplacian.

7.2 Weak Formulation and Discretisation

The weak form of (7.8) is given by

$$\begin{gathered} F(u)(v) \equiv \int_\Omega \alpha \left|\nabla u\right|^2 \nabla u \nabla v \,\mathrm{d}\vec{x} - \int_\Omega f v \,\mathrm{d}\vec{x} = 0, \quad \vec{x} \in \Omega \\ u = 0, \quad \vec{x} \in \partial\Omega, \end{gathered} \tag{7.9}$$

for all $v \in \mathcal{V} = W_0^{1,4}(\Omega)$. It is known that the solution may have limited regularity even for infinitely smooth boundary data or right-hand-side data and for convex polygonal Ω [12]. Hence using a higher order basis than linear may not be advantageous. Therefore we assume, in this case, that the most suitable discretisation is a low order one such as S_0 (see (2.14)). The Fréchet derivative of (7.9) is given by

$$F_u(v, w) = \int_\Omega \alpha \left|\nabla u\right|^2 \nabla w \nabla v \,\mathrm{d}\vec{x} + \int_\Omega 2\alpha (\nabla u \nabla w)(\nabla u \nabla v) \,\mathrm{d}\vec{x}. \tag{7.10}$$

Inspection of (7.10) shows that the Fréchet derivative is a symmetric positive definite bilinear form (using homogeneous Dirichlet boundary conditions). The application of the operator and Jacobian matrix is calculated on an element-by-element basis, as outlined in Equations (6.3) and (6.6). The discrete system of nonlinear equations to solve on grid Ω_j is given by

$$F(u_j)(\varphi_i) = 0, \quad i = 1, \ldots, N_j \tag{7.11}$$

where the $\{\varphi_i\}_{i=1}^{N_j}$ forms a basis of the space $\mathcal{V}_j = S_0(\Omega_j) \subset W_0^{1,4}(\Omega)$, as outlined at the start of this chapter. Therefore an approximation $u_j \in \mathcal{V}_j$ is represented as

$$u_j = \sum_{i=1}^{N_j} u_i \varphi_i. \tag{7.12}$$

Let the Newton iteration be defined as

$$u_j^{(k+1)} = u_j^{(k)} + \delta^{(k)}$$

for $u_j^{(k)}, \delta^{(k)} \in \mathcal{V}_j$, $k = 0 \ldots$. Then the discrete system of linear equations to solve at each Newton iteration is given by

$$F_{u_j^{(k)}}(\varphi_i, \delta^{(k)}) = -F(u)(\varphi_i), \quad i = 1, \ldots, N_j, \tag{7.13}$$

where $\delta^{(k)}$ is also represented in terms of the basis $\{\varphi_i\}_{i=1}^{N_j}$ (see (7.12)).

As discussed in Chapter 5 it is not clear how to analyse the convergence of a Newton-MG or FAS iteration for this problem. However, we can treat (7.13) as an independent linear problem and use existing theory to approximate how a linear multigrid iteration will perform. As the bilinear form is dependent on the nonlinear solution u it is not possible to say, in general, the properties that the linear operator possesses without at least making some assumptions on u. For the purposes of this chapter the following assumptions are made regarding the solution of the problem:

Assumption 7.1.a. *For a sequence $(u^{(i)})$, $i = 1, 2, 3, \ldots$ of Newton iterates, each $u^{(i)}$ is Lipschitz continuous. This implies that $u^{(i)} \in W^{1,\infty}$ [71, pg. 279].*

Assumption 7.1.b. *For some small constant C, independent of mesh parameters or coefficient values*

$$\left\|\nabla u^{(i)}\right\|_\infty \leq C \tag{7.14}$$

holds, which ensures that any oscillations over the domain are small, and that there aren't strong anisotropies in the term

$$\int_\Omega 2\alpha(\nabla u \nabla w)(\nabla u \nabla v)\, d\vec{x}. \tag{7.15}$$

in (7.10)*.*

Assumption 7.1.c. *Any discontinuities in $\nabla u^{(i)}$ are aligned with discontinuities in coefficient α.*

Note that discrete approximations $u_j \in \mathcal{V}_j$ satisfy Assumption 7.1.a by construction. Assumption 7.1.b implies that a point-wise smoother is likely to be appropriate for the problem, as there are no strong anisotropies on the domain. Assumptions 7.1 are not so restrictive, and in practice these are often satisfied. A typical example is when α represents a material property for different materials defined on a domain. In this case some jump

in the derivative of the function can be expected, although the function itself is assumed continuous. Under Assumptions 7.1 the convergence behaviour will be dominated by the coefficient function α, as demonstrated in Section 7.7.

In a linear setting the solution u may depend on the diffusion coefficient α, but it is still possible to show when the convergence of a linear multigrid iteration is independent of the coefficient α and the size of the jumps in α [156]. In the nonlinear case the solution will generally depend upon α, but convergence of the method cannot be independent of α unless the solution is independent of α. In this chapter we characterise the cases in which this can be expected to happen. We also show that nonlinear multigrid methods can achieve mesh independent linear convergence and show that the linear theory can be used to predict when divergence of a nonlinear iteration will occur. Comparison of the results demonstrates the superior computational efficiency of an implementation of Newton-MG over FAS, but the theoretical considerations also demonstrate that the theory is much more powerful and applicable in the case of a Newton-MG iteration. Results in the next section demonstrate that the theory introduced in Chapter 6 gives sharp estimates for the running time per V-cycle iteration for the p-Laplacian. After this the robustness and running time of the nonlinear iterations are assessed when the iterations are employed as solution algorithms.

7.3 V-Cycle Execution Time

This section presents results demonstrating the sharpness of the theoretical bounds developed in the previous section. The execution times are predicted in Chapter 6 relative to the computational cost required to calculate the nonlinear residual, and so in this section we compare computational results to verify these estimates. For brevity we restrict ourselves to a 512×512 uniform grid, which is a large enough problem for the asymptotic timings to be observed. The calculation of the residual is timed for 1000 successive iterations, and the average run time of the residual calculation is then taken. Our empirical experiments suggest that

$$W_{512} \approx 0.035 \text{ seconds} \tag{7.16}$$

is a good approximation for the execution time of a work unit for the p-Laplacian. The measured time is the wall-clock time rather than CPU time. We adopt the slightly different notation from the previous chapter, such that W_{512} indicates the amount of work required on a regular 512×512 mesh. Table 7.1 shows the predicted *vs*. actual execution times re-

	p	$\tilde{C}_{\text{NMG}}$	Act.	$\hat{C}_{\text{NMG}}$	Act.	ν	C_{FAS}	Act.
$\gamma = 1$	1	23.92	21.30	22.75	22.11	1	24.5	23.76
	2	29.17	27.69	28.00	27.69	2	43.17	40.21
	3	34.41	33.26	33.25	33.08	3	61.83	57.09
	4	39.67	38.62	38.5	38.43	4	80.5	73.78
$\gamma = 2$	1	26.54	24.68	25.38	25.14	1	36.75	47.27
	2	34.41	30.85	33.25	31.25	2	64.75	69.21
	3	42.29	37.86	41.13	37.74	3	92.75	90.68
	4	50.17	43.12	49.0	43.50	4	120.75	111.34

Table 7.1: Predicted *vs*. actual execution time required to perform one hundred Newton / FAS iterations for the 4-Laplacian. Timings are presented in seconds.

quired to perform one hundred Newton or FAS iterations for varying multigrid cycles and numbers of inner iterations or smoothing iterations. In the table γ represents the number of multigrid iterations performed per grid level such that $\gamma = 1$ gives a V-cycle and $\gamma = 2$ gives a W-cycle; p is the number of inner iterations performed per Newton iteration; and ν is the number of pre- and post-smoothing iterations performed in an FAS iteration. As discussed in the previous chapter the number of pre- and post-smoothing iterations used as part of the linear multigrid iterations is kept constant at three. $\tilde{C}_{\text{NMG}}$ (see (6.16)) is the predicted execution time of a Newton iteration when the coarse grid operators are re-discretised on each grid, and $\hat{C}_{\text{NMG}}$ (see (6.16)) is the predicted execution time of a Newton iteration when the Galerkin coarse grid operators (see (4.20)) are used. C_{FAS} is the predicted execution time when an FAS iteration is used, and all columns labelled 'Act.' are the measured running times. The number of linear multigrid iterations performed is $100p$ as p multigrid iterations are performed in each of 100 Newton iterations.

The results clearly demonstrate that the running times are predicted well when using W_{512} as a work unit in the theoretical bounds introduced in the previous chapter, especially in the case of the Newton-MG iteration. The running times in this case are expected to be predicted well, as it is for this model problem that the estimation of the execution times for the linear multigrid and Galerkin coarse grid operator are taken. Results in Chapter 8 show that these estimates are realistic also when applied to other problems. It should be noted that the execution time of the FAS W-cycles are initially optimistic, which is interesting. If an accurate prediction is obtained for the V-cycle it should also be the case for the W-cycle. The issue here is that computer code (in this case C) is compiled with optimisation flags set. For the case of the W-cycle it does not seem to be very good at optimising the code, so the iteration takes longer than expected. When compiled without

automatic optimisation this behaviour is not observed. Results are presented here for the optimised case, as this is what would be run in practice. The fact that the Newton-MG is more suited to optimisations may be a result of our specific implementation. However, the authors' knowledge of compilers and optimisations is not sufficient to be able to state whether this would be the case for another implementation. The same results are observed for other model problems (see Chapter 8). The increase in the execution time for the W-cycle with increasing numbers of smooths is predicted in a satisfactory manner, though, even if the estimate for $\nu = 1$ is optimistic.

In Table 7.1 a slight advantage in the use of the Galerkin coarse grid operator in terms of the computational cost is observed in the case of the V-cycle ($\gamma = 1$). The advantage is not so clear in the case of cycles with $\gamma > 1$, due to the fact that the amount of computational time required to calculate the coarse grid operators becomes less significant compared to the time required for the solution of the linear system. This effect is seen in the estimates (6.8) and (6.9), as the calculation of the coarse grid operators is independent of the number of linear iterations performed. Qualitatively the computational efficiency of the iterations is clearly captured by the results, and the formulae for $\tilde{C}_{\text{NMG}}$ and $\hat{C}_{\text{NMG}}$, as introduced in Chapter 6, predict the execution time well.

The estimates for the execution time of an FAS iteration are more pessimistic and less sharp, the reasons for which were discussed briefly on the previous page. However, the correct rate of growth when increasing smoothing iterations and γ is captured. Indeed, we see that the ratio between the running time of a V- and W-cycle is appropriately captured. As predicted by the theory, the execution time for a W-cycle should be 1.5 times that of a V-cycle, which is closely mirrored by the ratios when performing 4 pre- and post-smooths. This indicates that the estimates presented can be useful in characterising the execution time, and the order of magnitude with which they are estimated is correct.

From the sharpness of the bounds the estimate of the ratio between the running time of a single linear and nonlinear V-cycle, as given in (6.28), should be accurate. Equation (6.28) gives the approximate ratio between the execution times when 3 pre- and post-smooths are used in an FAS iteration, and 3 linear V-cycles are used as part of a Newton iteration as ~ 5.4. From Table 7.1 the ratio per V-cycle of the actual execution times is given by

$$\frac{57.09}{100} \times \frac{300}{33.26} \approx 5.15, \tag{7.17}$$

which is close to the predicted value, demonstrating that the order of magnitude increase in performance is accurately captured in the framework presented.

Although FAS is more computationally expensive per iteration it may still be a more computationally efficient solver, depending on the rate of convergence per V-cycle iteration. Using the above discussion we require the FAS convergence factor to be five times smaller than that of the Newton-MG per multigrid iteration. Whilst the computational complexity is important, the quality of an iterative method is also judged on robustness. This could be robustness with respect to algorithmic details, or robustness with respect to initial guesses. A presentation of the convergence factors and the robustness of the methods is given in the following sections. We begin in Sections 7.4 and 7.5 with an investigation of the convergence when the exact solution is known. For this we consider that the coefficient function α in (7.9) is constant. This is done so that control on a good and poor initial approximation can be gained. Section 7.4 investigates the convergence properties of the methods when the initial guess is in a ball around the exact solution where super-linear convergence can be expected. This is followed by Section 7.5 in which the robustness of the methods with respect to poor initial estimates is investigated. Section 7.6 compares techniques used to extend so called trust regions, in which the nonlinear iterations are convergent. Section 7.7 considers the more interesting case of a highly varying coefficient α and results are compared to those for linear problems. The computational efficiency of the methods is discussed where this is appropriate. Results are also presented in which FAS appears to perform better than Newton-MG, and a discussion clearly demonstrates that this can be attributed to implementation rather than algorithmic issues.

7.4 Convergence for Good Initial Guesses

In this section we give results demonstrating the robustness of FAS and Newton-MG applied to Equation (7.8). The linearisation of this problem is given in (7.10). As previously noted there is no current theory available to analyse the convergence of a Newton-MG iteration when the solution u^* of the weak form (7.9) and the exact Newton correction δ^*, which solves,

$$F_u(v, \delta) = -F(u)(v), \quad \forall v \in \mathcal{V}, \tag{7.18}$$

are in the space $\mathcal{V} = W^{1,4}$. Instead we consider the convergence of the linear problem (7.18) under the assumption that the exact correction term δ^* is in the space H_0^1. This is a restrictive assumption, but allows for the standard linear multigrid theory (see [196, 202] for a review) to be used to predict how the linear iteration will converge. These predictions are compared to observed convergence behaviours to show that for model problems at the

scales presented it seems as though the assumption on δ^* is a useful one.

In order to be able to choose 'good' and 'bad' initial estimates the right-hand side is set so that the solution u^* to (7.9) is given by

$$u^* = C_1 \sin(\pi x) \sin(\pi y), \tag{7.19}$$

for scalar $C_1 > 0$. In this case the solution u obviously has no dependence on the parameter α, as the right-hand side changes with the coefficient function. As α is modified the nonlinear operator and the Jacobian matrix are scaled by a constant factor, meaning that the convergence behaviour is independent of α. Although this independence of α is manufactured in this example, the right-hand side often depends on some scalar parameters which may be highly varying on the domain. A typical example is when the coefficient represents some material properties on a domain with several different materials.

To begin with we consider that the approximation is in a ball of guaranteed convergence. Empirical evidence suggests that

$$u^{(0)} = 0.95 C_1 \sin(\pi x) \sin(\pi y) \tag{7.20}$$

satisfies this condition. As the convergence is independent of α for the nonlinear problem we set $\alpha = 1.0$ on the domain and consider varying the scalar C_1. In this case, under Assumptions 7.1, the symmetric positive definite bilinear form (7.13) has a continuous, bounded coefficient, and hence the theory of Xu [196] says that convergence of the linear iteration should occur for all values of u, so long as the coarsest grid represents the error in approximation appropriately. From (7.10) we find that the entries in the Jacobian matrix scale as C_1^2 exactly, meaning that a change in the constant C_1 should have no effect on the convergence of either FAS or Newton-MG, as a re-scaling does not change the spectral properties of the operators.

We first consider the results when $C_1 = 1.0$, as given in Table 7.2. The results presented give the number of multigrid V-cycles (nonlinear and linear) for the FAS and Newton-MG iterations required to reduce the original residual in approximation by a factor 1e-7, where the coarsest grid used has 9 unknowns. Mesh independent convergence is characterised by a constant number of V-cycle iterations required for convergence. The results for the FAS iteration display this behaviour, although a significant number of iterations are required to reach convergence. On the other hand, the convergence of Newton-MG seems to deteriorate as the mesh spacing decreases, especially in the case that a numeric approximation to the Jacobian is used. The results presented here are for the case

Grid Size	FAS		Newton-MG	
	Analytic	Numeric	Analytic	Numeric
64^2	22	22	15	15
128^2	26	26	15	15
256^2	28	28	15	15
512^2	29	29	15	15
1024^2	29	29	15	15
2048^2	29	29	18	27

Table 7.2: Number of V-cycles required to reduce the original residual, when $C_1 = 1$, by a factor 1e-7 for FAS and Newton-MG (with Galerkin coarse grid operators) using a coarse grid with 9 unknowns (4^2 grid).

that the Jacobian is approximated using a forward difference formula. The perturbation parameter used is $\epsilon = 4\text{e-}10$, which was found to be the best value through experimentation. A demonstration of the effect of the change in ϵ is shown in Figure 7.6, although this ties in more closely with the discussion later in this chapter. We note, however, that even in the case that the analytic Jacobian is used there seems to be a deterioration in the convergence of the Newton method.

The explanation for the deterioration in the convergence of the Newton method in this case is due to the fact that the coarsest grid is too coarse. Changing the coarsest grid to have 225 unknowns gives the improved results shown in Table 7.3. As can be seen, using

Grid Size	Analytic	Numeric
64^2	12	12
128^2	12	12
256^2	12	12
512^2	12	12
1024^2	12	12
2048^2	12	27

Table 7.3: Number of V-Cycles required to reduce the original residual, when $C_1 = 1$, by a factor 1e-7 for Newton-MG (with Galerkin coarse grid operators) using a coarse grid with 225 unknowns (16^2 grid).

the analytic Jacobian the Newton-MG method displays mesh independent convergence, which is characterised by a constant number of V-cycle iterations in Table 7.3. It is interesting to note that the convergence of the FAS method does not improve when this finer coarsest mesh is used. This suggests that for the FAS iteration and this model problem the 4^2 mesh provides a good enough representation of the solution for the method to display

robust convergence. The same is not true for the Newton iteration, as we are not trying to represent the exact solution on the coarsest grid, but some error in the approximation. This is evidently poorly approximated on the 4^2 mesh, but appropriate data is captured on a 16^2 mesh. This behaviour only becomes apparent when a fine enough resolution is used, as using a 4^2 coarsest mesh does seem to display mesh independent convergence for grids with resolution 1024^2 or less. Using the more appropriate coarse grid we also see an improvement in the number of V-cycles required for convergence on all grid levels, comparing Tables 7.2 and 7.3.

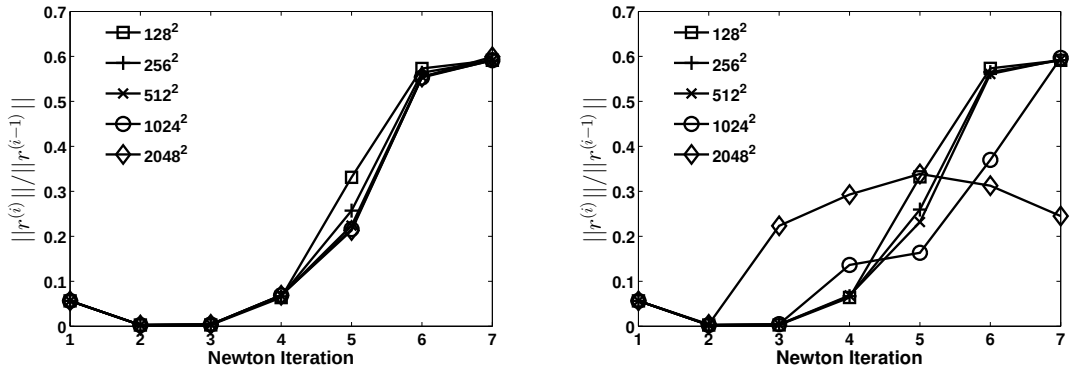
The number of V-cycles performed does not tell us the overall running time of the algorithms. For this the execution time per V-cycle should also be taken into account. The execution time per V-cycle was measured in Section 7.3, where it was demonstrated that the Newton-MG iteration is a much more efficient iteration per V-cycle, so that a large difference in execution times should be observed. The results are shown in Figure 7.8, but the discussion of these is left until an investigation of the reasons for the deterioration in the convergence of the Newton-MG iteration using a numeric approximation to the Jacobian has been completed, which is performed below.

The results presented in Tables 7.2 and 7.3 are for when a forward difference approximation to the entries in the Jacobian matrix is used, as per (6.6). From empirical experiment it was found that for this problem the best results were obtained for a perturbation size of

$$\epsilon = 4\text{e-}10$$

in (6.6). To demonstrate that the convergence does deteriorate when a numeric approximation to the Jacobian is used the convergence histories for the first 7 Newton iterations using the exact and numeric approximation is shown in Figure 7.1. The deterioration in the convergence of the method for the 2048^2 grid is clear to see. What seems to be deterioration in the convergence at finer grid resolutions is actually just the exact solution being approached: after the fourth Newton iteration the norm of the nonlinear residual is $\sim 1\text{e-}13$, so that significant rounding errors will occur, and the convergence of the method will necessarily slow down (see [92, §25.1], [173, §2.2]). Figure 7.1a shows that the convergence should be independent of the mesh parameters for this problem when the exact Jacobian matrix is used.

Using (3.66) and the definition of Fréchet differentiability (3.65), it is clear that, as the nonlinear approximation approaches the exact solution, the Jacobian matrix approximates the action of the nonlinear operator at the exact solution more and more closely. The



(a) Analytic Jacobian (b) Numeric Jacobian

Figure 7.1: Convergence history of Newton-MG applied to (7.9) with solution (7.19) where $C_1 = 1.0$, $\alpha = 1.0$ using an analytic and numeric Jacobian with coarsest grid 16^2

deterioration in convergence near the exact solution therefore suggests that the numeric Jacobian is a poor approximation to the actual Jacobian matrix at a fine grid resolution. However, it is strange that this should be the case only at fine grid resolutions since the same approximation is used to form the Jacobian matrix at lower grid resolutions, where the convergence closely matches that of the exact Jacobian matrix. To investigate the reasons why this could be some extra notation is introduced below.

Let a matrix $A \in \mathbb{R}^{n\times m}$ be denoted as

$$A = [a_{ik}], \quad i = 1, \ldots, n, \quad k = 1, \ldots, m.$$

For matrices $A, B, C \in \mathbb{R}^{n\times m}$ the difference $C = A - B$ is given by

$$C = [a_{ik} - b_{ik}], \quad i = 1, \ldots, n, \quad k = 1, \ldots, m. \tag{7.21}$$

Let $F^{(u_j)}$ denote the analytic Jacobian matrix at approximation u_j in space $\mathcal{V}_j$ on grid Ω_j, *i.e.*

$$F^{(u_j)} \equiv \left[F_{u_j}(\varphi_k^{(j)}, \varphi_i^{(j)})\right], \quad i, k = 1, \ldots, N_j, \tag{7.22}$$

and let $F_{\text{FD}}^{(u_j)}$ denote the numeric Jacobian calculated using a forward difference approximation. We also introduce the error matrix

$$E_{\text{FD}}^{(u_j)} = F_{\text{FD}}^{(u_j)} - F^{(u_j)}, \tag{7.23}$$

where

$$E_{\text{FD}}^{(u_j)} = \left[e_{ik}^{(j)} \right], \quad i,k = 1,\ldots,N_j. \tag{7.24}$$

Using the above notation we are ready to prove the following lemma, which bounds the size of the entries in the error matrix $E_{\text{FD}}^{(u_j)}$.

Lemma 7.1. *Each entry in the error matrix $E_{FD}^{(u_j)}$ is bounded by*

$$\left| e_{ik}^{(j)} \right| \leq C\epsilon h_j^{-1} \left(\epsilon h_j^{-1} + u_j^{(ik)} \right), \tag{7.25}$$

where $u_j^{(ik)}$ is the average of $|\nabla u|$ over

$$\omega_{ik}^{(j)} \equiv \left\{ T \in \mathcal{T}_j \mid T \cap \operatorname{supp}(\varphi_i^{(j)}) \cap \operatorname{supp}(\varphi_k^{(j)}) \neq \emptyset \right\}.$$

Proof. For the 4-Laplacian each entry in the error matrix is given by

$$e_{ik}^{(j)} = \frac{F(u_j + \epsilon\varphi_k)(\varphi_i) - F(u_j)(\varphi_i)}{\epsilon} - F_{u_j}(\varphi_k, \varphi_i). \tag{7.26}$$

Using the representation of $F(u_j)(\varphi_i)$ from (7.9), $F_{u_j}(\varphi_k, \varphi_i)$ from (7.10) and setting $\alpha = 1.0$ we show that

$$\begin{aligned} F(u_j + \epsilon\varphi_k)(\varphi_i) &= \int_\Omega |\nabla(u_j + \epsilon\varphi_k)|^2 \nabla(u_j + \epsilon\varphi_k)\nabla\varphi_i \,\mathrm{d}\vec{x} - \int_\Omega f\varphi_i \,\mathrm{d}\vec{x} \\ &= \int_\Omega \left(|\nabla u_j|^2 + 2\epsilon\nabla u_j\nabla\varphi_k + \epsilon^2 |\nabla\varphi_k|^2 \right) (\nabla u_j + \epsilon\nabla\varphi_k) \nabla\varphi_i \,\mathrm{d}\vec{x} \\ &\quad - \int_\Omega f\varphi_i \,\mathrm{d}\vec{x}, \end{aligned} \tag{7.27}$$

which simplifies to

$$\begin{aligned} F(u_j + \epsilon\varphi_k)(\varphi_i) = F(u_j)(\varphi_i) + \epsilon F_{u_j}(\varphi_k, \varphi_i) + \int_\Omega \epsilon^2 2(\nabla u_j\nabla\varphi_k)(\nabla\varphi_k\nabla\varphi_i) \,\mathrm{d}\vec{x} \\ + \int_\Omega \epsilon^2 |\nabla\varphi_k|^2 (\nabla u_j + \epsilon\nabla\varphi_k) \nabla\varphi_i \,\mathrm{d}\vec{x}. \end{aligned} \tag{7.28}$$

Therefore we have

$$e_{ik}^{(j)} = \int_\Omega \epsilon \left|\nabla\varphi_k\right|^2 (\nabla u_j \nabla\varphi_i)\,\mathrm{d}\vec{x} + \int_\Omega 2\epsilon(\nabla u_j\nabla\varphi_k)(\nabla\varphi_k\nabla\varphi_i)\,\mathrm{d}\vec{x} + \int_\Omega \epsilon^2 \left|\nabla\varphi_k\right|^2 (\nabla\varphi_k\nabla\varphi_i)\,\mathrm{d}\vec{x}. \quad (7.29)$$

We consider the absolute value $\left|e_{ik}^{(j)}\right|$, bounded by

$$\left|e_{ik}^{(j)}\right| \le \left|\int_\Omega \epsilon \left|\nabla\varphi_k\right|^2 (\nabla u_j \nabla\varphi_i)\,\mathrm{d}\vec{x}\right| + \left|\int_\Omega 2\epsilon(\nabla u_j\nabla\varphi_k)(\nabla\varphi_k\nabla\varphi_i)\,\mathrm{d}\vec{x}\right| + \left|\int_\Omega \epsilon^2 \left|\nabla\varphi_k\right|^2 (\nabla\varphi_k\nabla\varphi_i)\,\mathrm{d}\vec{x}\right|, \quad (7.30)$$

and consider each term in (7.30) individually.

For a piecewise linear basis ∇u_j and $\nabla\varphi_i$, $i = 1,\ldots,N_j$ are constant vectors on each element. Using standard finite element approximation properties [23, §II.6] $\left|\nabla\varphi_i^{(j)}\right| \le Ch_j^{-1}$ for some constant C independent of mesh parameters, for a quasi-regular triangulation. Then

$$\int_\Omega \epsilon \left|\nabla\varphi_k\right|^2 (\nabla u_j\nabla\varphi_i)\,\mathrm{d}\vec{x} = \sum_{T\in\omega_{ik}} \int_T \epsilon \left|\nabla\varphi_k^2\right| (\nabla u_j^T \nabla\varphi_i)\,\mathrm{d}\vec{x} \quad (7.31)$$

where the number of elements T satisfying $T \in \mathrm{supp}(\varphi_k)\cap\mathrm{supp}(\varphi_i)$ is small and independent of the mesh, and u_j^T is the function u_j restricted to element T. Let $\bar{u}_j^T$ be the average value of u_j on T and consider the integral over a single element

$$\begin{aligned}\left|\int_T \epsilon \left|\nabla\varphi_k\right|^2 (\nabla u_j^T\nabla\varphi_i)\,\mathrm{d}\vec{x}\right| &= \epsilon\left|\nabla\varphi_k\right|^2 \left|(\nabla u_j^T, \nabla\varphi_i)_{L^2,T}\right| \\ &\le \epsilon\left|\nabla\varphi_k\right|^2 \left\|\nabla u_j^T\right\|_{L^2,T} \left\|\nabla\varphi_i\right\|_{L^2,T} \\ &\le C\epsilon h_j^{-2} \left|u_j^T\right|_{1,T} \\ &\le C\epsilon h_j^{-1} \left|\nabla u_j^T\right|\end{aligned} \quad (7.32)$$

where the inner product $(\,\cdot\,,\,\cdot\,)_{L^2,T}$ is the L_2 inner product over element T, with induced norm $\|\cdot\|_{L^2,T}$. The H^1 semi-norm on element T is denoted $|\cdot|_{1,T}$, and $\left|\nabla u_j^T\right|$ is the absolute value of the gradient of u_j on element T. In (7.32) we have used the Cauchy-Schwarz inequality for functions and vectors, as well as the fact that for a piecewise linear

finite element basis function

$$\begin{aligned} \|\nabla\varphi_i\|^2_{L^2,T} &= \int_T |\nabla\varphi_i|^2 \,\mathrm{d}\vec{x} \\ &= |\nabla\varphi_i|^2 \int_T \mathrm{d}\vec{x} \\ &\leq Ch_j^{-2} A^{(T)} \leq C \end{aligned} \tag{7.33}$$

where $A^{(T)} \leq Ch_j^2$ is the area of element T. Similarly we have

$$\begin{aligned} \left|u_j^T\right|_{1,T} &= \left(\left|\nabla u_j^T\right|^2 \int_T \mathrm{d}\vec{x}\right)^{\frac{1}{2}} \\ &\leq Ch_j \left|\nabla u_j^T\right| \end{aligned} \tag{7.34}$$

using the fact that u_j is linear on each element T. Summing (7.32) over the elements $T \in \omega_{ik}$ gives

$$\begin{aligned} \left|\int_\Omega \epsilon \left|\nabla\varphi_k\right|^2 (\nabla u_j \nabla\varphi_i)\,\mathrm{d}\vec{x}\right| &\leq \sum_{T\in\omega_{ik}} \left|\int_T \epsilon \left|\nabla\varphi_k\right|^2 (\nabla u_j^T \nabla\varphi_i)\,\mathrm{d}\vec{x}\right| \\ &\leq C_1 \epsilon h_j^{-1} u_j^{(ik)}, \end{aligned} \tag{7.35}$$

recalling that $u_j^{(ik)}$ is the average of $|\nabla u_j|$ over ω_{ik}. Using similar reasoning

$$\begin{aligned} \left|\int_T 2\epsilon(\nabla u_j^T \nabla\varphi_k)(\nabla\varphi_k \nabla\varphi_i)\,\mathrm{d}\vec{x}\right| &\leq \int_T 2\epsilon \left|\nabla u_j \nabla\varphi_k\right| \left|\nabla\varphi_k \nabla\varphi_i\right| \,\mathrm{d}\vec{x} \\ &\leq 2\epsilon \left|\nabla\varphi_k\right|^2 \left|\nabla\varphi_i\right| \left|\nabla u_j^T\right| \int_T \mathrm{d}\vec{x} \\ &\leq 2\epsilon C h_j^{-3} \left|\nabla u_j^T\right| h_j^2 \\ &= \epsilon C h_j^{-1} \left|\nabla u_j^T\right|. \end{aligned} \tag{7.36}$$

Hence the integral over the entire domain is bounded as

$$\begin{aligned} \left|\int_\Omega 2\epsilon(\nabla u_j \nabla\varphi_k)(\nabla\varphi_k \nabla\varphi_i)\,\mathrm{d}\vec{x}\right| &\leq \sum_{T\in\omega_{ik}} \left|\int_T 2\epsilon(\nabla u_j^T \nabla\varphi_k)(\nabla\varphi_k \nabla\varphi_i)\,\mathrm{d}\vec{x}\right| \\ &\leq C_2 \epsilon h_j^{-1} u_j^{(ik)}. \end{aligned} \tag{7.37}$$

For the final term in (7.30) we have

$$\begin{aligned}\left|\int_T \epsilon^2 \left|\nabla\varphi_k\right|^2 (\nabla\varphi_k \nabla\varphi_i)\,\mathrm{d}\vec{x}\right| &\leq \int_T \epsilon^2 \left|\nabla\varphi_k\right|^2 \left|\nabla\varphi_k\right| \left|\nabla\varphi_i\right| \,\mathrm{d}\vec{x} \\ &= \epsilon^2 \left|\nabla\varphi_k\right|^3 \left|\nabla\varphi_i\right| \int_T \mathrm{d}\vec{x} \\ &\leq C\epsilon^2 h_j^{-2},\end{aligned} \tag{7.38}$$

so that

$$\begin{aligned}\left|\int_\Omega \epsilon^2 \left|\nabla\varphi_k\right|^2 (\nabla\varphi_k \nabla\varphi_i)\,\mathrm{d}\vec{x}\right| &\leq \sum_{T\in\omega_{ik}} \left|\int_T \epsilon^2 \left|\nabla\varphi_k\right|^2 (\nabla\varphi_k \nabla\varphi_i)\,\mathrm{d}\vec{x}\right| \\ &\leq C_3\epsilon^2 h_j^{-2}.\end{aligned} \tag{7.39}$$

Each entry in the error matrix $E_{\mathrm{FD}}^{(j)}$ is therefore bounded by

$$\begin{aligned}\left|e_{ik}^{(j)}\right| &\leq \epsilon h_j^{-1}\left(C_3\epsilon h_j^{-1} + (C_1+C_2)u_j^{(ik)}\right) \\ &\leq \max\{C_1,C_2,C_3\}\epsilon h_j^{-1}(\epsilon h_j^{-1} + 2u_j^{(ik)}). \qquad \square\end{aligned} \tag{7.40}$$

This shows that as the grids are refined the error in the Jacobian matrix increases, if the perturbation ϵ is kept constant. In general ϵ cannot be scaled relative to h_j, as there is a limit on how small ϵ can be taken before the quality of approximation will be affected by rounding errors [46]. This may have an effect on the quality of the approximation relative to the exact Jacobian, unless the entries in the Jacobian matrix are also scaled by the grid spacing. To investigate this we prove the following lemma.

Lemma 7.2. *The entry in row i, column k of the exact Jacobian matrix F_{u_j} is bounded by*

$$\left|F_{u_j}(\varphi_k,\varphi_i)\right| \leq C\left(u_j^{(ik)}\right)^2 \tag{7.41}$$

for $u_j^{(ik)}$ the average of ∇u_j over ω_{ik}.

Proof. The absolute value of the entry in row i, column k of the exact Jacobian matrix is given by

$$\left|F_{u_j}(\varphi_k,\varphi_i)\right| = \left|\int_\Omega \left|\nabla u_j\right|^2 \nabla\varphi_k\nabla\varphi_i\,\mathrm{d}\vec{x} + \int_\Omega 2(\nabla u_j\nabla\varphi_k)(\nabla u_j\nabla\varphi_i)\,\mathrm{d}\vec{x}\right|. \tag{7.42}$$

The bound of this is obtained by considering the sum of the integral over elements in ω_{ik} for each of the two terms in (7.42) separately. Firstly

$$\begin{aligned}\left|\int_T \left|\nabla u_j^T\right|^2 \nabla\varphi_k\nabla\varphi_i \,\mathrm{d}\vec{x}\right| &\leq \int_T \left|\left|\nabla u_j^T\right|^2 \nabla\varphi_k\nabla\varphi_i\right| \,\mathrm{d}\vec{x} \\ &\leq |\nabla\varphi_k|\,|\nabla\varphi_i|\left|u_j^T\right|_{1,T}^2 \\ &\leq Ch_j^{-2}\left|u_j^T\right|_{1,T}^2 \\ &\leq C\left|\nabla u_j^T\right|^2\end{aligned} \tag{7.43}$$

such that

$$\begin{aligned}\left|\int_\Omega |\nabla u_j|^2 \nabla\varphi_k\nabla\varphi_i \,\mathrm{d}\vec{x}\right| &\leq \sum_{T\in\omega_{ik}} \left|\int_T \left|\nabla u_j^T\right|^2 \nabla\varphi_k\nabla\varphi_i \,\mathrm{d}\vec{x}\right| \\ &\leq C\bar{u}_j^{(ik)} \\ &\leq C(u_j^{(ik)})^2,\end{aligned} \tag{7.44}$$

where $\bar{u}_j^{(ik)}$ is the average of $\left|\nabla u_j^T\right|^2$ over ω_{ik}. Let $\#\omega_{ik}$ be the number of elements in ω_{ik}. In the bound (7.44) we have used

$$\begin{aligned}\#\omega_{ik}(u_j^{(ik)})^2 &= \#\omega_{ik}\left[\frac{1}{\#\omega_{ik}}\sum_{T\in\omega_{ik}}\left|\nabla u_j^T\right|\right]^2 \\ &= \frac{1}{\#\omega_{ik}}\sum_{T\in\omega_{ik}}\left|\nabla u_j^T\right|^2 + \frac{1}{\#\omega_{ik}}\sum_{T\in\omega_{ik}}\sum_{\substack{T'\in\omega_{ik}\\T'\neq T}}\left|\nabla u_j^T\right|\left|\nabla u_j^{T'}\right| \\ &= \bar{u}_j^{(ik)} + \frac{1}{\#\omega_{ik}}\sum_{T\in\omega_{ik}}\sum_{\substack{T'\in\omega_{ik}\\T'\neq T}}\left|\nabla u_j^T\right|\left|\nabla u_j^{T'}\right| \\ &\geq \bar{u}_j^{(ik)}.\end{aligned} \tag{7.45}$$

Using similar reasoning

$$\begin{aligned}\left|\int_T 2(\nabla u_j^T\nabla\varphi_k)(\nabla u_j^T\nabla\varphi_i)\,\mathrm{d}\vec{x}\right| &\leq 2\left|\nabla u_j^T\right|^2|\nabla\varphi_k|\,|\nabla\varphi_i|\int_T \mathrm{d}\vec{x} \\ &\leq C\left|\nabla u_j^T\right|^2,\end{aligned} \tag{7.46}$$

so that

$$\begin{aligned}\left|\int_\Omega 2(\nabla u_j \nabla\varphi_k)(\nabla u_j \nabla\varphi_i)\,\mathrm{d}\vec{x}\right| &\leq \sum_{T\in\omega_{ik}} \left|\int_T 2(\nabla u_j^T \nabla\varphi_k)(\nabla u_j^T \nabla\varphi_i)\,\mathrm{d}\vec{x}\right| \\ &\leq C\bar{u}_j^{(ik)} \\ &\leq C(u_j^{(ik)})^2.\end{aligned} \tag{7.47}$$

Combining (7.43) and (7.47) gives the desired result

$$\left|F_{u_j}(\varphi_k,\varphi_i)\right| \leq C(u_j^{(ik)})^2 \qquad \square \tag{7.48}$$

Assume that ϵ is chosen constant. Under Assumption 7.1.b, $u_{ik}^{(j)}$ is independent of mesh parameters, *i.e.*

$$\left|e_{ik}^{(j)}\right| = \mathcal{O}(h_j^{-2}), \qquad \left|F_{u_j}(\varphi_k,\varphi_i)\right| = \mathcal{O}(1), \tag{7.49}$$

and asymptotically the relative error

$$\frac{\left|e_{ik}^{(j)}\right|}{\left|F_{u_j}(\varphi_k,\varphi_i)\right|} = \mathcal{O}(h_j^{-2}). \tag{7.50}$$

This behaviour is likely to be seen only when $h_j u_{ik}^{(j)} \leq C\epsilon$ for some small constant C. When the gradient is large (*i.e.* $u_{ik}^{(j)}$ is large) $h_j u_{ik}^{(j)}$ is likely to be much larger than ϵ, and from Lemmas 7.1 and 7.2 we see that it is likely that $\left|F_{u_j}(\varphi_k,\varphi_i)\right| \gg \left|e_{ik}^{(j)}\right|$ unless h_j is extremely small. Therefore the relative error is likely to be small where the gradient is large. However, around turning or inflexion points, where $u_{ik}^{(j)} \ll 1$, it is likely that the product $h_j u_{ik}^{(j)}$ will be of a similar size to (if not smaller than) ϵ, and the relative error will be large even at relatively modest mesh spacing such as spacings of the order 1e-3. We note that this is more likely to occur for small curvatures where larger regions of the solution have gradient close to zero, as is the case in the case of the example considered, where solution $u^* = \sin(\pi x)\sin(\pi y)$. Although the relative error is likely to be large only on small parts of the grid results are presented to demonstrate that this behaviour causes the breakdown in convergence observed for Newton's method.

In order to demonstrate that the relative error in few areas of the mesh has a large detrimental effect on convergence of Newton's method we seek to reduce the size of the

relative error. This can be done in a number of ways. Firstly, a more accurate approximation of the derivative can be taken when calculating the Jacobian matrix. For example, the central difference formula

$$F_{u_j}(\varphi_k, \varphi_i) = \frac{F(u_j)(\varphi_i + \epsilon\varphi_k) - F(u_j)(\varphi_i - \epsilon\varphi_k)}{2\epsilon}, \tag{7.51}$$

could be used rather than a forward difference formula when approximating the derivative. The error in approximation is then proportional to ϵ^2, rather than to ϵ, and the approximation will be much more accurate. Figure 7.2 shows the convergence history of a Newton-MG iteration over the first 7 Newton iterations when a central difference formula is used to approximate the entries in the Jacobian matrix. Comparison of Figure 7.2

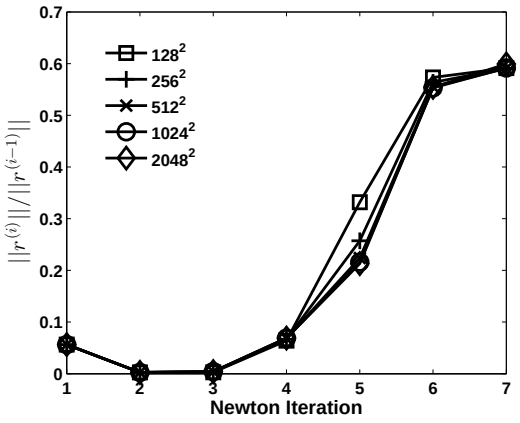


Figure 7.2: Convergence history of Newton-MG applied to (7.9) with solution (7.19) where $C_1 = 1.0$, $\alpha = 1.0$ using a central difference formula to approximate entries in the numeric Jacobian with coarsest grid 16^2.

with Figure 7.1a shows that the convergence with the more accurate numerical Jacobian matches that of the analytic Jacobian.

This result does not demonstrate that the improvement in the convergence is due to a reduction of the *relative* error, as the overall error is reduced by taking a more accurate approximation to the derivative. Therefore we consider how the relative error may be reduced for a forward difference approximation. In the discussion above it is noted that the relative error is large due to the solution used. Hence, changing the solution should change the convergence behaviour. Inspection of the bounds (7.25) and (7.41) shows that

the relative error satisfies

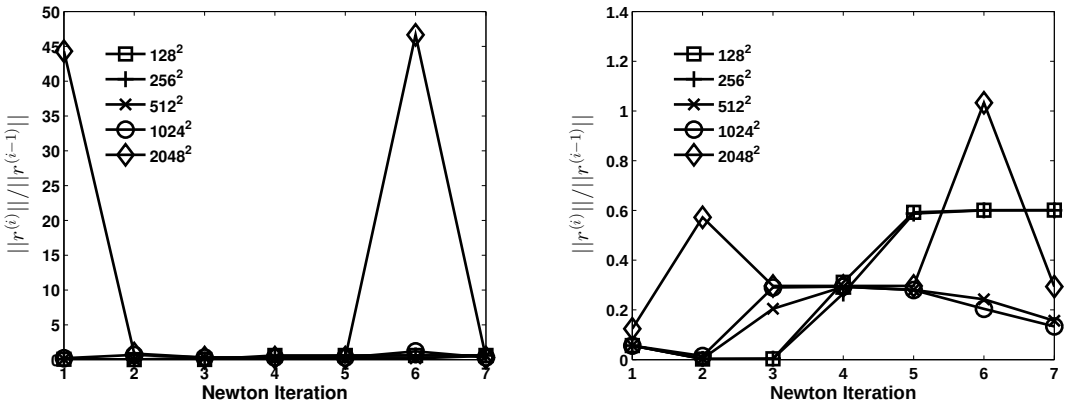
$$\frac{\left|e_{ik}^{(j)}\right|}{\left|F_{u_j}(\varphi_k, \varphi_i)\right|} \propto \frac{1}{C_1} \tag{7.52}$$

for C_1 as in (7.19) and fixed perturbation ϵ and grid spacing h_j. The linear system arising from the Newton linearisation is scaled by a factor C_1^2, so that the convergence properties of the iteration should not be affected by a change in C_1. Hence, if the convergence improves with increasing C_1 and deteriorates with decreasing C_1 this suggests that Newton's iteration is sensitive to the size of the relative error in the Jacobian matrix. Figure 7.3 shows the convergence histories for Newton-MG with varying C_1. The size of the perturbation ϵ is kept constant to demonstrate that the improvement in convergence is attributed only to a change in the parameter C_1. We can clearly see the improvement in the convergence of the method for increasing C_1 (*i.e.* decreasing relative error), and that any breakdown that does occur in the convergence occurs at later iterations when the defect is already very small (*i.e.* close to machine precision). This supports the proposition that the breakdown in convergence is caused by large relative errors on small parts of the domain. Results from another example supporting this claim are shown in Figure 7.5. This example is introduced after a note on a suitable choice of perturbation ϵ to use in the divided difference formula.

Increasing C_1 indefinitely in (7.19) will not necessarily stabilise the convergence when a forward difference approximation is used. Figure 7.4 shows the convergence when $C_1 = 1\text{e}5$ using different values for the perturbation to use in the calculation of the numeric derivative. As can be seen the same value that is used for smaller values of C_1 is no longer appropriate for the larger parameter. A much larger perturbation is appropriate to avoid rounding errors due to division by small numbers. This phenomenon is due to the fact that numerical differentiation is an inherently *unstable* process [46, pg. 175].

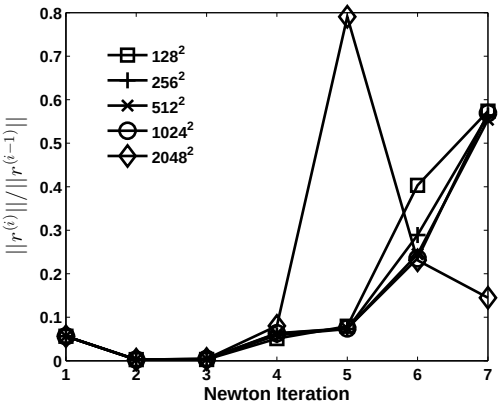
As well as increasing the size of the gradient on the domain the relative error should be reduced if the solution has no turning points. Consider that the exact solution to problem (7.9) is given by $u^* = \sin(\pi x)\sin(\pi y) + 2(x + y)$, such that there are no turning points of the solution in Ω. Figure 7.5 shows the convergence history of Newton-MG over the first 7 iterations for this updated solution using a forward difference approximation to entries in the Jacobian matrix. As can be seen the convergence is independent of mesh parameters, again supporting the hypothesis that the decrease in the size of the relative error in the Jacobian results in better convergence of the Newton iteration.

The results presented here are used to demonstrate the sensitivity of the Newton iter-

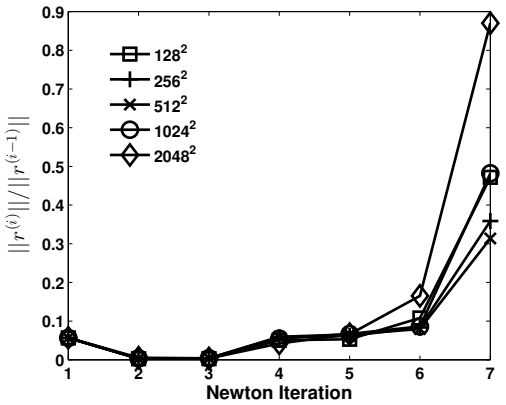


(a) $C_1 = 0.01$

(b) $C_1 = 0.1$

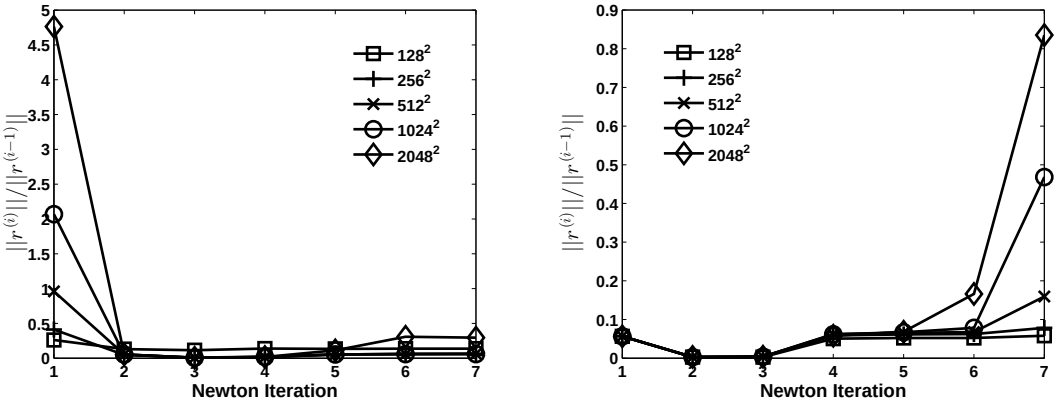


(c) $C_1 = 10.0$



(d) $C_1 = 100.0$

Figure 7.3: Convergence history of Newton-MG applied to (7.9) with solution (7.19) where, $\alpha = 1.0$ using a forward difference formula to approximate entries in the numeric Jacobian with coarsest grid 16^2 and varying C_1.



(a) $\epsilon = 4\text{e-}10$ (b) $\epsilon = 1\text{e-}6$

Figure 7.4: Convergence history of Newton-MG applied to (7.9) with solution (7.19) where, $\alpha = 1.0$, $C_1 = 1\text{e}5$ using a forward difference formula to approximate entries in the numeric Jacobian with coarsest grid 16^2.

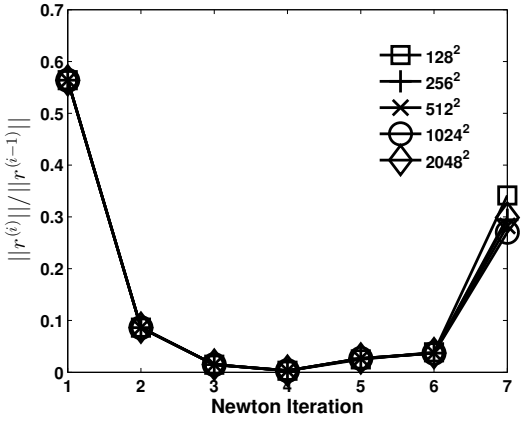


Figure 7.5: Convergence history of Newton-MG applied to (7.9) with exact solution $u^* = \sin(\pi x)\sin(\pi y) + 2(x + y)$ using a forward difference formula to approximate entries in the numeric Jacobian with coarsest grid 16^2.

ation to the numerical Jacobian matrix. For a general nonlinear problem it is not possible to say whether or not problems may arise when a simple approximation to the entries in the Jacobian matrix is taken. The discussion here serves to show that convergence may deteriorate, especially in cases when entries in the exact Jacobian matrix are small on part of the domain. We also demonstrate that a good choice of perturbation parameter depends on the approximation to the problem. Hence, for time-dependent problems, where the solution may evolve over time over a number of orders of magnitude, using a fixed perturbation may cause numerical instabilities to adversely affect the convergence of the Newton iteration. These instabilities can manifest themselves in slower convergence rates, as depicted in Figure 7.1b, or can lead to individual steps in the iteration being divergent, as in Figure 7.4a. In an extreme case the entire Newton iteration may be divergent. As an example of how much of an effect a bad value for the forward difference can have on the convergence, Figure 7.6 presents the convergence factors over the first 7 Newton iter-

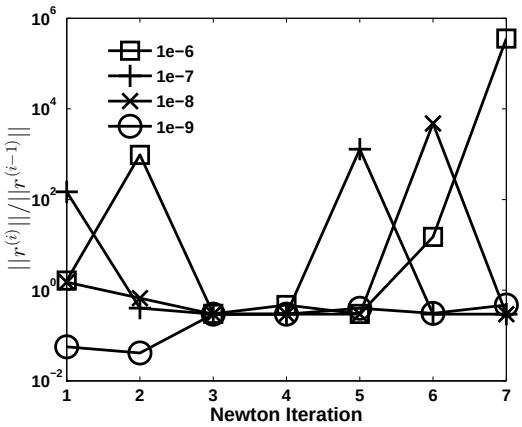
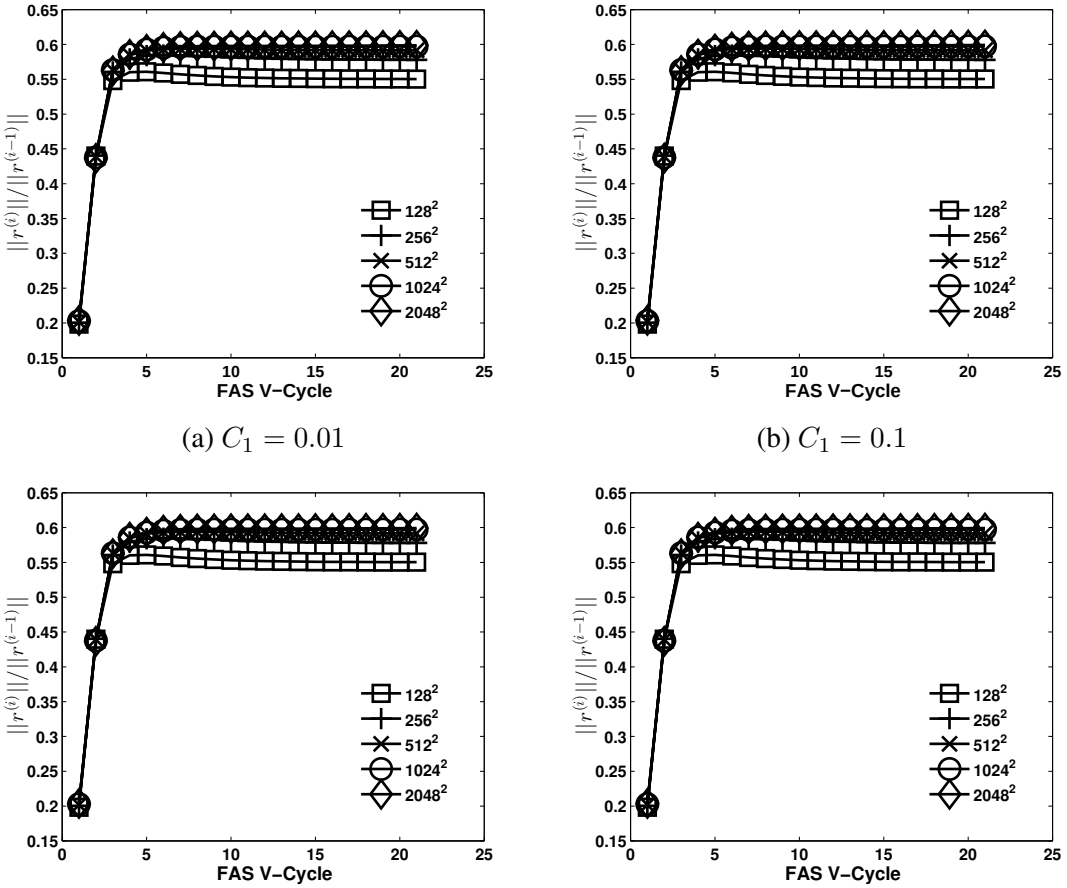


Figure 7.6: Convergence history of Newton-MG applied to (7.9) with solution (7.19) where $\alpha = C_1 = 1.0$ using a forward difference method to approximate the Jacobian matrix for various perturbation sizes ϵ on a 512^2 grid.

ations for problem (7.9), with exact solution given by (7.19) ($\alpha = C_1 = 1.0$), for varying sizes of perturbation used in a forward difference approximation. As can be seen only a single value (1e-9) does not give a history where at least one iteration gives an increase in the nonlinear residual, and the fact that the y-axis has a logarithmic scale shows the orders of magnitude over which the convergence factor may increase. In many of these situations the Newton iteration would be classified as divergent, although the convergence is known to be good for a good approximation to the Jacobian matrix.

In order to avoid instabilities of the type described here a more accurate approximation to the Jacobian matrix should be taken in the Newton iteration. In many cases it is possible to calculate the exact entries in the discrete Jacobian, especially when a low

order basis is used. Generally, though, calculating the Jacobian matrix to a large degree of accuracy may require more computational effort, but results for the running time of Newton-MG and FAS methods show that a Newton-MG iteration will still be a more efficient algorithm by some way (see Figure 7.8). Before discussing these results we contrast the sensitivity of the FAS iteration to the quality of the Jacobian matrix. Figure 7.7 shows



(a) $C_1 = 0.01$ (b) $C_1 = 0.1$

(c) $C_1 = 10.0$ (d) $C_1 = 100.0$

Figure 7.7: Convergence history of FAS V-Cycles applied to (7.9) with solution (7.19) where $\alpha = 1.0$ using a forward difference formula to approximate derivative contributions with coarsest grid 4^2 and varying C_1.

the convergence of the first 21 FAS V-cycles for varying values of C_1 using a forward difference approximation to the local derivative. 21 V-cycles are used here as this is the number of V-cycles performed in 7 Newton iterations, for which the results are presented above. As can be seen the convergence histories are almost identical in Figures 7.7a to 7.7d, which demonstrates that the FAS iteration is much more robust with respect to inaccuracies in the Jacobian matrix. Hence much less effort is required when coding a solution to find a robust iteration when the initial approximation is taken close to the exact

solution. We also see mesh independent convergence, and a clear asymptotic convergence factor of ~ 0.6. However, this convergence rate is not very quick, and is not what would one would wish for from an efficient linear multigrid iteration, in which a defect reduction of the order 1e-1 per V-cycle can often be observed.

For the problem being investigated here, when a good initial approximation is taken, a Newton iteration displays a better rate of convergence per V-cycle, on top of the faster execution time per V-cycle. Hence, the advantage of the Newton-MG iteration over FAS is very clear. Figure 7.8 shows the running times of the FAS iteration using a forward difference approximation to the derivative compared to a Newton iteration in which a forward difference and a central difference are used. The empty squares in the graph represent the running times for a forward difference, and the solid squares represent the running time for a central difference method. There is little difference in using the central or the forward difference method, although when a forward difference is appropriate it is slightly faster. Therefore, if speed is of the most importance a forward difference approximation should be preferred, bearing in mind that this method may lead to non-robust convergence behaviours. If a more robust method is desired a central difference formula should be preferred. Relative to both cases the FAS takes considerably longer to run. The numbers of V-cycles required for convergence, along with the exact running times are given in Table 7.4. The running times of the algorithms are shown in Figure 7.8 in graphical form.

Grid Size	FAS		Newton (FD)		Newton (CD)	
	V-Cycles	Time	V-Cycles	Time	V-Cycles	Time
128^2	26	0.82	12	0.11	12	0.12
256^2	28	3.52	12	0.46	12	0.50
512^2	29	20.08	12	2.08	12	2.36
1024^2	29	86.25	12	8.36	12	9.56
2048^2	29	354.64	27	57.40	12	38.44

Table 7.4: Number of V-cycles and running times required to reduce the initial residual by a factor 1e-7, which is depicted in Figure 7.8

From this the superiority of the Newton algorithm in terms of computational efficiency is clear to see. On the finest grid level, even in the case when the Newton iteration does not converge well, the Newton-MG iteration is 5.5 times faster than the FAS iteration. When the more costly central difference formula is used, the running time of Newton-MG is approximately 8 times faster than the FAS. Note that the running times presented include the set up of various grid parameters, and so are not entirely defined by the number of V-Cycles performed.

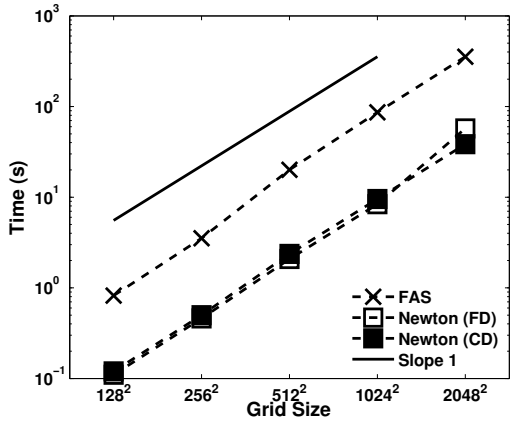


Figure 7.8: Running times for Newton-MG and FAS for problem (7.9) with solution (7.19) with $\alpha = 1.0$, $C_1 = 1.0$ using a numerical approximation to entries in the Jacobian matrix.

Results presented in this section show that, for the 4-Laplacian, when an approximation is close to the exact solution, the Newton iteration is a much better algorithm to use, provided that a sufficiently accurate approximation the Jacobian matrix is used. The results are qualitatively the same for $p > 4$, but are not presented in this thesis. Furthermore, a discussion has been given in order to show how, in cases when the Newton iteration seems to be non-convergent, this behaviour can be caused by a number of implementation, rather than theoretical, issues. This is an important note to make, since in previous publications (see [86]) it is stated that the Newton iteration is less robust than the FAS iteration. We do not believe this to be the case, and in Section 7.6 a discussion of globalisation techniques provides strong evidence suggesting that the convergence of an FAS iteration is dependent on the convergence of a Newton iteration. In the next section we consider a comparison of the methods when the initial approximation is further away from the exact solution, thus investigating the robustness of the methods with respect to an initial approximation.

7.5 Convergence for Poor Initial Guesses

In this section, as in the previous section, we consider the convergence of nonlinear multigrid methods for (7.9) when the coefficient function α is constant and the exact solution is known. Initial approximations are considered which are far from the exact solution, but are similar in shape and also initial approximations which are near values at which the Jacobian matrix is singular.

Initial approximations are taken which are similar in shape to the exact solution, since for a completely random initial guess it is likely that the nonlinear iterations will not be convergent. Also, for time-dependent problems, if an initial approximation is taken from a previous time step it is likely to be similar in shape to the solution at a current time step provided that a small enough time step is used. Considering cases where the problem is singular is also interesting, as it may be that a sequence of nonlinear iterates may unintentionally pass close to approximations which are singular before reaching the exact solution. A method should be as robust as possible in these cases.

The experiments undertaken demonstrate the superior robustness of Newton's method over FAS. Globalisation of the methods, which may increase the robustness, is not considered in this section, but is discussed in Section 7.6. The detailed discussion of the choice of algorithm components, in particular the choice of the perturbation in the approximation of the numerical derivative as discussed in the previous section, is not repeated here. Results presented are for best case executions of both algorithms. This makes the comparison of the methods fairer as the best performing variants of each algorithm are compared.

For the remainder of this section we set the exact solution to (7.9) to be

$$u^* = \sin(\pi x)\sin(\pi y). \tag{7.53}$$

To begin with, initial approximations on grid Ω_j of the form

$$u_j^{(0)} = C_2 \sin(\pi x)\sin(\pi y) \tag{7.54}$$

are considered for some constant C_2. Results in the previous section are for $C_2 = 0.95$, which may be seen to be close enough to the exact solution that super-linear convergence is gained for the Newton iteration. This convergence behaviour is expected only close to the exact solution, and outside of a ball of guaranteed convergence at best linear convergence of the Newton method can be expected, if at all [58]. It is not known what convergence behaviour to expect far from the exact solution in the case of FAS, although at best linear convergence is observed even close to the exact solution. Since there is no theory to predict convergence behaviour of the FAS, results from empirical experiments are presented and investigated below.

An initial approximation is considered far from the exact solution for values of C_2 in (7.54) for which $|C_2 - 1| > 1$. In this case the initial approximation is far from the exact solution in the sense that the absolute values of the error in the initial approximation are larger than the absolute values of the solution. Table 7.5 shows the number of V-

cycles required for convergence for various $|C_2| \geq 1$. In all tables, an entry 'div' denotes

	Grid	C_2				
		-100	-10	-1	10	100
Newton-MG	64^2	51	33	30	27	39
	128^2	51	39	27	27	39
	256^2	51	39	27	27	39
	512^2	51	42	30	27	39
	1024^2	51	45	33	27	39
	2048^2	51	48	36	27	39
FAS	64^2	13	18	24	28	12
	128^2	26	20	28	19	20
	256^2	div	23	30	20	div
	512^2	14	div	div	21	16
	1024^2	div	28	>30	div	div
	2048^2	div	>30	>30	div	div

Table 7.5: Number of V-cycles required to reduce the initial residual in approximation by a factor 1e-7 for FAS and Newton-MG for initial guesses $u^{(0)} = C_2 \sin(\pi x) \sin(\pi y)$ with varying C_2.

divergence of the method.

It is now useful to recall the discussion from Subsection 4.2.3, in which the coarse grid correction of FAS is shown to be close to a Newton correction on the coarse grid for an approximation close to the exact solution. Using this theory, assuming convergence of the nonlinear smoother (Lemma 3.10) and convergence of the Newton iteration (Theorem 3.8) suggests that an FAS iteration will converge for a sufficiently good approximation. For an approximation further from the exact solution the FAS coarse grid correction no longer approximates a Newton correction, so convergence theory from Newton methods cannot be used to predict how, or if, an FAS iteration will converge.

Results in Table 7.5 suggest that it is likely to be very difficult to predict the way in which an FAS iteration will converge, if at all. There is no discernible pattern in the number of iterations required for convergence and convergence does depend on mesh parameters. It is clear that FAS is much more sensitive to a poor choice of initial guess, in that the absolute value between the exact and approximate solutions should be small for convergence to be guaranteed. To illustrate the variability in FAS convergence histories, convergence rates on various grids, for the initial approximation with $C_2 = 100$, are shown in Figure 7.9. As previously noted there is no obvious pattern in the convergence histories, and the ratios between one iteration and the next may vary over several orders

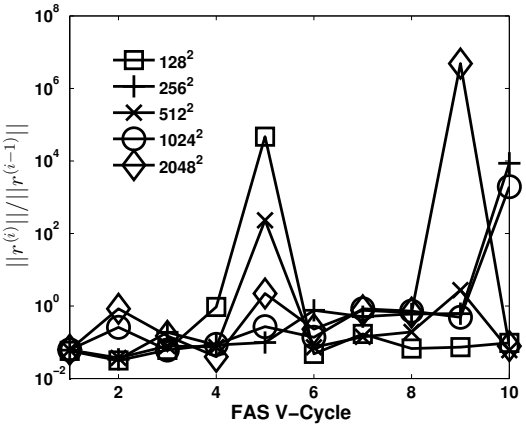
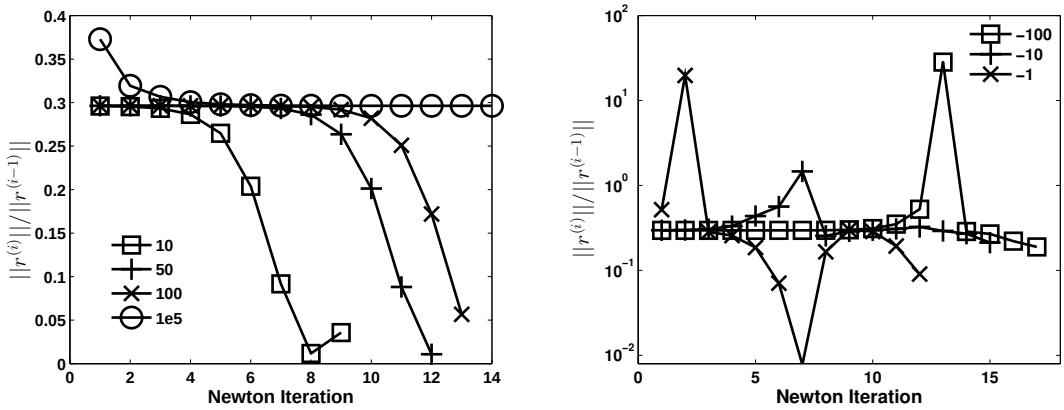


Figure 7.9: Convergence history for FAS on varying grids with initial estimate $u^{(0)} = 100\sin(\pi x)\sin(\pi y)$.

of magnitude.

In contrast Newton-MG performs very well. Results in Table 7.5 show that Newton-MG converges for all values of C_2 used. The convergence behaviour is different for positive and negative values, though, which is worth discussing further. For positive C_2 the convergence is robust and independent of mesh parameters, but not of C_2. Figure 7.10 includes the convergence histories for a wider range of values of C_2 than are included in Table 7.5. A clear asymptotic regime of linear convergence and then super-linear convergence can be observed. The line for $C_2 = 1\text{e}5$ hints at the convergence behaviour for larger values of C_2, where convergence starts to deteriorate. However, even for large C_2 the convergence of the Newton iteration is very predictable. The more interesting behaviour is for $C_2 < 0$. In this case a spike in the convergence history is observed. This occurs at a later iteration the larger the absolute value of C_2. Before this spike the convergence is robust, and the rate is the same linear rate observed in Figure 7.10a for positive values of C_2. The issue here is that the approximation approaches the solution from below, and must pass through a stage where the approximation is close to zero. Inspection of Equation (7.9) shows that the Jacobian is singular when the approximation $u = 0$, and is close to singular for $u \approx 0$. This can cause problems for the Newton method, and may cause a correction to be over approximated, in which case the residual norm may increase. In order to stabilise the method some damping strategy may be employed. This approach is discussed in Section 7.6. For now the convergence properties of the method without any damping parameters are considered.

Both FAS and Newton-MG diverge in a single iteration when an initial approximation of zero is used for problem (7.9). This is due to the fact that the Jacobian is singular (in



(a) $C_2 > 1$ (b) $C_2 \leq -1$

Figure 7.10: Convergence history for Newton-MG for varying initial approximations with similar shape to the exact solution.

fact all entries are zero), so that the Newton iteration and the smoother diverge immediately. For initial estimates where the Jacobian is almost singular it can be expected that convergence of a Newton iteration will be poor, if it is successful at all. Results in Table 7.6 show that FAS also does not converge well when the Jacobian is almost singular. In practice it is possible that a sequence of iterates may pass close to an approximation at which a problem is no longer well defined, such as in this case. A robust numerical method should be able to pass close to one of these values without diverging.

	Grid	C_2				
		0.05	0.1	0.2	0.4	0.7
Newton-MG	64^2	div	42	33	21	15
	128^2	div	42	33	24	15
	256^2	div	45	33	24	15
	512^2	div	45	33	24	18
	1024^2	div	54	42	30	18
	2048^2	div	57	45	30	18
FAS	64^2	27	29	26	24	23
	128^2	>30	>30	>30	28	26
	256^2	>30	div	30	>30	29
	512^2	div	div	>30	>30	30
	1024^2	div	div	>30	div	30
	2048^2	div	div	div	div	30

Table 7.6: Number of V-Cycles required to reduce the initial residual in approximation by a factor 1e-7 for FAS and Newton-MG for initial guesses $u^{(0)} = C_2 \sin(\pi x) \sin(\pi y)$ with $0 < C_2 < 1$.

Table 7.6 shows the number of V-cycles performed for varying small C_2. The results clearly show that the Newton iteration is the more robust with respect to the initial approximation across all grid sizes. If an iteration is convergent on a coarse grid it will also be convergent on a fine grid, which is not observed for FAS. Far away from the exact solution, convergence is not independent of the mesh parameters, whereas for Newton-MG the convergence histories, shown in Figure 7.11 for the case $C_2 = 0.1$, are very similar. It is clear from this figure that the problem with the Newton iteration is the very poor

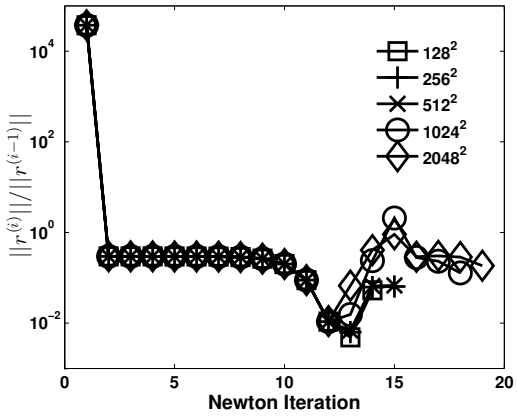


Figure 7.11: Convergence history of Newton-MG for $C_2 = 0.1$

first iteration. The phenomenon of Newton's iteration giving a correction which is too large is well researched [58, §3.1.3], and it is for this reason that a damping strategy is often employed in order to ensure that the iterates remain within a *trust region* [58, §3]. Damping strategies are available for both the Newton iteration and the FAS iteration and their effectiveness is discussed in the next section.

7.6 Globalisation Techniques

In this section we look at techniques designed to stabilise convergence of Newton and FAS methods. In practice convergence of a Newton iteration is known to be unpredictable outside of *trust regions* or balls of guaranteed convergence. Figure 7.9 shows that the same can be true for an FAS iteration. In order to increase the size of trust regions, scalar damping parameters are used to control the size of a correction term. For Newton's method usually a single scalar parameter is introduced, as in Algorithm 3.4. One method of stabilisation of FAS is given in Algorithm 4.4, in which a scaling parameter is introduced to control the size of the coarse grid correction term (see lines 6-12 in Algorithm 4.4). An alternative is the Damped Nonlinear Multilevel Method (DNMLM), introduced in [87, 88],

which is outlined in its original form in Algorithm 7.1. This solves the discrete nonlinear problem

$$A_J(u_J) = f_J.$$

Compared with the NMLM (see Algorithm 4.4) there are two additions to Algorithm 7.1.

Algorithm 7.1 Damped Nonlinear Multilevel Method

Require: $A_j, A_{j-1}, \tilde{u}_{j-1}, S_j, P_{j-1}^j, R_j^{j-1}$

1: **function** DNMLM($j, u_j, f_j, \gamma, \nu_1, \nu_2$)
2: $u_j \leftarrow S_j^{\nu_1}(f_j, u_j)$ ▷ *Pre-smooth*
3: $f_{j-1} \leftarrow R_j^{j-1}(f_j - A_j(u_j))$
4: $s \leftarrow \sigma(j-1, f_{j-1})$ ▷ *Calculate scaling parameter*
5: $f_{j-1} \leftarrow A_{j-1}(\tilde{u}_{j-1}) + s f_{j-1}$ ▷ *Use pre-defined $\tilde{u}_{j-1}$ on coarse grid*
6: **if** $j = 2$ **then**
7: $u_{j-1} \leftarrow A_{j-1}^{-1}(f_{j-1})$
8: **else**
9: **for** $i = 1, \ldots, \gamma$ **do**
10: $u_{j-1} \leftarrow$ DNMLM($j-1, u_{j-1}, f_{j-1}, \gamma, \nu_1, \nu_2$)
11: $e_{j-1} \leftarrow (u_{j-1} - \tilde{u}_{j-1})/s$
12: $\psi \leftarrow \psi(f_j, u_{j-1}, u_j)$ ▷ *Calculate damping parameter*
13: $u_j \leftarrow u_j + \psi P_{j-1}^j e_{j-1}$
14: $u_j \leftarrow S_j^{\nu_2}(f_j, u_j)$ ▷ *Post-smooth*

The first is the damping parameter ψ (see lines 12 and 13), and the second is the introduction of the coarse grid approximations $\tilde{u}_{j-1}$, $j = 2, \ldots, J$ (see line 5).

The coarse grid approximations $\tilde{u}_{j-1}$ are specified at the start of the algorithm and are used to simplify the calculation of a 'good' scaling parameter s (see line 4). From the discussion in Subsection 4.2.3 and study of the application of the method [85–88] we see that a 'good' scaling parameter is one such that the coarse grid correction closely approximates a Newton correction. For a steady-state problem, using information regarding current approximations, the right-hand side and expected solution, it is feasible that a set of approximations $\tilde{u}_{j-1}$, $j = 2, \ldots, J$ may be chosen so that the parameter s becomes easier to compute. However, for a time-dependent problem it is likely that a single set of $\tilde{u}_{j-1}$ will not be appropriate at all time steps as the solution evolves. Therefore, as part of a time-dependent solve some computational effort will be required to select a 'good' set of coarse grid approximations. To the best of our knowledge there exists no literature describing how a set of coarse approximations should be chosen, and there exists no literature where this method is applied to a time-dependent problem. This suggests that

this approach is either too complicated or too costly to be a competitive algorithm. In our research we have found it difficult to select a good set of coarse approximations. Results obtained did not show robust convergence behaviour, so they are not presented in this thesis, where we restrict our comparisons to the use of FAS.

If the initial approximation on each grid varies, as in Algorithm 4.4, the choice of an appropriate scaling parameter s in line 5 of Algorithm 4.4 becomes difficult. Again, as far as we are aware, there exists no literature describing how an appropriate parameter s can be chosen.

The choice of damping parameter ψ (see line 12 of Algorithm 7.1) is discussed in [87,88]. The calculation is based on sound theory and, in combination with a good scaling s, allows for a proof of convergence of the method to be obtained for simple nonlinear problems (see [88]) where the nonlinearity is in a source term. Convergence has not been shown for problems with a nonlinearity in the diffusion coefficient. In [88], the calculation of ψ requires the calculation of the Jacobian matrix on the fine grid, and requires an extra residual calculation on the fine grid. This adds considerable computational costs into what we have already shown is an expensive iteration (see Chapter 6), and may require the storage of the Jacobian matrix. The fact that FAS does not need to store the Jacobian matrix is one of the few advantages of an FAS iteration over a Newton iteration that we have observed so far, so that the storage of this is undesirable.

In contrast, there exist simple procedures for stabilising Newton's method which are computationally cheap compared to the method for stabilising DNMLM in [88]. For this reason, as well as the fact that the DNMLM has not been adopted in practice, we do not consider the application of the DNMLM. Results in the rest of this section also demonstrate that the NMLM, as given in Algorithm 4.4, is a much less desirable iteration than a global Newton method.

The damping strategy that is employed for the Newton iteration for the results presented in this section is the popular Armijo rule [3], which often works well in practice. This is one of the most simple globalisation procedures available. Before stating the rule some necessary notation is introduced. We recall from (7.11) that component i of the residual is given by

$$F(u)(\varphi_i), \quad i = 1, \ldots, N_j.$$

For an approximation u to the exact solution $u_j^* \in \mathcal{V}_j$ the residual vector $F(u)$ is defined

as

$$F(u) \equiv [F(u)(\varphi_i)]_{i=1}^{N_j}, \tag{7.55}$$

and the discrete L_2 norm of the residual is given by

$$\|F(u)\|^2 = \frac{1}{N_j} \sum_{i=1}^{N_j} |F(u)(\varphi_i)|^2 . \tag{7.56}$$

Armijo's rule is given in Algorithm 7.2 and provides a simple procedure for choosing an appropriate Newton update once the correction term $\delta^{(k)}$ has been calculated at the k^{th} Newton iteration. The choice of scaling factor, γ, is theoretically justified, see [3], but the

Algorithm 7.2 Armijo's Rule

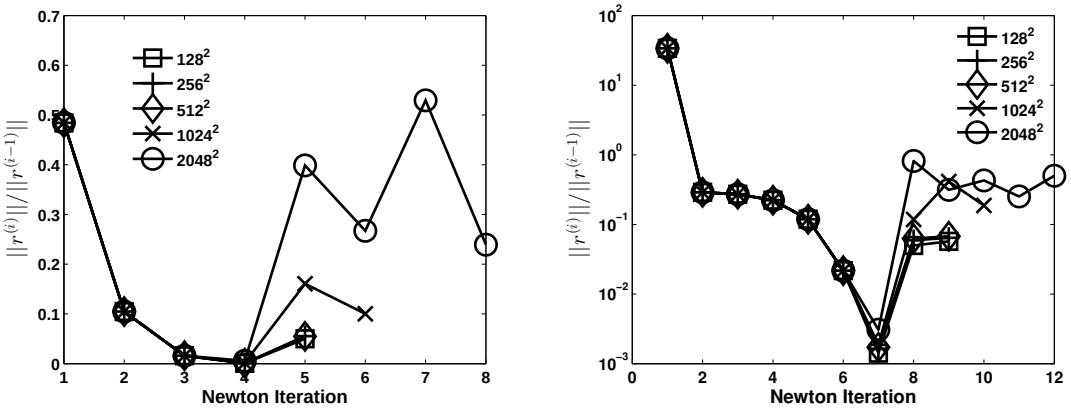
Require: $\Gamma = \{1, 1/2, 1/4, \ldots, \gamma_{\min}\}$, F, $u_j^{(k)}$

1: $l \leftarrow 1, \gamma \leftarrow \Gamma_l$
2: **while** $\left\| F(u_j^{(k)} + \gamma\delta^{(k)}) \right\|^2 > (1 - \frac{1}{2}\gamma) \left\| F(u_j^{(k)}) \right\|^2$ and $\gamma \neq \gamma_{\min}$ **do**
3: $\quad l \leftarrow l + 1, \gamma \leftarrow \Gamma_l$
4: $u_j^{(k+1)} \leftarrow u_j^{(k)} + \gamma\delta^{(k)}$

discussion of this is outside of the scope of the thesis.

In the following, Newton's method damped with Armijo's rule is compared to a NMLM. We begin with an investigation of how the robustness of the algorithm is improved for approximations where the Jacobian (7.22) is nearly singular. Results from the previous section (see Figure 7.11) show that the Newton correction is over-approximated (*i.e.* too large) for such approximations. Using Armijo's rule to damp the correction stabilises the correction term, and improved convergence can be observed in Figure 7.12a when compared to Figure 7.11. The increase in robustness is clear. The iteration also seems to give mesh independent convergence. The deterioration in the convergence on finer grids can be attributed to rounding errors, as the residuals are of the order 1e-12 when the convergence deteriorates on the finer grids.

Figure 7.12b demonstrates that a stabilised Newton method is still not convergent at every iteration for certain initial approximations. However, the stabilisation of the method is clear. Convergence is obtained for an initial approximation $u^{(0)} = 0.01u^*$ in Figure 7.12b, whereas a divergent iteration occurs at $u^{(0)} = 0.05u^*$ in Table 7.6. The non-damped Newton method diverges after a single iteration when using the initial approximation shown in Figure 7.12b, but the damped variant is ultimately convergent on all grid



(a) $u^{(0)} = 0.1u^*$ (b) $u^{(0)} = 0.01u^*$

Figure 7.12: Convergence histories for Newton-MG stabilised using Armijo's rule for problem (7.9) with exact solution $u^* = \sin(\pi x)\sin(\pi y)$ and varying initial approximations. γ_{min} in Armijo's rule is set to 2^{-10}.

levels. Note that the convergence can be obtained for initial approximation $u^{(0)} = 0.01u^*$ on every iteration if γ_{min} is reduced to $\sim$ 1e-4. This comes at a computational expense however, since the number of times the nonlinear residual needs to be recalculated is at worst 14, as $2^{-14} \approx$ 1e-4. This can be a considerable cost, and it may also be that a small damping parameter can cause the method to converge unnecessarily slowly. Therefore, small values of the damping parameter are often avoided in practice. Ultimately convergent, rather than monotone convergent iterations, are acceptable.

For the NMLM it has not been possible, through a combination of looking for any theory in the literature, or undertaking empirical experiment, to find a suitable s for problem (7.9) so that convergence is stabilised in a predictable manner. However, for the purposes of this discussion we assume a 'good' parameter has been chosen and consider what effect this good scaling *should* have on the convergence of the method. From the discussion in Subsection 4.2.3, as well as the literature in which NMLM is described [85–88], the scaling parameter is used in order to make the coarse grid correction approximate a Newton correction. The reason for this is made clear in the discussion in Subsection 4.2.3. Therefore, an *ideal* s is defined as one that causes the NMLM coarse grid correction, calculated in lines 6 to 12 of Algorithm 4.4, to be the actual Newton correction. To investigate the convergence of an ideal correction Algorithm 7.3 is introduced, which is a hybrid of FAS and Newton for solving the nonlinear problem

$$F(u) = 0.$$

The procedure is called *ideal NMLM*, and is a combination of nonlinear smoothing on a fine grid and an (approximate) Newton correction on the coarse grid. The solve in line 4

Algorithm 7.3 Ideal NMLM

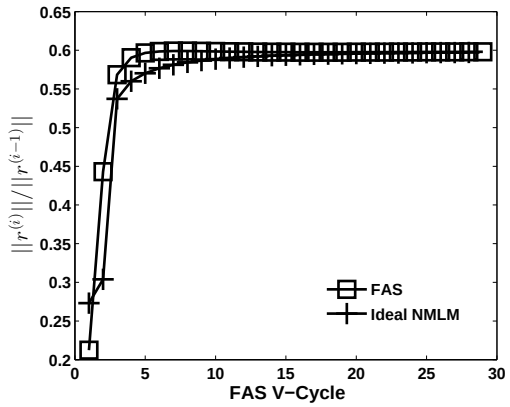
Require: S_j, P_{j-1}^j, R_j^{j-1}

1: **function** FAS-NEWTON($j, u_j, \gamma, \nu_1, \nu_2$)
2: $u_j \leftarrow S_j^{\nu_1}(0, u_j)$ ▷ *Perform pre-smoothing*
3: $F'_{j-1} \leftarrow R_j^{j-1} F'_j P_{j-1}^j$ ▷ *Set the coarse grid Jacobian*
4: $\delta_{j-1} \leftarrow (F'_{j-1})^{-1} \left(R_j^{j-1} F(u_j)\right)$ ▷ *Calculate the Newton coarse grid correction*
5: $u_j \leftarrow u_j - P_{j-1}^j \delta_j$
6: $u_j \leftarrow S_j^{\nu_2}(0, u_j)$ ▷ *Perform post-smoothing*

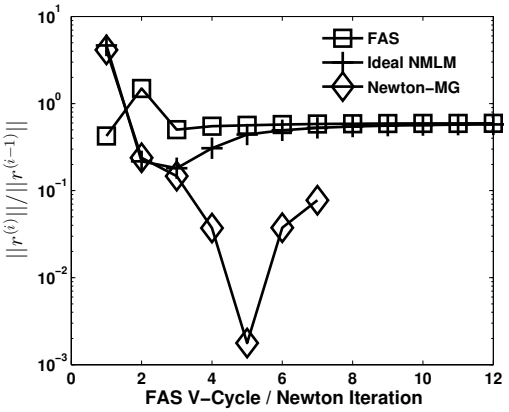
may be replaced by an approximate solve, such as a fixed number of multigrid iterations. The results presented in this section are for the case that the linear solve is performed approximately using a fixed number (three) linear multigrid iterations. Figure 7.13 shows some interesting behaviour of the ideal NMLM iteration. Firstly, in Figure 7.13a, the convergence of the ideal NMLM and FAS are compared when the approximation is close to the exact solution. The theory from Subsection 4.2.3 shows that in this case the FAS iteration should be close to the ideal update. This behaviour is certainly observed, and the convergence of both methods is asymptotically equal.

The convergence behaviour further away from the exact solution highlights some more interesting behaviour. In Figures 7.13b to 7.13d we see that the ideal NMLM converges like a Newton method further from the solution, but when the Newton iteration enters the super-linear regime, the convergence of the ideal NMLM tends to the same convergence as FAS. In Figure 7.13b the initial approximation is closer to a value for which the Jacobian (7.22) is singular. Ideal NMLM displays the same over-approximation of the correction as the Newton iteration. The convergence of the FAS iteration is less stable for the first few iterations. The over-approximation of the Newton step can be damped using an Armijo strategy, but it is not known how the FAS should be stabilised.

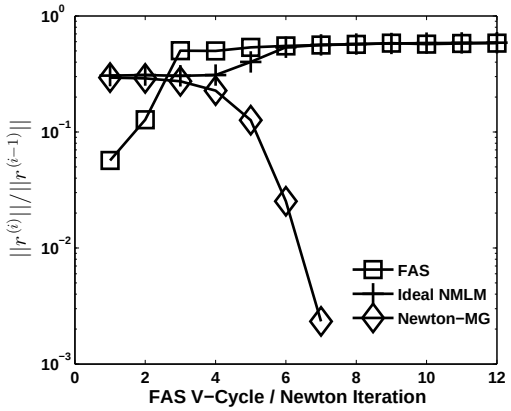
When an initial approximation far from the solution is taken (see Figures 7.13c and 7.13d) the ideal NMLM again converges like a Newton method until the asymptotic super-linear regime, when it then converges like FAS. As previously noted, the convergence of FAS outside of a trust region is unpredictable. In Figure 7.13c the convergence is very good for the first few iterations, and operates better than an ideal NMLM. However, there is no theoretical justification as to when the FAS iteration should perform this well. It is just as likely that the iteration may not monotonically decrease, as in Figure 7.13b, or that it diverges, as in Figure 7.13d.



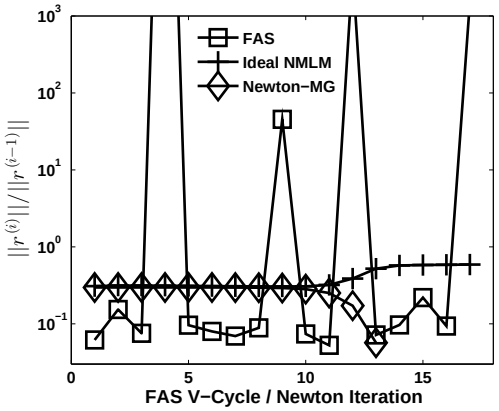
(a) $u^{(0)} = 0.9\sin(\pi x)\sin(\pi y)$.



(b) $u^{(0)} = 0.5\sin(\pi x)\sin(\pi y)$.



(c) $u^{(0)} = 5\sin(\pi x)\sin(\pi y)$.



(d) $u^{(0)} = 100\sin(\pi x)\sin(\pi y)$.

Figure 7.13: Convergence of FAS, ideal NMLM and Newton-MG for problem (7.9) with $\alpha = 1.0$ on a 1024^2 grid for varying initial approximations.

This section highlights that for a problem with a continuous right-hand side FAS lacks the fast convergence of Newton's method close to the exact solution, and lacks the robustness, or an easy way to regain robustness, for approximations far from the exact solution. This, combined with the better computational efficiency demonstrated in Chapter 6, makes Newton's method by far the better choice for this problem. This behaviour is observed also for other more complicated problems presented in Sections 7.7, 8.1 and 8.3. We now turn our attention to the case in which there are large discontinuities in a diffusion coefficient on the domain.

7.7 Convergence for Discontinuous α

In this section we consider the application of Newton-MG and FAS to the problem (7.9), which is given again in (7.57) below:

$$\begin{gathered} F(u)(v) \equiv \int_\Omega \alpha(x) \left|\nabla u\right|^2 \nabla u \nabla v \,\mathrm{d}x - \int_\Omega f v \,\mathrm{d}x = 0, \quad x \in \Omega, \\ u = 0, \quad x \in \partial\Omega. \end{gathered} \tag{7.57}$$

The linearisation is given in (7.10), and repeated in (7.58):

$$F_u(v, w) = \int_\Omega \alpha \left|\nabla u\right|^2 \nabla w \nabla v \,\mathrm{d}x + \int_\Omega 2\alpha (\nabla u \nabla w)(\nabla u \nabla v) \,\mathrm{d}x. \tag{7.58}$$

In the previous sections α was constant throughout the domain. Now we consider the case that $\alpha > 0$ is piecewise constant and discontinuous. The theoretical detail of the convergence of linear multigrid methods applied to a problem with highly varying coefficients is outside of the scope of this thesis, but a note is made as to how the theory can be adapted for the current problem. The theory presented in [156] (summarised in Subsection 5.1.3) is for a symmetric positive definite linear problem of the form

$$-\nabla \cdot (\alpha \nabla u) = f, \tag{7.59}$$

where boundary conditions are not specified. Any bounds obtained in the theory rely on the equivalence of the energy norm

$$\|u\|_\alpha^2 = \int_\Omega \alpha \left|\nabla u\right|^2 \,\mathrm{d}x \tag{7.60}$$

with the H_1 norm. For (7.58) the energy norm is given by

$$\|v\|_{u,\alpha}^2 = \int_\Omega \alpha \left|\nabla u\right|^2 \left|\nabla v\right|^2 \, \mathrm{d}x + \int_\Omega 2\alpha (\nabla u, \nabla v)^2 \, \mathrm{d}x. \tag{7.61}$$

We introduce the notation $A \eqsim B$ to mean that $A \leq CB$ and $B \leq cA$ for some benign constants C, c. Taking $\alpha \left|\nabla u\right|^2$ as the coefficient function in (7.60) such that

$$\|v\|_{\alpha|\nabla u|^2}^2 = \int_\Omega \alpha \left|\nabla u\right|^2 \left|\nabla v\right|^2 \, \mathrm{d}x, \tag{7.62}$$

the following lemma is simple to prove.

Lemma 7.3. *The energy norm* $\|v\|_{u,\alpha}$ *is equivalent to* $\|v\|_{\alpha|\nabla u|^2}$, i.e.

$$\|v\|_{u,\alpha} \eqsim \|v\|_{\alpha|\nabla u|^2} . \tag{7.63}$$

Proof. From the definition of the two norms we immediately have

$$\|v\|_{\alpha|\nabla u|^2} \leq \|v\|_{u,\alpha} . \tag{7.64}$$

To show the reverse inequality requires application of the Cauchy-Schwarz inequality for vectors as follows

$$\begin{aligned} \|v\|_{u,\alpha}^2 &= \int_\Omega \alpha \left|\nabla u\right|^2 \left|\nabla v\right|^2 \, \mathrm{d}x + \int_\Omega 2\alpha (\nabla u, \nabla v)^2 \\ &\leq \int_\Omega \alpha \left|\nabla u\right|^2 \left|\nabla v\right|^2 \, \mathrm{d}x + \int_\Omega 2\alpha \left|\nabla u\right|^2 \left|\nabla v\right|^2 \, \mathrm{d}x \\ &= 3 \|v\|_{\alpha|\nabla u|^2}^2 . \end{aligned} \tag{7.65}$$

□

Using the above lemma, letting $v, w \in H_0^1$ and $u \neq 0$, as well as utilising the fact that ∇u is piecewise constant on Ω for a piecewise linear basis, the theory in [144, 156] is directly applicable to the linear system of equations

$$F_{u_j}(\varphi_i, \delta) = -F(u_j)(\varphi_i), \quad i = 1, \ldots, N_j, \tag{7.66}$$

on grid Ω_j with N_j non-Dirichlet nodes. Note that the coefficient function for the linear

problem (7.66) is not defined solely by α, but that it is influenced by the approximation u to the exact solution u^*. The coefficient is given by $\alpha |\nabla u|^2$ and highlights that the convergence of the iteration must consider a discontinuous coefficient function for a piecewise linear finite element discretisation. This leads to some interesting convergence behaviour of the Newton iteration.

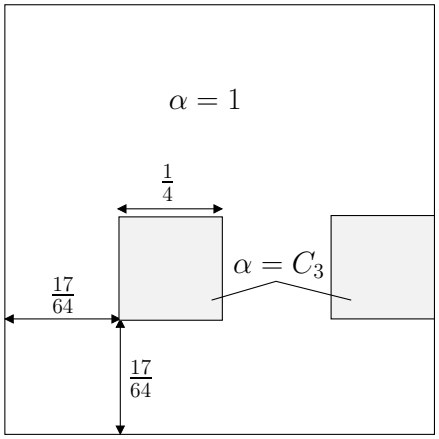
In the following it is assumed that the Newton linearisation close to the exact solution approximates closely the action of the nonlinear operator. This means that the convergence behaviour of a Newton iteration will be determined by the convergence of the linear inner iteration. Further from the exact solution it is not possible to say what effect the coefficient function will have on the convergence of a Newton method, so we concentrate on the case that an initial approximation is close to the exact solution.

In order to obtain an initial approximation close to the exact solution for the problems presented, an iteration is performed until convergence on the finest grid level. Some continuation or damping strategies may need to be employed to achieve this. An iteration is assumed to have converged when the nonlinear residual norm is not reduced by a Newton iteration. The solution on each level is then scaled by a factor to move it away from the exact solution. When results are presented this scaling factor is given.

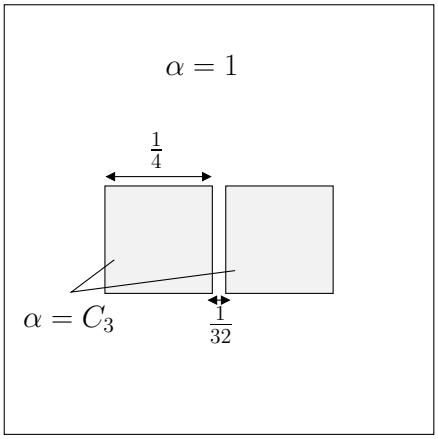
Results presented in this section are for the case that the function α is not resolved on some of the coarser grids, which should not affect the convergence of the linear multigrid iteration, or therefore the Newton iteration. As mentioned in Subsection 5.1.3, for multigrid to converge independent of the size of jumps in the coefficient, the coefficient function $\alpha |\nabla u|^2$ must satisfy a quasi-monotonicity (see Definition 5.1) property on all meshes.

Figure 7.14 shows the distribution of the coefficient on the unit-square domain for the results shown in Table 7.7. The scalar C_3 is set such that α alone would not satisfy some quasi-monotonicity conditions. Without *a priori* knowledge of the solution the exact effect on the convergence of the inner iteration is difficult to predict. Observed results are justified with an *a posteriori* investigation of the solution.

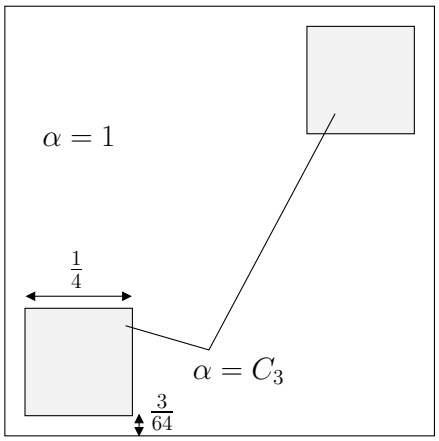
Results presented in Table 7.7 are for a 512^2 grid with coarsest grid 16^2. Note the inclusion of results for a preconditioned GMRES algorithm (see Subsection 3.3.3). This has not been used so far, as the convergence of this method closely matches that of the multigrid iteration in the constant coefficient case. However, for this problem a clear improvement over the use of pure multigrid is observed in the results, which are included to demonstrate that a simple change in the Newton iteration can give much improved convergence behaviour. No corresponding improvements are known for the FAS iteration.



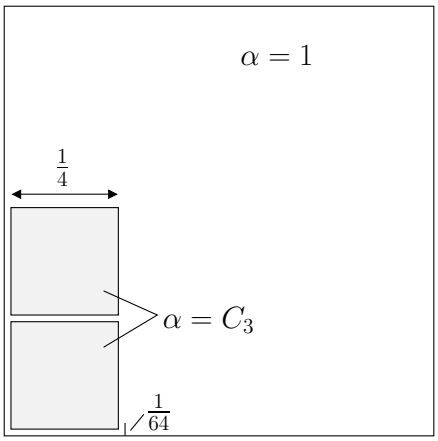
(a) Quasi-monotone and Γ-quasi-monotone on all grids finer than 16^2.



(b) Γ-quasi-monotone but not quasi-monotone on grids coarser than 128^2.



(c) Quasi-monotone but not Γ-quasi-monotone on grids coarser than 128^2.



(d) Neither quasi- nor Γ-quasi-monotone on grids coarser than 256^2.

Figure 7.14: Distributions of α satisfying, or not, quasi-monotonicity conditions.

	Quasi- and Γ-Quasi-Monotone			Only Γ-Quasi-Monotone		
C_3	MG	PGMRES	FAS	MG	PGMRES	FAS
10^1	15	12	27	15	12	28
10^2	15	12	30	21	15	44
10^3	18	12	84	27	18	>100
10^4	18	15	>100	45	21	>100
10^5	18	15	>100	60	24	>100

	Only Quasi-Monotone			Neither Quasi- nor Γ-quasi-monotone		
C_3	MG	PGMRES	FAS	MG	PGMRES	FAS
10^1	15	12	27	12	12	26
10^2	15	12	82	21	15	82
10^3	24	12	>100	36	18	>100
10^4	48	15	>100	78	21	>100
10^5	102	18	>100	>150	33	>100

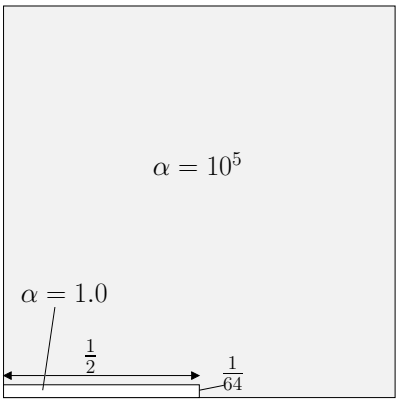
Table 7.7: Number of V-cycles / preconditioned GMRES iterations required to reduce the residual by a factor 1e-7 from initial approximation $u^{(0)} = 0.9u^*$ for varying C_3 on a 512^2 mesh.

For the distribution of α shown in Figure 7.14a the convergence of a linear multigrid iteration is expected to be independent of the size of the jump in the coefficient. This behaviour also holds true for coefficient $\alpha \left|\nabla u\right|^2$. The convergence is independent of the size of the jump in the coefficients, but the convergence of the Newton iteration is not independent of the mesh spacing for large values of C_3 (see Figure 7.16). This is investigated in more detail after a discussion of the convergence results for the configurations of α shown in Figures 7.14b to 7.14d. If α were to determine the coefficient function it would be expected that the convergence would deteriorate as the size of the jump in coefficient increases, as predicted in Theorem 5.1. The rate of deterioration is not known. A dependence on the size of the jump in the coefficients is clear also in the case of the Newton iteration.

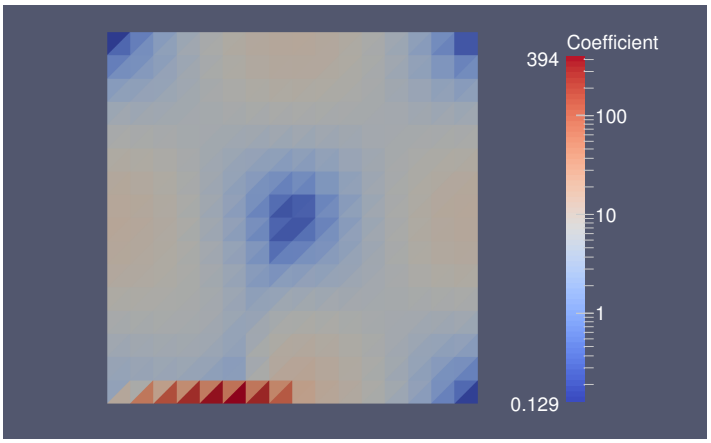
For all of the configurations shown in Figure 7.14 the FAS iteration does not scale well with the size of the jumps in coefficient. The convergence is dependent on the size of the jump of the coefficient even in the case that quasi-monotonicity conditions hold, and the deterioration of the convergence is much more rapid than that of the approximate Newton iterations. The results clearly show that the Newton methods are better to use in this situation. A preconditioned GMRES iteration is also superior to just a multigrid algorithm. This is due to the spectral properties of the linear operator. The regions of high coefficient add large eigenvalues to the operator (see [156]), which increases the size of the condition number of the linear operator. However, a GMRES iteration is much less

affected by individual large (or small) eigenvalues and convergence requires the spectrum to be grouped into bounded intervals (see [153, §6.11]).

Results in Table 7.7 suggest that the convergence of a Newton-MG iteration for highly varying coefficients can be predicted by the distribution of the function α. However, this is true only for the examples shown. In general the product $\alpha \left|\nabla u\right|^2$ gives the coefficient function, and good convergence can be observed in the nonlinear setting where the distribution of α on the domain does not satisfy quasi-monotonicity conditions, an example of which is shown in Figure 7.15. The number of V-cycles required for convergence of



(a) Distribution of α over Ω is not Γ-quasi-monotone.



(b) Distribution of coefficient $\alpha \left|\nabla u\right|^2$ is type 0 quasi-monotone even on a 16^2 grid.

Figure 7.15: Distribution of α and coefficient $\alpha \left|\nabla u\right|^2$. Due to the shape of the solution the coefficient is quasi-monotone, even when α is not.

Newton-MG and FAS are shown in Table 7.8. FAS does not converge well again, as the coefficient is not resolved on the coarsest grid. On the other hand Newton-MG converges very well, and is independent of the size of the jumps in the coefficient. Although not presented, results are very similar when using a multigrid preconditioned GMRES iteration for the linear solve. As predicted in Theorem 5.1 the convergence depends logarithmically on the size of the mesh around regions where the coefficient is type 0 quasi-monotone. Table 7.8b shows that fewer iterations are required as the coarsest mesh is refined, and when the coefficient is resolved on all grid levels the convergence improves drastically. However, the computational effort starts to be dominated by coarse grid solves when the grid is uniformly refined, but the same increase in performance should be observed for a locally refined mesh [156].

The results presented so far suggest that the convergence of the Newton iteration may be entirely defined in terms of how the linear iteration converges, which is not true. For the distribution of α in the previous example (see Figure 7.15a) the solution does not

Grid Size	Newton-MG	FAS
128^2	27	>100
256^2	27	>100
512^2	27	>100
1024^2	27	>100
2048^2	27	>100

(a) Number of V-cycles required for convergence using coarsest grid 16^2.

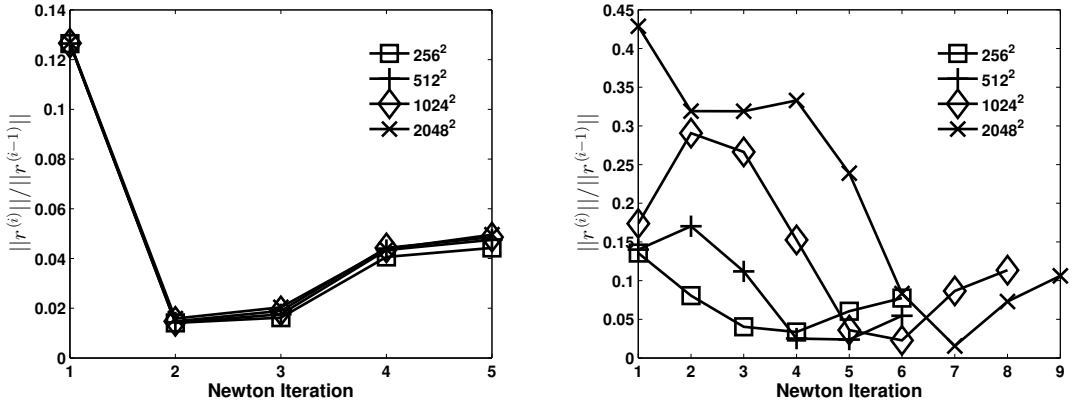
	Grid Size			
Coarsest Grid	128^2	256^2	512^2	1024^2
8^2	30	30	30	30
16^2	27	27	27	27
32^2	24	24	24	24
64^2	12	15	15	15

(b) Number of V-cycles required for convergence for Newton-MG for varying coarsest grids. This displays dependence on the grid spacing for type 0 quasi-monotone coefficient, as predicted in Theorem 5.1.

Table 7.8: Number of V-cycles required for convergence for Newton-MG and FAS for the distribution of α shown in Figure 7.15a on various grids.

depend on the value of α taken on the bulk of the domain. Hence, Assumptions 7.1 are satisfied. In particular, $\|\nabla u\|_\infty$ is uniformly bounded independent of the size of the jump in the coefficient.

For a distribution of α where a region of high value is in the interior of the domain, as in Figure 7.14a, the solution does depend on the size of the jump in α. The solution grows as the size of jump in α grows. Therefore $\|\nabla u\|_\infty$ is not bounded independent of α and Assumptions 7.1 are not satisfied. That these assumptions are necessary for mesh independent convergence is demonstrated in Figure 7.16. At relatively low C_3 the conver-



(a) $C_3 = 10^2$ (b) $C_3 = 10^5$

Figure 7.16: Convergence histories for varying mesh sizes for $C_3 = 10^2$ and $C_3 = 10^5$ for distribution of α shown in Figure 7.14a. Convergence is not independent of mesh spacing, as Assumptions 7.1 are not satisfied.

gence is independent of the mesh. However, for the larger value of C_3 the convergence is clearly dependent on the mesh.

The results in this section show that the convergence of a nonlinear iteration depends acutely on the value of the exact solution to the nonlinear problem. In this section varying convergence behaviours have been observed for the same nonlinear operator. This highlights a difficulty in the convergence analysis of nonlinear operators, as it will be difficult to predict convergence in general for a specific nonlinear problem without prior knowledge of the exact solution.

7.8 Summary

In this chapter results have been presented to demonstrate that, for a non-time-dependent nonlinear elliptic operator with nonlinearity in the diffusion coefficient, a Newton-MG iteration is superior to an FAS iteration in terms of execution time and robustness. Newton's method is much more robust with respect to a poor initial approximation, and is less sensitive to changes in coefficient on the domain. As well as this there exist a number of well researched methods for the stabilisation of Newton methods, whereas the stabilisation of an FAS iteration is difficult. It is also not necessary for the solution to be resolved on coarse grid levels for Newton-MG to converge independent of mesh parameters. Discussion has been included to demonstrate situations in which a Newton iteration may seem to be performing poorly, but simple algorithmic corrections restore expected fast convergence of the method.

Although only a single nonlinear operator was considered the results clearly show the significant effect that the exact solution to the problem has on the convergence of a method. Therefore, a convergence theory for a nonlinear iterative method must include *a priori* assumptions on the solution in order to be able to predict convergence behaviour. Results also demonstrate that the linear iteration may have highly varying coefficients, depending on the shape of the exact solution.

In the next chapter it is demonstrated that Newton-MG is also a better iteration when applied to time-dependent problems, where the linearisation is not symmetric positive definite. The discussion in this chapter helps to explain the observed convergence behaviour also for these more challenging problems.

Chapter 8

Application to Time-Dependent Problems

This chapter presents the comparison of Newton-MG and FAS iterations applied to time-dependent problems with physical applications. Section 8.1 deals with the porous medium equation (PME), and Sections 8.2 and 8.3 give different formulations of the Richards equation. As in the previous section experiments are performed demonstrating the computational efficiency and robustness of the algorithms. Robustness is considered in terms of a time-step size, and results are explained using the discussion from the previous chapter and the theory introduced in Chapters 4 and 5. All equations are solved in a two-dimensional setting. In the previous chapter a nonlinear iteration was considered to have completed successfully if a reduction of 1e-7 in the nonlinear residual was obtained. For the time-dependent problems we consider that it is sufficient to reduce the residual by a factor 1e-3 at each time step, as this is more computationally efficient. This stopping criterion is also used in practice (*e.g.* [154]) and is valid under the assumption that the initial residual at each time step will be small for a sufficiently small time step parameter.

8.1 Porous Medium Equation

In this section the solution of the porous medium equation (PME) (see [97, 131]) is considered. The problem to solve is given by

$$\begin{aligned} \frac{\partial}{\partial t}u &= \nabla \cdot (u^m \nabla u), \quad \vec{x} \in \Omega \\ u &= 0, \quad \vec{x} \in \partial\Omega, \\ u &= u^{(0)}, \quad t = t_0, \end{aligned} \tag{8.1}$$

on domain $\Omega = (-1, 1)^2$ for $(m > 0) \in \mathbb{N}$. This equation can be used to model various diffusion processes, such as the diffusion of gases through porous media [55] or the intrusion of pollutants in monuments [63, 158]. A radially symmetric, exact solution for (8.1) is known due to Barenblatt [11], given by

$$\begin{aligned} u(x,t) &= \frac{1}{\mu^2}\left[\max\left\{1 - \left(\frac{r}{r_0\mu}\right)^2, 0\right\}\right]^{\frac{1}{m}}, \\ r = \sqrt{x_1^2 + x_2^2}, \quad \mu &= \left(\frac{t}{t_0}\right)^{\frac{1}{2(1+m)}}, \quad t_0 = \frac{r_0^2 m}{4(1+m)} \end{aligned} \tag{8.2}$$

in two dimensions. Here x_i, $i = 1, 2$ are the component directions in two-dimensions, t is time and r is a radius from the origin. The solution is non-negative, and has compact support, which grows as time progresses. This solution is only valid for as long as the support is contained in the domain Ω. Here we have that $r < 1$, as $\Omega = (-1, 1)^2$, hence using the definitions in (8.2) requires

$$t < t_0 \left(\frac{1}{r_0}\right)^{2(1+m)},$$

and $r_0 < 1$. For brevity results are only presented for $m = 2$ in a two-dimensional setting. The case $m = 2$ is challenging because the gradient of the solution at the boundary of its support is unbounded (in theory). Therefore, the gradient near this boundary will increase as a mesh is refined and it cannot be expected that the convergence of a nonlinear iteration will be independent of the grid spacing. However, it is shown in [63, 158] that if u satisfies a local Lipschitz condition the convergence deteriorates in proportion to the grid spacing for the porous medium equation solved using a finite difference scheme. There, taking the time-step proportional to the grid spacing gives an optimal algorithm, in terms of the

running time. The exact solution u^* given in (8.2) is clearly not Lipschitz continuous in the continuous case, but results demonstrate the same optimal convergence behaviour (see Table 8.4 and Figure 8.1) for the discrete problem. Before giving empirical results in Subsections 8.1.2 and 8.1.3, the discretisation of the problem is outlined in the following subsection.

8.1.1 Discretisation and Weak Formulation

This subsection outlines the temporal and spatial discretisation of (8.1). As in Section 2.6, let δt be a constant time-step parameter such that $t_{k+1} - t_k = \delta t$, and let $t_k = t_0 + k\delta t$ for some given initial time t_0. Let $u^{(k)} = u(\,\cdot\,, t_k)$ denote the solution u at time t_k. The results presented are for the case that (8.1) is discretised in time using the Crank-Nicolson method (see Section 2.6) to give the time-discrete problem

$$u^{(k+1)} - \frac{\delta t}{2}\nabla \cdot \left[\left(u^{(k+1)}\right)^2 \nabla u^{(k+1)}\right] = u^{(k)} + \frac{\delta t}{2}\nabla \cdot \left[\left(u^{(k)}\right)^m \nabla u^{(k)}\right]. \tag{8.3}$$

Assume that the solution exists, is unique, and that $u^{(k)} \in W^{1,p}$, for some $p > 2$ at all time steps k. To discretise in space let a hierarchy of grids be defined as in (7.2). With each grid Ω_j, $j = 1, \ldots, J$, associate the finite element space $\mathcal{W}_j \subset W^{1,p}$, the basis of which is the set of piecewise linear nodal basis functions of $S_0(\Omega_j)$ (see Equation (2.15)). A basis function is denoted by φ_i, $i = 1, \ldots, N_j$ for N_j the number of non-Dirichlet boundary nodes on Ω_j. Taking test functions $\varphi_i \in \mathcal{W}_j$ and integrating (8.3) over Ω gives the weak system of equations

$$\int_\Omega u^{(k+1)}\varphi_i \,\mathrm{d}\vec{x} + \frac{\delta t}{2}\int_\Omega \left(u^{(k+1)}\right)^2 \nabla u^{(k+1)}\nabla\varphi_i \,\mathrm{d}\vec{x} = f(u^{(k)}, \varphi_i), \quad i = 1, \ldots, N_j, \tag{8.4}$$

where

$$f(u, v) = \int_\Omega uv \,\mathrm{d}\vec{x} - \frac{\delta t}{2}\int_\Omega u^2 \nabla u \nabla v \,\mathrm{d}\vec{x}.$$

Defining

$$F(u)(v) \equiv \int_\Omega uv \,\mathrm{d}\vec{x} + \frac{\delta t}{2}\int_\Omega u^2 \nabla u \nabla v \,\mathrm{d}\vec{x}, \tag{8.5}$$

it is possible to write (8.4) as

$$F(u^{(k+1)})(\varphi_i) = f(u^{(k)}, \varphi_i), \quad i = 1, \ldots, N_j. \tag{8.6}$$

The Jacobian matrix is formed using the derivative of (8.5), given by

$$F_u(w, v) = \int_\Omega wv \, \mathrm{d}\vec{x} + \frac{\delta t}{2} \left[\int_\Omega 2uw\nabla u \nabla v \, \mathrm{d}\vec{x} + \int_\Omega u^2 \nabla w \nabla v \, \mathrm{d}\vec{x} \right], \tag{8.7}$$

which may be split into a symmetric and non-symmetric part F_u^{s} and F_u^{ns}, respectively. These are given by

$$F_u^{\mathrm{s}}(w, v) = \int_\Omega wv \, \mathrm{d}\vec{x} + \frac{\delta t}{2} \int_\Omega u^2 \nabla w \nabla v \, \mathrm{d}\vec{x}, \tag{8.8}$$

$$F_u^{\mathrm{ns}}(w, v) = \frac{\delta t}{2} \int_\Omega 2uw\nabla u \nabla v \, \mathrm{d}\vec{x}. \tag{8.9}$$

Let

$$r^{(k+1)} = \left[F(u^{(k+1)})(\varphi_i) - f(u^{(k)}, \varphi_i) \right]_{i=1}^{N_j} \tag{8.10}$$

be the vector of residual values of problem (8.6), and let

$$F_{(k+1)} = [F_{u^{(k+1)}}(\varphi_l, \varphi_i)]_{i,l=1}^{N_j} \tag{8.11}$$

be the Jacobian matrix at time $k+1$. The linear system of equations to solve in a Newton iteration at time step k is given by

$$F_{(k+1)} \delta u = r^{(k+1)}, \tag{8.12}$$

for unknown correction δu. We assume that the inverse $F_{(k+1)}^{-1}$ exists at all times. Unlike in the previous chapter, the linearisation contains a non-symmetric term, which can cause problems if the non-symmetry becomes too large, as shown in the theory of Bramble, Kwak and Pasciak [26].

We consider if the non-symmetric part of the PME will become large in the case of

the self-similar solutions given in (8.2). Inspection of (8.2) shows that

$$\left\|u^{(k)}\right\|_\infty \leq 1$$

for all times. However, the gradient is unbounded at the boundary of the support of the function, such that as the mesh spacing h is decreased $\left|\nabla u^{(k)}\right| \to \infty$. It is safe to assume, therefore, that the non-symmetric term (8.9) will be significantly larger than the symmetric term (8.8) near the boundary of the support of $u^{(k)}$, unless it is scaled by a sufficiently small time-step parameter. Note that the symmetric part of the Jacobian in (8.8) is composed of a mass matrix term, and a term that is scaled by the time-step. Therefore the symmetric part of the Jacobian is bounded from below independent of the time-step, and the non-symmetric part of the Jacobian can be made arbitrarily small by making the time-step arbitrarily small. By choosing δt small enough it is possible to ensure the convergence of the linear multigrid iteration in each Newton iteration and results in Subsection 8.1.3 show that the Newton iteration and the FAS iteration are convergent at small time-steps. What is more interesting is the behaviour for larger time-steps, when the non-symmetric term starts to impact on the convergence of the methods.

We note that this problem contains a mass matrix term. This is the matrix M obtained by integrating the basis functions, *i.e.*

$$M = \left[\int_\Omega \varphi_i \varphi_l \,\mathrm{d}\vec{x}\right]_{i,l=1}^{N_j}. \tag{8.13}$$

In order to stabilise the methods the mass matrix term is 'lumped', which means that only diagonal terms exist, which are the sum of the remaining terms in the column of the matrix, *i.e.* the diagonal entries in the lumped mass matrix M^{L} are given by

$$M^{\mathrm{L}}_{i,i} = \sum_{l=1}^{N_j} M_{l,i}. \tag{8.14}$$

For simple equations this is known to preserve a maximum principle [147] and evidence suggests that the same is true for more complicated problems [21, 48]. In particular spurious oscillations and non-physical values in a solution are often avoided using a lumped mass matrix [53, pg. 383]. Results for the PME show that non-physical negative values of the solution are prevented using the lumping of the mass matrix term. For small values of the time-step the diagonal lumped mass matrix term dominates. It is therefore likely that

the nonlinear Jacobi iteration will be a good *solver* as well as a good smoother at small time scales, and it is expected that the FAS iteration will be a much more competitive solution algorithm. Results in Subsection 8.1.3 support this claim.

The theory in [26] states that so long as a non-symmetric linear system is a 'compact' perturbation from a symmetric positive definite system that convergence of linear multigrid remains independent of mesh parameters (see Subsection 5.1.3). This property also holds for the nonlinear PME (see Subsection 8.1.3) and Richards equation (see Subsections 8.2.3 and 8.3.3). There exists no analysis to state why this should be the case. The results presented suggest that a good heuristic for predicting the convergence of the nonlinear Newton iteration is to use the linear theory to predict the convergence of the linear inner iteration. This relies on the assumption that the Newton iteration is convergent, and in particular that at the exact solution the action of the linearisation matches the action of the nonlinear operator.

Before moving on to a comparison of the convergence behaviours of the methods for the PME, results are presented in the next subsection to show that the theory from Chapter 6 can be used to gain an accurate bound on the running times of Newton-MG and FAS iterations for the PME. A discussion of how the results suggest the running time of the algorithms is likely to evolve for more complex nonlinear problems is also given. Results in Subsection 8.1.3 demonstrate the robustness of the methods with respect to changes in the time-step parameter.

8.1.2 V-Cycle Execution Time

For a time-dependent problem, such as the PME, which is discretised implicitly in time, the Newton-MG and FAS iterations are used to solve the system of nonlinear equations arising, which are given in (8.6) for the PME. In order to assess the execution time per V-cycle it is not necessary to consider the time-dependence of the problem, and we can instead consider the execution time of the algorithm for a steady state problem of the same form as (8.7). Therefore, we introduce

$$\tilde{F}(u)(v) = \int_\Omega uv \,\mathrm{d}\vec{x} + C\int_\Omega u^2 \nabla u \nabla v \,\mathrm{d}\vec{x} - \int_\Omega fv \,\mathrm{d}\vec{x}. \tag{8.15}$$

where f is some known function. The steady state problem for which we present timing results is given by

$$\tilde{F}(u)(v) = 0. \tag{8.16}$$

The timing results presented ignore the time required for the set-up of time-dependent parameters. Table 8.1 shows the actual *vs*. predicted timing results for the nonlinear iterations. Approximations for the running time are gained using the formulas derived in Equations (6.25) to (6.27), once the size of a work unit has been measured. A work unit is measured as $W_j \approx 0.028$, which was established through empirical experiment. This is considerably faster than the work unit for the p-Laplacian. Using the same estimate as in (6.24) for the execution time of the linear multigrid (*i.e.* $C^{(j)}_{\text{LMG}} = 3/2W_j$) will give an optimistic estimate, likely to be smaller than the recorded execution time. This is because, whilst the work unit grows, the amount of time required for the linear solve remains constant – assuming that the same discretisation scheme is used. We therefore expect to see under-approximation of the execution time of Newton-MG iterations. For such a small work unit it is also expected that operations such as grid transfers will have a small effect on the execution time. In the estimate of the FAS iteration the cost of grid transfers is ignored, so that it is expected that the execution times will also be under-predicted. The column $\tilde{C}_{\text{NMG}}$ gives the predicted time for a Newton-MG iteration in the case that the operator is re-discretised on each grid; $\hat{C}_{\text{NMG}}$ gives the running times of Newton-MG using Galerkin coarse grid operators; and C_{FAS} gives the execution times for the FAS iteration. The columns labelled 'Act.' give the measured running times for each iteration. Recall

	p/ν	$\tilde{C}_{\text{NMG}}$	Act.	$\hat{C}_{\text{NMG}}$	Act.	C_{FAS}	Act.
$\gamma = 1$	1	15.03	17.62	14.23	18.66	15.40	18.29
	2	18.33	24.84	17.60	24.63	27.13	30.86
	3	21.63	30.79	20.90	30.03	38.86	42.97
	4	24.93	34.83	24.20	34.92	50.60	53.93
$\gamma = 2$	1	16.68	19.80	15.95	21.16	23.10	40.18
	2	21.63	29.29	20.90	28.80	40.69	58.10
	3	27.34	35.95	25.85	35.17	58.30	75.80
	4	31.53	41.96	30.80	41.00	75.90	91.25

Table 8.1: Predicted and actual timing results required to perform 100 Newton / nonlinear V-cycles.

that in Chapter 6 the theory predicted that when 3 pre- and post-smoothing iterations are used in the FAS iteration and 3 linear MG iterations are performed per Newton iteration, that Newton-MG will be ≈ 5.4 times more efficient than FAS. From the results in

Table 8.1 we see that the computed ratio is

$$\frac{42.97}{100.0} \times \frac{300}{30.79} \approx 4.19, \tag{8.17}$$

which is as large an improvement as previously. The difference here is that the cost of the linear multigrid iteration is under-predicted. However, the speed up is still significant.

The results in Table 8.1 are for when the integrals over elements are calculated using an analytic formula. For a simple problem such as the porous medium equation it is possible to use an analytic formula, but for more complex problems it may be necessary to use a numerical quadrature on the elements. The effect of the use of numerical quadrature on the execution times of the algorithms is interesting, results for which are presented below. The quadrature for which results are presented in Table 8.2 is a Simpson's rule on a triangle given by

$$\int_T F(x)\,\mathrm{d}x \approx A_T \left(\sum_{i=1}^{3} \left[\frac{1}{20} F(x_i^{(T)}) + \frac{2}{15} F(m_i^{(T)}) \right] + \frac{9}{20} f(c^{(T)}) \right), \tag{8.18}$$

where $x_i^{(T)}$, $i = 1, 2, 3$, are the nodal values on element T; $m_i^{(T)}$, $i = 1, 2, 3$, are the mid-points of the edges; and $c^{(T)}$ is the centroid of the element. This formula is not the most sophisticated, and other formulae may give more accurate results, see [54]. We are only interested in the effect the use of quadrature has on the execution times of the algorithms, so that accuracy is not of much concern.

Table 8.2 shows the actual *vs*. predicted timing results using the above Simpson's rule to approximate integrals over an element. Advantage is taken of the fact that basis functions are zero on some of the quadrature points to make the implementation more efficient. It can be seen in Table 8.2 that the FAS execution times are under-predicted to a larger extent in this case, whereas the estimated execution times for Newton-MG are much more accurate. This can be explained by considering the results in Table 8.3, which show the predicted *vs*. the recorded execution times of the individual components used in the FAS and Newton-MG iterations.

The timings of the algorithm components are shown when using the analytic formula and numeric integration to evaluate integrals over an element. Columns with heading 'Predict.' show the theoretically predicted running times, where the theory in Chapter 6 is used. The other columns show the recorded execution times. The calculation of the coarse grid right-hand side (which involves a calculation of the residual on the current grid and the calculation of the residual on the coarse grid) is accurately predicted, since this is di-

	p/ν	$\tilde{C}_{\text{NMG}}$	Act.	$\hat{C}_{\text{NMG}}$	Act.	C_{FAS}	Act.
$\gamma = 1$	1	19.13	19.23	18.20	19.23	19.60	25.90
	2	23.33	24.91	22.4	24.05	34.53	43.49
	3	27.53	28.82	26.60	28.35	49.46	61.15
	4	31.73	32.65	30.80	31.45	64.40	78.19
$\gamma = 2$	1	21.23	20.04	20.30	21.76	29.40	57.03
	2	27.53	26.49	26.60	28.96	51.80	82.93
	3	33.83	32.46	32.90	35.78	74.20	107.08
	4	40.13	39.90	39.20	40.22	96.60	130.96

Table 8.2: Predicted and actual timing results required to perform 100 Newton / nonlinear V-cycles when using quadrature to calculate integrals.

rectly related to the calculation of the residual. There are more significant deviations from the expected value for the calculations involving all or part of the Jacobian matrix. For these calculations the timings are over-predicted. This can often be expected in practice, as in an implementation advantage can be taken of re-using previously calculated values. It can be seen that better advantage is taken of this when calculating the diagonals of the Jacobian compared to the entire Jacobian, when the analytic integrals are used, and that the converse is true when using quadrature.

This behaviour can be explained by considering the calculations performed on each element as part of the quadrature. From (8.18) it can be seen that function evaluations are required at several points on an element. These function evaluations are performed at all quadrature points when calculating the diagonals of the Jacobian matrix. The same values may be re-used when calculating off-diagonal entries in the Jacobian matrix. Therefore, a relative saving is made when calculating the entire Jacobian matrix as previously calculated values are re-used more often.

Component Calculation	Analytic	Predict.	Numerical	Predict.
Residual	0.022	0.022	0.028	0.028
Residual + RHS	0.029	0.027	0.037	0.035
Residual + Jacobian	0.087	0.088	0.086	0.108
Residual + Jacobian Diagonals	0.041	0.044	0.055	0.056

Table 8.3: Actual *vs*. predicted execution times (s) for the individual components used in the solution of the PME. Times are given in seconds correct to 3 decimal places.

Extrapolating, using the above reasoning, we see that if a quadrature formula is used with more quadrature points per element, the relative speed of the calculation of the entire

Jacobian matrix will decrease compared to the calculation of the diagonals of the Jacobian matrix. Since for complex problems it is often necessary to perform a numerical quadrature, the results suggest that Newton-MG will give a better performance the more accurately quadrature approximates the integrals over each element. For more accurate quadrature see the summary of Gauss-Legendre quadrature on triangles of Cowper [54].

For the Simpson's rule in (8.18) the ratio of the execution times of a V-cycle in the FAS and Newton-MG iterations, as performed in (8.17) grows to approximately

$$\frac{61.15}{100} \times \frac{300}{28.82} \approx 6.36, \tag{8.19}$$

which is larger than the predicted value of 5.4 (see Equation (6.28)), and demonstrates the relative increase in efficiency of Newton-MG using numerical quadrature.

In this subsection we have shown that the framework for approximating the execution time of the algorithms, as given in Chapter 6, is sharp and useful also for the PME. The theory also helps to highlight that the relative efficiency of Newton-MG compared to FAS will increase as the amount of work per element is increased. The next subsection demonstrates the robustness of the methods, for the PME, with respect to a time-stepping parameter.

8.1.3 Robustness

In this subsection the robustness of Newton-MG and FAS for the PME with exact solution given in (8.2) is considered with respect to a time stepping parameter δt. The Crank-Nicolson scheme is used to discretise in time and the arising discrete system of equations to solve at each time step is given by (8.6). The initial approximation for the solution at time step $k+1$ is taken to be the computed solution at time step k. It was demonstrated in the previous chapter that the convergence of a nonlinear iteration is sensitive to an initial approximation. In particular, the convergence is better the closer the initial approximation is to the exact solution. For a theoretical justification of why this should be the case see Subsection 3.4.1. Therefore, convergence should be best for a small time step. For a time step too large the convergence may deteriorate or divergence of a nonlinear method may occur due to a poor initial guess.

For an increasing time step it was shown in Subsection 8.1.1 that the influence of a non-symmetric term in the linearisation (8.7) begins to grow. A standard linear multigrid implementation, as considered in this thesis, does not perform well for a problem with a 'large' non-symmetric part [26]. Hence, not only is it possible that the initial approxima-

tion to the nonlinear problem becomes poor with increasing time step, but the linearisation becomes more challenging for a standard multigrid iteration to solve. Based on heuristics from the linear multigrid convergence theory it is therefore expected that the convergence of a Newton-MG iteration will deteriorate with increasing time step either because an initial approximation becomes poor, or because the inner linear multigrid iteration is no longer convergent. We investigate which one of these is the case in this subsection. There is no theory to suggest how an FAS iteration will perform, and we are required to draw conclusions from empirical results.

Often, when an implicit-in-time discretisation is used, taking a large time-step is desirable, as very small scale changes are not of interest. Therefore, a solution method may be considered 'good' if a large time step is permissible. Table 8.4 shows the maximum allowable time step such that an FAS or approximate Newton iteration reduces the residual norm by a factor 1e-3 at each time step, over the first 5 time steps. The solution is given by (8.2) and the initial condition is set with $r_0 = 0.3$ (see (8.2)). In theory the gradient of the solution is infinite at the boundary of the support of the function. In practice, because the function is represented as a piecewise linear continuous function, there is no infinite gradient. Numerical diffusion also serves to smooth out the gradient at the boundary such that the most extreme gradient is found in the initial condition, which represents the most difficult situation for the Newton-MG and FAS iterations to solve. Therefore, a maximum allowable time step can be found by considering whether the iterations converge for the first 5 time steps. If the iterations converge for these then they converge for the remaining time steps, if the time step is kept constant. This maximum allowable time step is shown in Table 8.4 for different grid sizes.

Results are also included for a Newton method where the linear system is solved using a multigrid-preconditioned GMRES iteration. Since it is known that the multigrid iteration is most suitable for symmetric problems the preconditioning is applied to the symmetric part of the Jacobian (8.8) only. The preconditioning procedure is also not a solution procedure, and is intended to make the system better conditioned. As such a single pre- and post-smooth are performed to make the iteration more efficient.

From Table 8.4 we see that a larger maximum time-step is permitted using a Newton-MG rather than an FAS iteration at all grid levels. A much larger time-step may be taken using a preconditioned GMRES iteration, and in fact the Newton-preconditioned GMRES iteration is convergent for even larger time-steps than are given in Table 8.4. The results show the maximum allowable time-step in which 20 or less nonlinear iterations are required to achieve the desired 1e-3 reduction in the residual.

The fact that the Newton-GMRES iteration converges when the Newton-MG does not

Grid Size	MG	FAS	GMRES
32×32	0.348	0.111	>10.0
64×64	0.0828	0.0560	6.74
128×128	0.0215	0.0158	0.544
256×256	0.00968	0.00702	0.138
512×512	0.00452	0.00311	0.0423
1024×1024	0.00193	0.00134	0.0145
2048×2048	0.000924	0.000702	0.00568

Table 8.4: Maximum allowable time-step (s) for FAS and inexact Newton iterations for the PME with exact solution given by (8.2) with $r_0 = 0.3$

demonstrates that the reason for divergence of the Newton-MG methods is the failure of the linear multigrid iteration to converge, rather than that the initial approximation is outside of a ball of guaranteed convergence. It is also clear that the FAS iteration is sensitive to non-symmetries in the linearisation, and to a larger (although marginally) extent than Newton-MG. For the approximate Newton iteration, using an appropriate linear solver gives a much more robust iteration. It is not known how to modify the FAS iteration in a simple manner to improve the convergence properties. The only obvious change is to take a different nonlinear smoother, but as previously discussed (see Subsection 6.1.1) this adds considerable computational cost to the solution procedure [167] and is not considered here.

In the introduction to this chapter it was mentioned that [63, 158] show that for a PME discretised using finite differences, where the solution is locally Lipschitz continuous, that the convergence for a Newton-preconditioned GMRES iteration deteriorates linearly with the mesh spacing. Using a finite element discretisation and a non-Lipschitz continuous solution we see the same asymptotic deterioration in the convergence in Table 8.4 for all solution methods. Figure 8.1 shows that the execution times of the iterations scales linearly with the number of unknowns when the time-step is taken proportional to the grid spacing. Note that this is a reasonable scaling when using second order schemes in both space and time.

The execution time of the Newton-MG iteration is slightly quicker than the execution time of the FAS. The speed-up is not so pronounced for this problem as the number of linear V-cycles required are much higher than the number of FAS V-cycles, as shown in Figure 8.2. The overall number of V-cycles performed for Newton-MG is approximately 5.3 times as many as for FAS. Using the measured ratio of execution times per V-cycle iteration in (8.19) the FAS iteration should take approximately 1.16 times longer to run than Newton-MG, which is what is observed. Interestingly, the multigrid-preconditioned

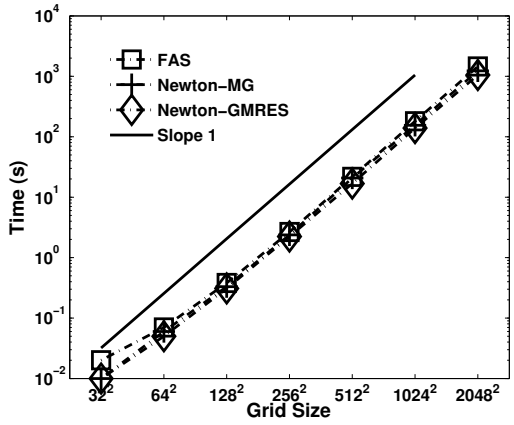


Figure 8.1: Execution time for FAS and Newton-MG for the PME with time-step $\delta t = h$ for grid spacing h.

GMRES iteration is the most efficient in this case. This is due to the fact that a single pre- and post-smooth are sufficient in the preconditioner to gain a convergent iteration. The fact that this method is also the most robust demonstrates that the linear solver may be changed easily without large penalties in terms of execution time.

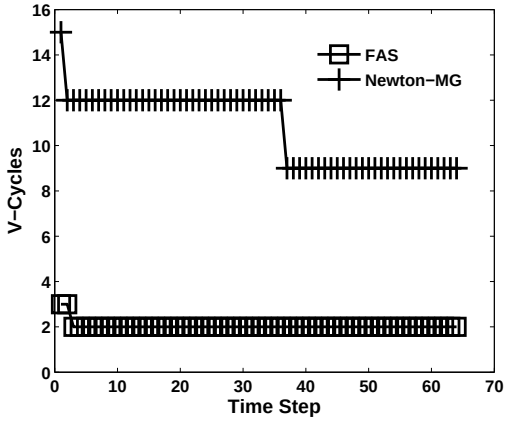


Figure 8.2: Number of linear / nonlinear V-cycles required per time-step for the PME with solution (8.2) and $r_0 = 0.3$ on a 2048^2 grid.

It was noted at the start of this subsection that for small time steps the nonlinear system of equations will be dominated by the diagonal lumped mass matrix. Hence, off-diagonal entries in the Jacobian have a small effect on the convergence of the iteration. The nonlinear Jacobi iteration uses information from the diagonal of the Jacobian matrix, which in this case will closely match the mass matrix term. This explains the increase in performance of the FAS iteration, relative to Newton-MG, in comparison to the 4-Laplacian considered in the previous chapter. This increase in performance comes simply

because performing a whole Newton step is not required for the almost diagonal system of equations to solve. However, a large advantage of a Newton method over FAS is observed when larger time steps are taken in conjunction with a Newton-GMRES iteration, as demonstrated in Table 8.4.

Results in this subsection have shown that for a time dependent problem a Newton-MG still seems to be a better iteration than FAS. Robustness is greater for a Newton-MG iteration, and a much more robust algorithm can be gained by replacing the inner iteration of the Newton method by a multigrid-preconditioned GMRES iteration. For small time steps there is little to choose between the two methods, however. In the next section we show that the same advantages are observed for a more complicated problem - the Richards equation - which models the flow of an incompressible fluid through a porous medium.

8.2 Richards Equation as a Single Equation

This section evaluates the application of nonlinear multigrid methods to different formulations of the *Richards equation*. The section is ordered as follows. We start with a brief introduction to the Richards equation. Section 8.2 considers the discretisation of the Richards equation posed as a single nonlinear diffusion equation. Estimates of the execution time and robustness with respect to a time-step parameter are considered, as in the previous section for the PME. Section 8.3 considers the discretisation of an equivalent saddle-point problem involving mixed finite elements. The results demonstrate the large advantage that can be expected in terms of execution time by performing a Newton-MG rather than an FAS iteration for more complicated nonlinear operators.

The Richards equation models the flow of water through an unsaturated porous medium (typically soil) [93, §8]. It is a special case of a two-phase flow problem in which one phase is air and the other an incompressible fluid (often taken to be water). The phases are assumed immiscible, and the air phase is assumed to be under constant pressure and in constant contact with the surface of a porous medium [93, pg. 158]. Under these conditions the only unknown becomes the flux (flow per unit volume) of the incompressible fluid. The flux of the incompressible fluid phase can be modelled using a nonlinear Darcy's law [43, 93] as

$$\vec{q} = -K(\psi)\nabla(\psi + z), \tag{8.20}$$

where $\vec{q}$ is the flux, ψ a pressure (or suction) head, and z the z-coordinate in a two- or

three-dimensional space representing a gravitational force. Here z is assumed positive downward. In one dimension z is the component direction in which the flux is measured, and gravity is aligned with the flow. In two-dimensions the xz-plane is used. The function $K(\psi)$ is termed the hydraulic conductivity, and represents the rate at which water may flow through a soil given the pressure head. The pressure head ψ is negative in an unsaturated regime, and a soil is saturated when $\psi \geq 0$. In order to form Richards equation the following continuity equation is required:

$$\frac{\partial}{\partial t}\theta(\psi) = -\nabla \cdot \vec{q} \tag{8.21}$$

where θ is termed the *volumetric wetness* and represents the volume fraction of water in the soil. The continuity equation is based on conservation of mass and states that the change in volume of water in a soil matrix is equal to the net flux into and out of the soil [93, pg. 155-7].

Combining the nonlinear Darcy law (8.20) with the continuity equation (8.21) gives Richards equation

$$\frac{\partial}{\partial t}\theta(\psi) - \nabla \cdot (K(\psi)\nabla(\psi + z)) = 0. \tag{8.22}$$

Note that in general $K(\psi)$ is a tensor representing the hydraulic conductivity in the component directions [113]. In the current study we consider an isotropic flow in all component directions such that $K(\psi)$ is represented as a scalar function.

In order to solve the Richards equation for an unknown pressure field ψ, explicit forms of the functions θ and K are required. There are several models available for the estimation of these functions with respect to a pressure head, see [43, 96, 126, 128, 179]. The difficulty in accurately measuring different soil properties has meant that early models have not been improved upon, to the best of our knowledge. One of the early models combining work by van Genuchten and Mualem [179] has remained popular in recent years (see for example [67, 102, 184, 188]). Most works use a modified version of the van Genuchten-Mualem model, given in [185, 186], which introduces a modification in order to simplify the numerical solution of Richards equation. It is this model which we use in this section, and introduce below.

Let $\psi_s < 0$ be some pressure above which a soil is considered saturated. This is termed the bubbling or air entry pressure [186]. Let θ_s be the volumetric wetness at saturation, and let $\theta_r < \theta_s$ be the residual volumetric wetness, which is the amount of water retained

in a 'dry' soil. The *effective saturation* S_e is then defined as

$$S_e(\theta) = \frac{\theta - \theta_r}{\theta_s - \theta_r}, \tag{8.23}$$

which gives a volume fraction of the amount of water in a soil *vs*. the water storage capacity of the soil.

As per the modification in [185] the parameter $\theta_m \geq \theta_s$ is introduced to avoid infinite gradients of the function $\theta(\psi)$ near saturation. The volumetric wetness is then modelled by [186]

$$\theta(\psi) = \begin{cases} \theta_r + \frac{\theta_m - \theta_r}{[1+(-\alpha\psi)^n]^m}, & \psi < \psi_s \\ \theta_s, & \psi \geq \psi_s. \end{cases} \tag{8.24}$$

for $n > 1$ and $m = 1 - 1/n$ [179]. Let

$$S_e = S_e(\theta(\psi)).$$

The hydraulic conductivity is described as a product

$$K(\psi) = K_s K_r(S_e) \tag{8.25}$$

for K_s the conductivity at saturation and $K_r \in (0, 1]$ the relative hydraulic conductivity given by

$$K_r(S_e) = S_e^{1/2} \left[\frac{1 - F(S_e)}{1 - F(1)} \right]^2, \tag{8.26}$$

where

$$F(S_e) = (1 - S_e^{*1/m})^m, \tag{8.27}$$

and

$$S_e^* = \frac{\theta_s - \theta_r}{\theta_m - \theta_r} S_e. \tag{8.28}$$

The parameters n, α, θ_r, θ_s and K_s are material properties that depend on the type of soil being modelled. For a list of approximate parameterisations of different soils see [186], from which the data in this thesis are taken.

The discussion so far gives enough information to be able to define a problem to solve using the Richards equation in the continuous case. In the following subsections results and discussion are presented for two discretisations of the Richards equation. Subsection 8.2.1 gives the discretisation of Richards equation using a continuous piecewise linear finite element discretisation, for which results are presented in Subsections 8.2.2 and 8.2.3. An alternative formulation and discretisation of Richards equation using mixed finite elements is presented in Subsection 8.3.1. The results presented demonstrate the superiority of Newton-MG over FAS also for this more complex nonlinear problem. Due to the complexity of the nonlinear problem considerations of software implementation also have an effect on the running time, and the effect of these are discussed.

8.2.1 Linear Finite Element Discretisation

In this section the discretisation of Richards equation (8.22) is given in the case that a continuous piecewise linear finite element discretisation is used. Although this is a low order method there has been research to show that using a higher order Lagrangian element can lead to spurious oscillations in a solution [101], especially at small time scales. It is therefore possible that the low order scheme will give the most accurate solution. It is usual for the discretisation to use a lumped mass matrix (see for example [13,48,101]), as it is found that spurious oscillations may arise in a solution if a consistent mass matrix is used. All results presented are for a lumped mass matrix. For simplicity the model is presented in two dimensions, but an extension to three dimensions follows a similar reasoning.

Define the computational domain as

$$\Omega \equiv (0, 100) \times (-100, 0).$$

Let $\Omega_j \subset \Omega$ be a grid with an associated simplicial triangulation $\mathcal{T}_j$ and continuous piecewise linear function space $\mathcal{V}_j \subset S_0(\Omega_j)$ (see (2.14)), the basis of which is the standard linear nodal basis (denoted by $\{\varphi_i\}_{i=1}^{N_j}$). As in previous chapters let N_j be the number of non-Dirichlet-boundary nodes on grid Ω_j. The unknown function to solve for in Richards

equation (8.22) is the pressure head $\psi = \psi(\vec{x}, t)$. Let δt be a fixed time step such that

$$t_k = t_0 + k\delta t,$$

where t_0 is some initial time. Then

$$\psi^{(k)} = \psi(\,\cdot\,, t_k). \tag{8.29}$$

Define

$$\theta^{(k)} \equiv \theta(\psi^{(k)}). \tag{8.30}$$

$K^{(k)}$ is defined similarly.

The problem which will be solved in this section is given by

$$\begin{gathered} \frac{\partial}{\partial t}\theta(\psi) - \nabla \cdot (K(\psi)\nabla(\psi + z)) = 0, \quad \vec{x} \in \Omega, \\ K(\psi)\nabla(\psi + z) \cdot \vec{\nu} = 0, \quad x = 0, 100, \quad \text{or} \quad z = -100, \\ \psi = 0, \quad z = 0, \\ \psi = \psi_0, \quad t = t_0, \end{gathered} \tag{8.31}$$

where $\vec{\nu}$ is the outward facing normal. This models the infiltration of water into a column of soil where impermeable boundaries are assumed on left, right and bottom boundaries. The boundary condition at the top sets the soil at saturation at the surface.

The first step in the discretisation is to discretise (8.31) in time. This is done using an implicit Euler scheme as follows:

$$\theta^{(k)} - \delta t \nabla \cdot \left(K^{(k)}\nabla(\psi^{(k)} + z)\right) = \theta^{(k-1)}. \tag{8.32}$$

To increase the efficiency of the method the nonlinear functions $\theta(\psi)$ and $K(\psi)$ are approximated using a continuous piecewise linear finite element basis as well as the func-

tion ψ as in [48], *i.e.*

$$\psi \approx \sum_{i=1}^{N_j} \psi(\vec{x}_i, t)\varphi_i = \sum_{i=1}^{N_j} \psi_i \varphi_i \tag{8.33}$$

$$\theta(\psi) \approx \sum_{i=1}^{N_j} \theta(\psi(\vec{x}_i, t))\varphi_i = \sum_{i=1}^{N_j} \theta_i \varphi_i \tag{8.34}$$

$$K(\psi) \approx \sum_{i=1}^{N_j} K(\psi(\vec{x}_i, t))\varphi_i = \sum_{i=1}^{N_j} K_i \varphi_i, \tag{8.35}$$

where $\vec{x}_i$ are nodal values on grid Ω_j. Taking test functions from the set of basis functions and integrating (8.32) gives the discrete system of equations to solve at each time step as

$$\overbrace{\int_\Omega \theta^{(k)}\varphi_i \,\mathrm{d}\vec{x} + \delta t \int_\Omega K^{(k)} \nabla(\psi^{(k)} + z)\nabla\varphi_i \,\mathrm{d}\vec{x}}^{\equiv F(\psi^{(k)})(\varphi_i)} = \overbrace{\int_\Omega \theta^{(k-1)}\varphi_i \,\mathrm{d}\vec{x}}^{\equiv f^{(k-1)}(\varphi_i)}, \quad i = 1, \ldots, N_j. \tag{8.36}$$

Note that the mass matrices arising in the discretisation of the integrals of $\theta^{(k)}$ are lumped. As outlined in Subsection 2.3.2, the integrals above can be evaluated as sums of integrals over elements $T \in \mathcal{T}_j$ on each grid. Let

$$F(\psi^{(k)}) = [F(\psi^{(k)})(\varphi_i)]_{i=1}^{N_j} \quad \text{and} \quad f^{(k-1)} = [f^{(k-1)}(\varphi_i)]_{i=1}^{N_j}.$$

The final stage of the discretisation requires the linearisation of (8.36). An entry in row i, column l of the Jacobian matrix is given by

$$F_{\psi^{(k)}}(\varphi_l, \varphi_i) = \int_\Omega D\theta(\psi^{(k)})\varphi_l\varphi_i \,\mathrm{d}\vec{x} + \delta t \left[\int_\Omega DK(\psi^{(k)})\varphi_l \nabla(\psi^{(k)} + z)\nabla\varphi_i \,\mathrm{d}\vec{x} + \int_\Omega K^{(k)}\nabla\varphi_l\nabla\varphi_i \,\mathrm{d}\vec{x}\right], \tag{8.37}$$

where $D\theta(\psi^{(k)})\varphi_l$ is the derivative of $\theta^{(k)}$ with respect to $\psi^{(k)}$ at φ_l, and similarly for $DK(\psi^{(k)})\varphi_l$. These derivatives are approximated using a forward divided difference formula (see Equation (6.6)). The Jacobian matrix is then formed as

$$F_{\psi^{(k)}} = \left[F_{\psi^{(k)}}(\varphi_l, \varphi_i)\right]_{i,l=1}^{N_j}. \tag{8.38}$$

Note that $F_{\psi^{(k)}}$ has symmetric part $F^{\mathrm{S}}_{\psi^{(k)}}$ and non-symmetric part $F^{\mathrm{NS}}_{\psi^{(k)}}$ with entries given by

$$F^{\mathrm{S}}_{\psi^{(k)}}(\varphi_l, \varphi_i) = \int_\Omega D\theta(\psi^{(k)})\varphi_l\varphi_i \,\mathrm{d}\vec{x} + \delta t \int_\Omega K^{(k)}\nabla\varphi_l\nabla\varphi_i \,\mathrm{d}\vec{x}, \tag{8.39}$$

$$F^{\mathrm{NS}}_{\psi^{(k)}}(\varphi_l, \varphi_i) = \delta t \int_\Omega DK(\psi^{(k)})\varphi_l\nabla(\psi^{(k)} + z)\nabla\varphi_i \,\mathrm{d}\vec{x}. \tag{8.40}$$

The corresponding matrices $F^{\mathrm{S}}_{\psi^{(k)}}$ and $F^{\mathrm{NS}}_{\psi^{(k)}}$ are defined similarly to $F_{\psi^{(k)}}$ in (8.38).

The Newton iteration for the Richards equation reads

$$\psi^{(k,p)} = \psi^{(k,p-1)} + \delta^{(p)}$$

where

$$F_{\psi^{(k,p-1)}}\delta^{(p)} = -(f^{(k-1)} - F(\psi^{(k,p-1)})). \tag{8.41}$$

Using the above linearisation gives a globally mass conservative scheme. If only the symmetric part of the Jacobian matrix is taken as the linearisation, *i.e.* instead of solving (8.41) the equation

$$F^{\mathrm{S}}_{\psi^{(k,p-1)}}\delta^{(p)} = -(f^{(k-1)} - F(\psi^{(k,p-1)}))$$

is solved for $\delta^{(p)}$, the scheme remains mass conservative (see Celia et al [48]). The use of this linearisation will be referred to as Celia's model. Using the symmetric part of the Jacobian means that the linear system of equations to solve is symmetric positive definite, which makes its solution appropriate for methods such as geometric multigrid and conjugate gradients [48, 101].

As discussed in Chapter 7 the Newton iteration converges rapidly only when an accurate approximation of the Jacobian matrix is taken. This implies that Newton-MG may not converge very quickly if $F^{\mathrm{S}}_{\psi^{(k,p-1)}}$ is taken as the linearisation. On the other hand, convergence of FAS was shown to be much less sensitive to inaccuracies in the Jacobian matrix, and FAS could converge quickly even if only the diagonals in the symmetric part of the Jacobian are used in a smoothing operator.

Results in Subsection 8.2.3 show that this behaviour is indeed observed. A Newton iteration converges much better when applied to the non-symmetric linearisation, so long

as a non-symmetric iterative method, such as a GMRES iteration, is applied to it. The FAS iteration converges better in the case that the symmetric approximation to the Jacobian is used. Before these results are presented, the execution time per V-cycle of the iterations are evaluated using the framework introduced in Chapter 6.

8.2.2 V-Cycle Execution Time

Results presented in this section show the predicted execution time of a V-cycle iteration *vs*. the measured execution time for Newton-MG and FAS on a 512^2 mesh. Before a prediction can be made a measure of the time required to calculate the nonlinear residual is required, which was empirically determined to be $W_{512} = 0.082$, where the subscript indicates that the mesh used is a 512^2 mesh.

As the time to calculate the residual is larger than for the model problems presented previously, it is expected that the approximation of the execution time of the linear multigrid iteration will be over-estimated. Hence, if the approximations for the execution times of the individual components of the algorithms are sharp, the predicted execution time should be larger than the observed time for a Newton-MG iteration. The predicted execution time of the FAS iteration should be accurate if the predicted execution times of the components of the algorithm are sharp.

Table 8.5 shows the predicted *vs*. actual execution times of V-cycles when Celia's model is used (*i.e*. the linearisation is approximated with the matrix F_ψ^{S} given in (8.39)). The linearisation in Celia's model has fewer terms than the actual Jacobian, which explains the over-estimation of the execution time of the Newton-MG iterations. For this more complex problem the advantage of using Galerkin coarse grid operators is apparent, with large savings in time ($\sim 15.83\%$ for the V-cycle with 1 pre- and post-smooth).

The results that are observed for the FAS iteration are more interesting. The theory does not predict the actual running time well at all, and the observed increase in execution time with an increase in the number of smooths performed is not uniform, as it is expected to be and as it was observed for previous problems. This suggests that the smoothing iteration does *not* have a linear order execution time. Theoretically, however, the smoothing operator is of linear complexity. Results for previous problems support this claim as well. The cause of the increase in the execution time in this case is a code specific issue. From the equations used in the Richards equation (see (8.23)-(8.28)) it can be seen that the calculations involve many floating point numbers being raised to the power of other floating point numbers. It is known in the programming community that the library libc implementation of the power function (especially on 64-bit systems) is not a constant

	p/ν	$\tilde{C}_{\text{NMG}}$	Act.	$\hat{C}_{\text{NMG}}$	Act.	C_{FAS}	Act.
$\gamma = 1$	1	56.03	41.38	53.3	34.83	57.4	74.16
	2	68.33	45.52	65.60	39.52	101.13	129.94
	3	80.63	50.18	77.90	44.65	144.87	188.07
	4	92.93	54.75	90.20	48.84	188.6	245.61
$\gamma = 2$	1	62.18	42.53	59.45	38.04	86.10	244.88
	2	80.63	49.08	77.90	44.63	151.70	271.99
	3	101.91	55.64	96.35	52.15	217.30	350.71
	4	117.53	62.29	114.80	55.43	282.90	435.16

Table 8.5: Predicted and actual timing results required to perform 100 Newton iterations when using Celia's model to obtain the linear operator to use in the inner iteration.

time operation [171]. Some inputs may require more time to execute than others, and large variations in execution time can occur for very similar numbers. Certain compilers are able to deal with this shortcoming better than others. The Intel C Compiler (icc) has been used here, which is a vast improvement on the GNU Compiler Collection (gcc) compiler. Another alternative would be to change the mathematics library to which the code is linked, as implementations differ from library to library. A 32-bit implementation may also be used.

Regardless of the implementation used, however, the 'pow' function remains expensive. This can be checked by executing the code and manually inspecting the stack (using a tool such as Cachegrind [178]) where it is found that a large amount of time is spent processing the 'pow' function. It is known in the computing community [171] that this function should be avoided where possible. It is difficult to see how the use of this would be avoided in the case that a floating point power is required. One way to avoid extensive use of the 'pow' function is to use a majority of linear calculations, as is done as part of the inner iteration of a Newton iteration. This is another advantage of Newton's method, as only a single nonlinear residual needs to be calculated per Newton iteration. Therefore the execution time of a Newton iteration is affected to much less an extent by the poor performance of the expensive nonlinear calculations. This shows that from a computational point of view nonlinear operations should be avoided in an efficient implementation. The extent to which the performance is affected for the FAS (in which the nonlinear operations are performed on every grid level) is shown by the fact that the ratio of the V-cycle execution time when performing 3 pre- and post-smooths and 3 inner iterations for the

Newton iteration is given by

$$\frac{188.07}{100} \times \frac{300}{50.18} \approx 11.24, \tag{8.42}$$

if the operator is re-discretised on each grid as part of a Newton-MG iteration, and

$$\frac{188.07}{100} \times \frac{300}{44.65} \approx 12.64 \tag{8.43}$$

if the Galerkin coarse grid operator is used. This demonstrates that for the more complicated nonlinear problem the Newton-MG iteration becomes significantly more competitive, as was claimed in the previous chapter. The measured ratios are much larger than the estimated 5.4 times speed up, but the framework outlined in Chapter 6 has been useful in helping to identify bottlenecks or speed-ups in performance encountered by the methods. This is done by breaking the timing of the methods up into manageable pieces which can be analysed and timed individually.

Results in the next subsection show that the convergence of the FAS iteration is satisfactory when using Celia's model. However, an improvement can be gained for the Newton iterations when the actual Jacobian is used. In this case a faster convergence per Newton iteration is gained for approximations close to the exact discrete solution, making the method more efficient overall despite the higher computational cost of calculating the full Jacobian matrix. Results are presented later when Newton-MG uses the full Jacobian matrix, so the timing results for the use of the full Jacobian are given in Table 8.6.

	p	$\tilde{C}_{\text{NMG}}$	Act.	$\hat{C}_{\text{NMG}}$	Act.
$\gamma = 1$	1	56.03	70.54	53.3	58.29
	2	68.33	75.02	65.60	63.12
	3	80.63	79.43	77.90	68.04
	4	92.93	86.61	90.20	73.07
$\gamma = 2$	1	62.18	73.80	59.45	60.36
	2	80.63	79.76	77.90	68.86
	3	101.91	86.38	96.35	74.03
	4	117.53	94.41	114.80	78.62

Table 8.6: Predicted and actual timing results required to perform 100 Newton iterations when using the actual Jacobian matrix in the inner iteration.

The advantage of using the Galerkin coarse grid operators is again clear. The ex-

ecution times are larger than for the Celia model presented in Table 8.5, as the actual Jacobian contains an extra term compared to the symmetric part of the matrix. However, the increase in time is unexpectedly large. Inspection of the form of the residual in (8.36), as well as comparing the linearisation (8.37) with that of the Celia model (8.39), show that the calculation of the term $DK(\psi^{(k)})$ is required in the full Jacobian. Using numeric integration, this requires extra function calculations of the hydraulic conductivity K (see (8.25)), which involves many powers of floating point values. It has been shown that the cost of performing this is not necessarily linear, so that the unexpected increase in execution time can be attributed to this.

The execution of a Newton method is still efficient compared to an FAS iteration when the more expensive Jacobian is used in the linearisation. Comparing the execution time of 3 pre- and post-smooths of the FAS with the Celia model to 3 inner iterations for Newton-MG gives the ratio

$$\frac{188.07}{100} \times \frac{300}{79.43} \approx 7.10 \tag{8.44}$$

when the Jacobian is re-discretised on each grid, and

$$\frac{188.07}{100} \times \frac{300}{68.04} \approx 8.29 \tag{8.45}$$

when the Galerkin coarse grid operator is used. This is again much larger than the predicted 5.4 increase from the theory in Chapter 6, and shows just how much an advantage computation of linear operations can have on the performance, as well as the improvement due to the linear solve becoming cheaper the more complex the nonlinear problem becomes.

In the next subsection the robustness of the most efficient of both methods is presented, so we concentrate on the use of the Galerkin coarse grid operators only. Results are presented in the next subsection to show the efficiency of the iterations, as well as investigating the robustness with respect to a time step parameter.

8.2.3 Robustness

In this subsection the robustness of the Newton-MG and FAS iterations is considered for the solution of the following Richards equation

$$\begin{aligned} \frac{\partial}{\partial t}\theta(\psi) - \nabla \cdot (K(\psi)\nabla(\psi + z)) = 0, \quad \vec{x} \in \Omega = (0, 100) \times (-100, 0), \\ K(\psi)\nabla(\psi + z) \cdot \vec{\nu} = 0, \quad x = 0, 100, \quad \text{or} \quad z = -100, \\ \psi = 0, \quad z = 0, \\ \psi = -1000, \quad t = t_0, \quad \vec{x} \in \Omega. \end{aligned} \tag{8.46}$$

This models the infiltration of a steep wetting front into (relatively) dry soil. The domain to solve on is given in Figure 8.3. The shaded region is a clay soil (A), and the

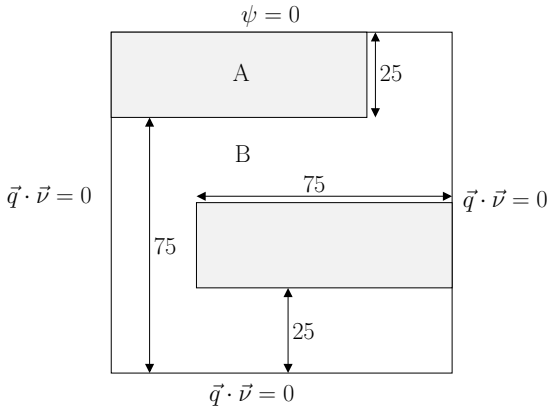


Figure 8.3: Distribution of soils on domain $\Omega = (0, 100) \times (-100, 0)$ for which to solve (8.46)

rest of the domain is loam (B). The soil properties are defined using the parameters given in Table 8.7, which are taken from [186, Table 2]. The parameter ψ_s in Equation (8.24) is taken as $-10.0(-6.0)$ for the clay (loam) soil, respectively. This value is known to change the shape of the hydraulic conductivity function and, in experimentation for which the Richards equation should give accurate results, these values should be determined by fitting curves to recorded data. This was not done for the current investigation, as the comparison between nonlinear multigrid methods, rather than obtaining improved results for the Richards equation, is of primary interest. The parameter n is termed the *tortuosity* of the soil [186], and gives a relation for how far water travels through a soil matrix for every unit distance along the component axes. The experimental set up in Figure 8.3 is chosen for numeric reasons, so that the wetting front does not enter uniformly and that there will be regions of saturated and unsaturated soils as time progresses. Also, the varying speeds

of filtration may lead to fine level variation of the pressure head as it infiltrates the soil. In this case, using a coarsest grid which does not capture this information may lead to a non-convergent FAS iteration, as the solution on the coarsest grid does not resemble the solution on finer grids, which may produce incorrect coarse grid corrections.

	Soil A: Clay	Soil B: Loam
θ_r $(\mathrm{m}^3/\mathrm{m}^3)$	0.068	0.078
θ_s $(\mathrm{m}^3/\mathrm{m}^3)$	0.380	0.430
α (cm^{-1})	0.008	0.036
n $(-)$	1.09	1.56
K_s $(\mathrm{cm/day})$	4.80	24.96

Table 8.7: Soil properties for the soils used in Figure 8.3, listed with the units in which they are measured. These are taken from [186, Table 2].

The different regions of soil are resolved on all grid levels, and we do not seek to perform experiments where the regions are not resolved on all grid levels. This is because a transfer function from a fine to a coarse grid for the parameters given in Table 8.7, as well as the parameter ψ_s, would be required for an FAS iteration. This is a non-trivial task and detracts from the comparison of the methods. We do note, however, that such a transfer function is *not* required for the Newton-MG iteration. This is because the linear coarse grid operators are determined using the Galerkin product, where a coarse grid operator is determined purely algebraically from a fine grid operator. In the case that a symmetric operator is taken (or the symmetric part of the operator) the Galerkin coarse grid product gives the projection in energy norm of the fine grid operator onto the coarse grid (see Subsection 4.1.2). So long as the soils are resolved on the finest grid level the Galerkin product does not require information regarding the underlying soil structure to form the coarse grid linear operators.

As discussed in Section 7.7 the coefficient of the linearised problem is piecewise constant. For problem Equation (8.46) there are large changes in pressure and a steep wetting front so that the gradients and function values are highly varying. The varying soil properties also contribute to jumps in coefficient. However, for this complex problem, for which the linearisation is non-symmetric, it is not known how the variation in the coefficient will affect the convergence of the methods. Results in Section 7.7 suggest that the Newton iteration is less susceptible to changes in the coefficient function. Results are investigated to see if this has an effect on any of the methods.

In order to demonstrate robustness of the methods in this subsection, instead of repeating a similar experiment as was performed for the porous medium equation (see Table 8.4), where the maximum allowable time step is given, we present the performance of

the methods in conjunction with a simple adaptive time stepping strategy. An experiment is run from time $t_0 = 0$ days to time $t_{\text{end}} = 1$ day. An initial time step size of 1e-6 is given to ensure that the nonlinear iterations converge for the first time step. The adaptation strategy is described in Algorithm 8.1 and follows a similar strategy put forward in [101]. In Algorithm 8.1 the choice of subscripts used for the different parameters are chosen as follows: η_s is the number of nonlinear iterations considered too (s)mall; η_l is the number of nonlinear iterations considered too (l)arge; η_{max} is the (max)imum number of nonlinear iterations to perform per time step; β_u is the factor used to scale the time step (u)p; β_d is the factor used to scale the time step (d)own; and δt_{min} and δt_{max} are the (min)imum and (max)imum time steps to use.

Algorithm 8.1 Adaptive Time Stepping

Require: η_s, η_l, η_{max}, β_u, β_d, δt_{min}, δt_{max}

```
function AdaptTimeStep(δt, η)
    if η > η_max then
        if δt = δt_min then
            Fail
        else
            δt ← β_d δt
            Repeat current time step
    else if η ≥ η_l then
        δt ← β_d δt
    else if η ≤ η_s then
        δt ← β_u δt
    Ensure δt ∈ [δt_min, δt_max]
```

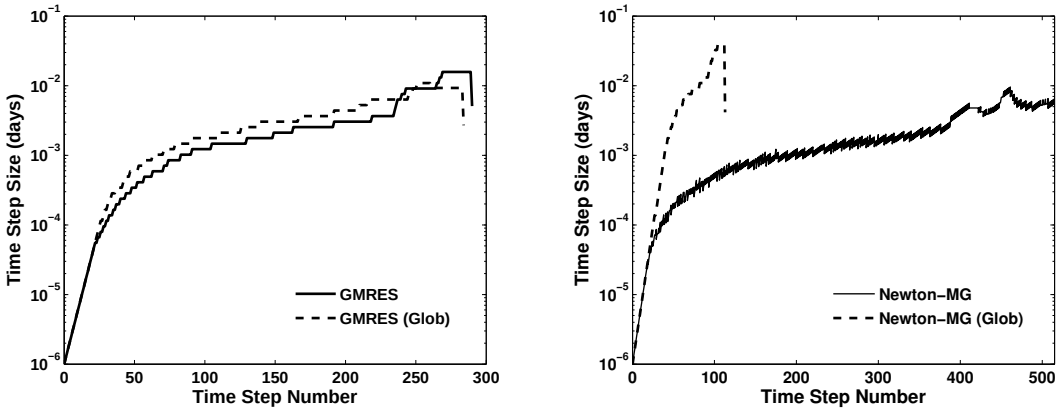
For each of the iterations the number of nonlinear iterations performed is taken as an indicator of whether the time step parameter should be increased or decreased, as is performed in other works (*e.g.* [101, 188]). This comes from the desire to minimise the number of nonlinear iterations performed. The fewer time steps that are performed the fewer nonlinear iterations need to be performed, so that taking a large time step is desirable. On the other hand, the number of iterations performed per time step is likely to be lower for smaller time steps, so that fewer nonlinear iterations are performed overall if more time steps are taken at lower time scales. Also, discussion of the results for the porous medium equation (see Subsection 8.1.3) suggest that when FAS converges fast it is due to the fact that the diagonal mass matrix term is dominant. Therefore the problem being solved is close to linear, and an increase in the time step is desirable to be able to take advantage of the fact that the solvers can deal with nonlinear problems. The parameters used in the time-stepping are presented in Table 8.8. The row with heading

'Inexact Newton' refers to both the Newton-MG and preconditioned Newton-GMRES iterations. As it has been shown in previous discussion that the Newton method is more robust and more efficient than FAS, the number of iterations performed for increasing and decreasing the time step is larger than for FAS.

Method	η_s	η_l	$\eta_{\max}$	β_u	β_d	$\delta t_{\min}$	$\delta t_{\max}$
Inexact Newton	6	12	20	1.2	0.85	1e-6	1e-1
FAS	5	10	20	1.2	0.85	1e-6	1e-1

Table 8.8: Parameters used in the adaptive time-step procedure (see Algorithm 8.1) for FAS and Newton-MG

From the previous discussion it is expected that the Newton iteration will allow a larger time step parameter than the FAS iteration, and hence that the Newton iteration will reach the target time in fewer time steps. This, combined with the much greater efficiency of the Newton method, suggests that a large reduction in the execution time should be expected compared to the FAS iteration.

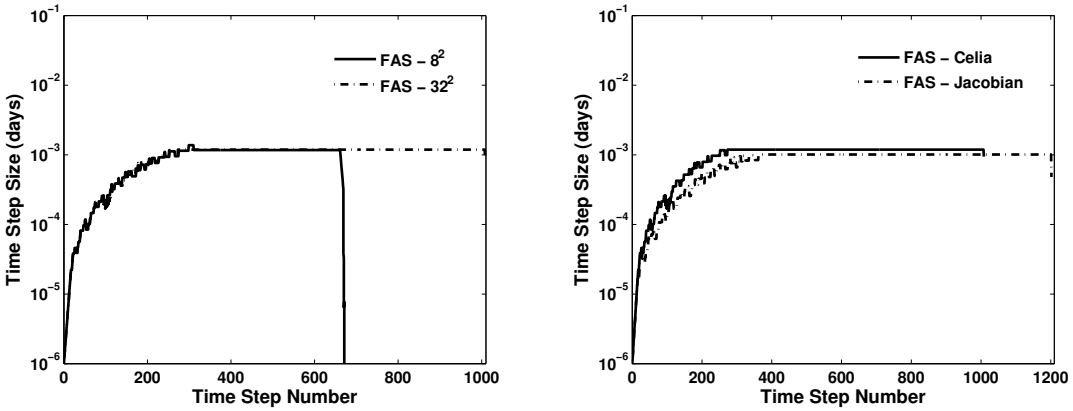


(a) Progression of time-step size for a Newton-GMRES iteration where multigrid is used to precondition the symmetric part of the Jacobian.

(b) Progression of time step size for a Newton-MG iteration.

Figure 8.4: Progression of time step size for inexact and global-inexact Newton iterations. Armijo's rule (see Algorithm 7.2) with minimum scaling fact $\gamma_{\min} = 2^{-7}$ is used as the globalisation method. The Jacobian matrix is used in the linearisation.

Figures 8.4 and 8.5 show the progression of the size of the time step for a problem discretised on a 512^2 grid. The sharp drop at the final iteration is so that the target time is reached exactly. Curves in Figure 8.4 show global as well as non-global Newton iterations. As we have not been able to stabilise the convergence of the FAS in a predictable



(a) Progression of time step size for an FAS iteration using different coarsest grids in the multigrid hierarchy. Linearisation follows the model by Celia.

(b) Progression of time step size for an FAS iteration using different linearisations with 32^2 coarsest grid.

Figure 8.5: Progression of time step size for an FAS iteration using different linearisations and different coarse grids.

manner (see Section 7.6) no global version of the FAS is used. As predicted, the time step for a Newton-MG iteration is significantly larger than that taken in an FAS iteration. The number of time steps performed is therefore much less, and the overall execution time (shown in Table 8.9) is significantly lower.

Figure 8.5a shows the progression of the time step size for an FAS iteration using the Celia model (see Subsection 8.2.2). The solid line is when an 8^2 coarsest grid is used in the multigrid hierarchy, and the dotted line is for a 32^2 grid. It is found that at some point in time the hierarchy with the coarser grid fails to converge, even for extremely low time scales. This is because the solution contains fine scale detail for which the 8^2 grid is too coarse for the solution to be well represented. Moving to a 32^2 grid resolves this issue, but comes at a computational expense. The extra work required to solve on the finer grid means that a V-cycle takes an extra 10% longer to complete. This can be found by taking the average amount of time per V-cycle for the iterations from the data given in Table 8.9. It is also clear to see that as time progresses a steady time step size is reached for the FAS iteration. The problem then does not get any more or less difficult for the FAS iteration to solve, so that the number of iterations required for convergence at each time step stays in the interval $[6, 9]$.

Figure 8.5b shows the progression of the time step size when the model by Celia is used (with a 32^2 coarsest grid), compared to using the full Jacobian matrix in the linearisation. Results in Chapter 7 showed that the convergence of the FAS iteration is less

sensitive to changes or errors in the linearisation than the Newton iteration. In fact we see an improvement in the nonlinear convergence when the 'worse' linearisation is taken in the Celia model. This is because the terms in the linearisation for the Celia model are symmetric positive definite, for which the FAS iteration works well. As was shown in Section 8.1, the FAS does not converge well for problems where a non-symmetric term arises in the linearisation. Using only the symmetric part of the linearisation improves the convergence of the nonlinear iteration, demonstrated by the faster rate of increase in time step, as well as the larger maximum time step, in Figure 8.5b. A saving is made in terms of computational expense as well, as the derivative of the hydraulic conductivity function does not need to be calculated in the linearisation (see (8.39)). The timing shown in Table 8.9 shows that the Celia model requires half the amount of time as when the full Jacobian is used in the linearisation. This suggests that computational savings may be made in the FAS iteration by dropping unnecessary or undesirable terms from a linearisation. However, the iteration will still likely take a lot longer to execute than an inexact Newton iteration.

Method	Time Steps	V-Cycles	Time (s)
Newton-MG	516	11382	2902
Newton-MG (Glob)	114	1995	596
Newton-GMRES	291	6081	1413
Newton-GMRES (Glob)	285	6102	1483
FAS (8^2)	671	5270	8985
FAS (32^2)	1009	6940	13028
FAS (Jacobian)	1203	8119	26622

Table 8.9: Time steps and V-cycles performed, as well as the execution time for the nonlinear iterations applied to (8.46). The Newton iterations use a coarsest grid 8^2.

In contrast to the FAS, the time step size for an inexact Newton iteration may increase beyond 1e-3, and many fewer time steps are required to reach the target time of 1 day. From Figure 8.4 and Table 8.9 it seems as though the best algorithm to use here would be a global Newton-MG iteration. However, this is not advisable. Although this method requires fewer time steps – and hence less time – to reach the target time, the convergence of the linear multigrid iteration is not guaranteed for larger time steps. The nonlinear iteration does not converge unless it is stabilised for some larger time steps. This can be seen in Figure 8.4b as reduction of the time step is required much more often, which is shown by regular drops in the graph. At time steps when a reduction in the size of the step is required, it is found that a multigrid iteration may be non-convergent and an increase in the linear residual is observed. Without a stabilisation of the Newton correction this

leads to a poor nonlinear correction term being calculated and the nonlinear iteration diverges. This results in the time step being reduced, and the iteration repeated, giving a convergent linear multigrid iteration. However, even with a poor convergence of the linear iteration it is found, for this example, that using a stabilisation parameter gives a reduction in the nonlinear residual from one iteration to the next. This allows for a much larger time step to be taken compared with the non-global iteration, and the iteration seems to be efficient. However, the fact that the linear iteration does not converge means that convergence behaviour is not predictable, and even though the method works well in this example, it may not do so for another.

A contraction in the linear residual is observed at every iteration when a multigrid-preconditioned GMRES iteration is used in the inner iteration of the Newton method. The preconditioner is applied only to the symmetric part of the Jacobian matrix (8.39), as it was found that the multigrid iteration may not be convergent for a given time step. The Newton correction calculated does not give as good a nonlinear convergence as in the case of the linear multigrid iteration, but we can be confident that the nonlinear and the linear iteration are performing predictably and stably. In the case that a non-global method is used the advantage of the preconditioned GMRES iteration is clear over the use of a Newton-MG iteration, as the size of the time step does not need to be reduced, and a time step is not repeated. This demonstrates the greater robustness of the method. The timing results in Table 8.9 show that almost an order of magnitude reduction in the execution time is still gained over the FAS iteration for this problem. An interesting behaviour can be seen in the comparison of the global and non-global iterations in Figure 8.4a. At early time steps the advantage of using the globalisation method can be seen, as a better correction term is chosen, and a larger time step may be taken. However, at later time steps we see that a larger time step is allowed for the non-global iteration. This highlights a problem with the globalisation procedure, as discussed in Section 7.6. By forcing a contraction in the nonlinear residual the correction term ends up being too small at the later iterations for the Newton method to converge quickly. What is better here is to not damp the correction term. This may cause an increase in the nonlinear residual, but causes the solution to change more significantly from its previous value, allowing a different (in this case better) Newton correction to be calculated, so that overall fewer nonlinear iterations are required for convergence. A more sophisticated stabilisation procedure that calculates a damping parameter based on properties of the nonlinear problem being solved may give an improved performance (see [58, §3]), and the discussion here highlights that whilst the simple Armijo strategy (see Algorithm 7.2) gives a convergent iteration it may not be the best stabilisation algorithm to use.

Results for other grid levels are qualitatively the same, and are not presented here for reasons of brevity. The experiments run previously should give confidence that the findings are the same for finer and coarser grid resolutions. The results that have been presented clearly show that the Newton iteration is a much more computationally efficient algorithm for this more complicated model problem. The time required for the linear multigrid iterations decreases with respect to the time to calculate a nonlinear residual, making the method more computationally efficient with respect to the FAS iteration. Results also show that Newton's method is more likely to be able to resolve fine mesh detail using coarse grids, as is backed up by results in Section 7.7. We again conclude that Newton-MG is a superior algorithm to FAS, and results show that changing the inner iteration to a preconditioned GMRES gives more robust and predictable results.

The formulation of the Richards equation given in this section is *globally* mass conservative [48]. Numerical inaccuracies will lead to a loss in mass in the system, which, although small, is undesirable. A judge of the quality of a discretisation of the Richards equation is the mass conservative properties of the discretisation [157]. Therefore we present an alternative discretisation of the Richards equation in the next section, using a mixed finite element formulation. This is taken from [13], and is *locally* mass conservative. This means that the loss of mass due to numerical and discretisation error is much less pronounced. The discussion and results again serve to highlight the advantage in reducing the number of nonlinear problems to solve as part of a solution procedure.

8.3 Richards Equation as a System of Equations

In this section the solution of the Richards equation is considered when the discretisation scheme used is locally mass conservative. Instead of a single equation to solve, the equation is considered in saddle point form. The discretisation method is described in detail in Subsection 8.3.1. This problem is investigated in particular to demonstrate the advantage in execution time provided by the Newton method in removing as many nonlinear components from the solution process as possible. For the mixed finite elements much of the linear multigrid convergence theory does not hold (such as a proof of convergence for the V-cycle [13]) and, to the best of our knowledge, an investigation of how FAS performs for a mixed finite element discretisation has not been conducted. Results in Subsection 8.3.3 suggest that the Newton approach provides a viable solution method, whilst the FAS performs poorly.

The section is structured as follows. Necessary notation is introduced in order to form a mixed finite element discretisation of the Richards equation before the problem formu-

lation and discretisation is outlined in Subsection 8.3.1. This is followed by a description of necessary adjustments to a multigrid algorithm applied to the mixed formulation in Subsection 8.3.2. The framework in Chapter 6 is not applicable for this problem, but the investigation of the methods within the framework suggests that a Newton-MG iteration will be more efficient in comparison to the FAS iteration for this more complex nonlinear problem. This prediction is confirmed by the results presented in Subsection 8.3.3, which investigates the execution time and the robustness of the nonlinear iterations.

The discretisation used in this section uses a space of finite elements which we have not encountered so far, called *Raviart-Thomas* elements. This space is introduced, along with some necessary notation, before the discretisation is outlined in Subsection 8.3.1. The summary given here is adapted from [155, §2.2.2], which gives an excellent introduction (including some of the theory) to Raviart-Thomas elements. A full description of the Raviart-Thomas element, along with technical proofs, can be found in [41].

Assume that the continuous spatial domain $\Omega \subset \mathbb{R}^2$ is polygonal with boundary $\partial\Omega = \Gamma_D \cup \Gamma_N$ for Γ_D the part of the boundary where a Dirichlet condition is specified, and Γ_N the part of the boundary where a Neumann condition is specified for some problem. Let Ω_j be a grid with associated quasi-regular triangulation $\mathcal{T}_j$. Then $\mathcal{E}_j^{(I)}$ is the set of edges in the interior of Ω_j, and $\mathcal{E}_j^{(D)}$ and $\mathcal{E}_j^{(N)}$ are the set of edges on the Dirichlet and Neumann boundary, respectively. The edge set $\mathcal{E}_j = \mathcal{E}_j^{(I)} \cup \mathcal{E}_j^{(D)} \cup \mathcal{E}_j^{(N)}$.

The following function spaces are required for the discussion in this section:

$$H(\Omega; \mathrm{div}) \equiv \{\vec{u} \in (L^2(\Omega))^d \mid \nabla \cdot \vec{u} \in L^2(\Omega)\}, \tag{8.47}$$

the space of d-dimensional vectors of L^2 measurable functions for which the divergence is L^2 measurable; $P_k(T)$, the set of polynomials of degree less than or equal to k on $T \in \mathcal{T}_j$; and

$$RT_k(T) \equiv \{\vec{\alpha} + \gamma\vec{x} \mid \vec{x} \in T,\ \vec{\alpha} \in (P_k(T))^d,\ \gamma \in P_k(T)\}, \tag{8.48}$$

the space of Raviart-Thomas elements of order k on $T \in \mathcal{T}_j$. We consider only the case $d = 2$. We also require the space

$$H_0(\Omega; \mathrm{div}) = \{\vec{u} \in H(\Omega; \mathrm{div}) \mid \vec{u} \cdot \vec{\nu}_{\Gamma^N} = 0\}. \tag{8.49}$$

For the purposes of this section we are interested in the lowest order Raviart-Thomas

space RT_0. For $\vec{u} \in RT_k$ the following holds (see [155, Proposition 2.12])

$$\vec{u} \cdot \vec{\nu}|_{\partial T} \in \{u \in L^2(\partial T) \mid u|_E \in P_k(E) \text{ for } E \in \partial T\}. \tag{8.50}$$

Therefore, the normal moment of a function in $RT_0(T)$ on each edge E of element T is constant.

In order to define the basis for RT_0 it is useful to define the set

$$R^+ \equiv \{\vec{x} \in \mathbb{R}^2 \mid x > 0\} \cup \{[0,1]^\top\}, \tag{8.51}$$

such that a unique normal $\vec{\nu}_E \in R^+$ can be associated with each edge on the mesh. A basis function for $RT_0(T)$ has three degrees of freedom, which are typically taken as the constant normal moments on ∂T with respect to $\vec{\nu}_E$. A typical basis function has one normal moment equal to one and the others equal to zero, *i.e.* basis function $\vec{\varphi}_E$ associated with edge E satisfies

$$\vec{\varphi}_E \cdot \vec{\nu}_{E'} = \delta_{E,E'}, \tag{8.52}$$

where $\delta_{E,E'}$ is the Kronecker delta. Equation (8.52) is sufficient to define the basis functions on the element. The normal with respect to an edge is used here so that basis functions on different elements sharing the same edge will have the same normal component.

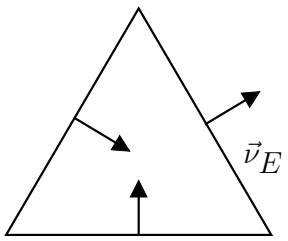


Figure 8.6: Normal vectors $\vec{\nu}_E \in R^+$ on an element.

The Raviart-Thomas space $RT_0(T) \subset H(T; \mathrm{div})$, which can be seen from the definition of the space (8.48). In particular $\nabla \cdot RT_0(T) = P_0(T)$. In order to obtain a subspace of $H(\Omega_j; \mathrm{div})$ the space

$$\mathcal{RT}_0(\Omega_j) = \{\vec{u} \in H(\Omega_j; \mathrm{div}) \mid \vec{u}|_T \in RT_0(T),\ T \in \mathcal{T}_j\} \tag{8.53}$$

is introduced. Note that the normal component over the edges of elements is continuous for functions in $\mathcal{RT}_0(\Omega_j)$ due to the condition $\vec{u} \in H(\Omega_j; \mathrm{div})$. The homogeneous Neumann boundary conditions are satisfied by the subspace

$$\mathcal{V}_j \equiv \{\vec{u} \in \mathcal{RT}_0(\Omega_j) \mid \vec{u} \cdot \vec{\nu}|_{\Gamma_N} = 0\}. \tag{8.54}$$

It can be shown that $\dim(\mathcal{V}_j) = |\mathcal{E}_j^{(I)} \cup \mathcal{E}_j^{(D)}|$ [155, §2.2.2], such that a function $\vec{u} \in \mathcal{V}_j$ can be identified by the vector $[u_E]$, which contains the sizes of the normal moments over all edges $E \in \mathcal{E}_j^{(I)} \cup \mathcal{E}_j^{(D)}$, *i.e.*

$$\vec{u} = \sum_{E \in \mathcal{E}_j^{(I)} \cup \mathcal{E}_j^{(D)}} u_E \vec{\varphi}_E. \tag{8.55}$$

This notation will be used in the mixed finite element discretisation of the Richards equation, as outlined in the next subsection.

8.3.1 Mixed Finite Element Discretisation

In this section we introduce the mixed finite element discretisation of the Richards equation. The domain on which to solve is given by $\Omega \equiv (0, 100) \times (-100, 0)$, and is in the xz-plane. The formulation used in the previous section (see (8.22)) is a single equation involving different nonlinear functions of pressure head ψ. It is a direct combination of the nonlinear Darcy law (8.20) and continuity of flux (8.21). If the equations are not combined we are led to the following saddle point problem:

$$\begin{aligned} \frac{\partial}{\partial t}\theta(\psi) + \nabla \cdot \vec{q} &= 0, \\ \vec{q} + K(\psi)\nabla(\psi + z) = 0, \quad \vec{x} \in \Omega, \quad t > t_0, \\ \vec{q} \cdot \vec{\nu} = 0, \quad \vec{x} \in \Gamma_N, \quad t > t_0, \\ \psi = \psi_D, \quad \vec{x} \in \Gamma_D, \quad t > t_0, \\ \psi = \psi_0, \quad t = t_0, \end{aligned} \tag{8.56}$$

where $\Gamma_N \subset \partial\Omega$ is the part of the boundary where a Neumann condition is specified, and $\Gamma_D \subset \partial\Omega$ is the part of the boundary where a Dirichlet condition is specified. The unknown functions to solve for are now the flux $\vec{q}$ and pressure head ψ. For simplicity we set $\psi_D = 0$, although non-homogeneous conditions are not complicated to include

(see [13]).

To develop a mixed finite element formulation the regularity of the functions is required. Some necessary notation is introduced before giving the regularity estimates. Let $\mathcal{J} = (t_0, t_{\max})$ for given times t_0, $t_{\max}$. Then

$$L^p(\mathcal{J}; W^{r,q}(\Omega)) = \{u(t) = u(t, \vec{x}) \mid (\int_{\mathcal{J}} \|u(t)\|^p_{W^{r,q}(\Omega)} \, \mathrm{d}t)^{1/p} < \infty\}. \tag{8.57}$$

Regularity and existence results for (8.56) are summarised in [13]. These may be relaxed under physically realistic assumptions in order to allow a discretisation of (8.56) using mixed P_0 and lowest order Raviart-Thomas elements [13]. The regularity estimates are given by

$$\begin{gathered} \theta(\psi) \in L^\infty(\mathcal{J}; L^\infty(\Omega)), \quad \vec{q} \in L^\infty(\mathcal{J}; H_0(\Omega; \mathrm{div})) \\ \frac{\partial}{\partial t}\theta(\psi) \in L^2(\mathcal{J}; (L^2)'(\Omega)). \end{gathered} \tag{8.58}$$

Although L^2 is self-dual, the notation $(L^2)'$ is kept to show that the derivative of θ is, strictly speaking, in the dual space.

We first discretise (8.56) in time using an implicit Euler discretisation, as in the previous section, to give

$$\begin{gathered} \theta^{(k)} + \delta t \nabla \cdot \vec{q}^{(k)} = \theta^{(k-1)}, \\ \vec{q}^{(k)} + K^{(k)} \nabla(\psi^{(k)} + z) = 0, \quad \vec{x} \in \Omega, \quad t > t_0, \\ \vec{q}^{(k)} \cdot \vec{\nu} = 0, \quad \vec{x} \in \Gamma_N, \quad t > t_0, \\ \psi^{(k)} = 0, \quad \vec{x} \in \Gamma_D, \quad t > t_0, \end{gathered} \tag{8.59}$$

where

$$\theta^{(k)} = \theta(\psi^{(k)}) = \theta(\psi(\vec{x}, t_k)),$$

and

$$t_k = t_0 + k\delta t$$

for constant time step δt and index $k > 0$. $K^{(k)}$ is defined similarly to $\theta^{(k)}$, and $\vec{q}^{(k)} =$

$\vec{q}(\vec{x}, t_k)$. We re-write the second equation in (8.59) as

$$(K^{(k)})^{-1}(\vec{q}^{(k)}) + \nabla(\psi^{(k)} + z) = 0. \tag{8.60}$$

For the time discrete equations we have the following inclusions

$$\psi^{(k)} \in L^2(\Omega), \quad \vec{q}^{(k)} \in H_0(\Omega; \mathrm{div}). \tag{8.61}$$

Taking test functions $w \in L^2$ and $\vec{v} \in H_0(\Omega; \mathrm{div})$ and integrating (8.59) over Ω gives the weak form

$$\begin{aligned} \int_\Omega \theta^{(k)} w \,\mathrm{d}\vec{x} + \delta t \int_\Omega \nabla \cdot \vec{q}^{(k)} w \,\mathrm{d}\vec{x} &= \int_\Omega \theta^{(k-1)} w \,\mathrm{d}\vec{x}, \\ \int_\Omega (K^{(k)})^{-1} \vec{q}^{(k)} \vec{v} \,\mathrm{d}\vec{x} - \int_\Omega \psi^{(k)} \nabla \cdot \vec{v} \,\mathrm{d}\vec{x} + \int_\Omega \nabla z \vec{v} \,\mathrm{d}\vec{x} &= 0, \end{aligned} \tag{8.62}$$

for all $w \in L^2(\Omega)$ and $\vec{v} \in H_0(\Omega; \mathrm{div})$. Here, the homogeneous boundary conditions have been used to simplify the form. The final step in the discretisation is to integrate over a grid Ω_j instead of the continuous domain, and choose appropriate finite dimensional function spaces. Let

$$W_j \equiv \{w \in L^2(\Omega_j) \mid w|_T \in P_0(T),\ T \in \mathcal{T}_j\} \tag{8.63}$$

and

$$V_j \equiv \{\vec{v} \in H_0(\Omega_j; \mathrm{div}) \mid \vec{v}|_T \in RT_0(T),\ T \in \mathcal{T}_j\}. \tag{8.64}$$

A basis for V_j is given by the Raviart-Thomas basis functions on the edges $\mathcal{E}_j^{(I)} \cup \mathcal{E}_j^{(D)}$ (see (8.55)). A basis for W_j is given by the set of characteristic functions

$$\chi_T = \begin{cases} 1, & \vec{x} \in T, \\ 0, & \text{otherwise.} \end{cases} \tag{8.65}$$

Then the discretised system of nonlinear equations to solve at each time step is given by (8.62), with $\psi^{(k)} \in W_j$ and $\vec{q}^{(k)} \in V_j$, for all $w \in W_j$ and $\vec{v} \in V_j$ on grid Ω_j. At this point it is possible to solve the discrete system of equations using a nonlinear iteration.

However, the system of equations to solve is of dimension $|W_j| + |V_j|$, which may be very large.

Instead of solving the entire system, the dimension of the nonlinear problem to solve can be reduced by introducing Lagrange multipliers, as described in [13]. Let

$$\hat{V}_j \equiv \{\vec{v} \in L^2(\Omega_j) \mid \vec{v}|_T \in RT_0(T),\ T \in \mathcal{T}_j,\ \vec{v} \cdot \vec{\nu}|_{\Gamma_N} = 0\}, \tag{8.66}$$

which is similar to V_j, except that the continuity of the normal components across element boundaries is not required. As such the normal component across an element boundary (*i.e.* an edge) is *not* uniquely defined, and the basis of $\hat{V}_j$ is taken as the set

$$\{\vec{\varphi}_{T,E}\},\ \text{for all } T \in \mathcal{T}_j, \quad E \in \mathcal{E}_j^{(I)} \cup \mathcal{E}_j^{(D)}, \tag{8.67}$$

where, instead of satisfying condition (8.52) the condition

$$\vec{\varphi}_{T,E} \cdot \vec{\nu}_{T,E'} = \delta_{E,E'} \tag{8.68}$$

is satisfied for $\vec{\nu}_{T,E'}$ the outward facing normal on edge E' with respect to element T. Hence the representation of $\vec{v} \in \hat{V}_j$ is

$$\vec{v} = \sum_{T \in \mathcal{T}_j} \sum_{E \in \partial T} v_{T,E} \vec{\varphi}_{T,E}. \tag{8.69}$$

An explicit form of $\vec{\varphi}_{T,E}$ is given by

$$\vec{\varphi}_{T,E}(\vec{x}) = \frac{\vec{x} - \vec{x}_N}{2\,|T|}, \tag{8.70}$$

where $\vec{x}_N$ is the vertex of element T not incident to edge E. As well as $\hat{V}_j$, the space of Lagrange multipliers

$$M_j \equiv \{u \in L^2(\mathcal{E}_j) \mid u|_E \in P_0(E),\ E \in \mathcal{E}_j,\ u|_{\Gamma_D} = 0\}, \tag{8.71}$$

is introduced, where $\mathcal{E}_j$ is the set of all edges in Ω_j and $\partial\Omega_j$. Note that due to the homogeneous Dirichlet conditions the Lagrange multipliers vanish on the Dirichlet boundary.

For a function $\vec{v} \in \hat{V}_j$ the condition

$$\sum_{T \in \mathcal{T}_j} \int_{\partial T} \vec{v} \cdot \vec{\nu} \mu \, \mathrm{d}S = 0, \tag{8.72}$$

for all $\mu \in M_j$ and $\vec{\nu}$ the outward facing normal on ∂T, holds if and only if $\vec{v} \in V_j$ [41, pg. 179]. In other words (8.72) enforces the condition that the normal component across element boundaries is continuous. Using this condition, and taking the basis functions of W_j, $\hat{V}_j$ and M_j as test functions, the discrete system of equations to solve becomes

$$\begin{aligned}
&\int_{\Omega_j} \theta(\psi^{(k)}) \chi_T \, \mathrm{d}\vec{x} + \delta t \int_{\Omega_j} \nabla \cdot \vec{q}^{(k)} \chi_T \, \mathrm{d}\vec{x} = \int_{\Omega_j} \theta(\psi^{(k-1)}) \chi_T \, \mathrm{d}\vec{x}, \quad \forall T \in \mathcal{T}_j, \\
&\int_{\Omega_j} (K(\psi^{(k)}))^{-1} \vec{q}^{(k)} \vec{\varphi}_E \, \mathrm{d}\vec{x} - \int_{\Omega_j} \psi^{(k)} \nabla \cdot \vec{\varphi}_E \, \mathrm{d}\vec{x} + \int_{\Omega_j} \nabla z \vec{\varphi}_E \, \mathrm{d}x = \\
&\qquad\qquad - \sum_{T \in \mathcal{T}_j} \int_{\partial T} \lambda^{(k)} \vec{\varphi}_E \cdot \vec{\nu} \, \mathrm{d}S, \quad E \in \mathcal{E}_j^{(I)} \cup \mathcal{E}_j^{(D)} \\
&\sum_{T \in \mathcal{T}_j} \int_{\partial T} \vec{q}^{(k)} \cdot \vec{\nu} \chi_E \, \mathrm{d}S = 0, \quad E \in \mathcal{E}_j \setminus \mathcal{E}_j^{(D)},
\end{aligned} \tag{8.73}$$

for $(\psi^{(k)}, \vec{q}^{(k)}, \lambda^{(k)}) \in W_j \times \hat{V}_j \times M_j$. At first sight this may seem to only have increased the size of the system. However, it is possible to eliminate internal degrees of freedom in order that the solution can be gained by solving only for the Lagrange multipliers $\lambda^{(k)}$. This follows the methodology in [13, 157], but in the following, mistakes in the presentation of [13, 157] are corrected. All details of the derivation are not repeated here for brevity, but the main calculations are shown in order for a reader to be able to reproduce the discretisation. All quantities given in this subsection can be calculated by performing the explicit integration of the terms above using the discrete representations

$$\begin{aligned}
\psi^{(k)} = \sum_{T \in \mathcal{T}_j} \psi_T^{(k)} \chi_T, \qquad \lambda^{(k)} = \sum_{E \in \mathcal{E}_j \setminus \mathcal{E}_j^{(D)}} \lambda_E^{(k)} \chi_E \\
\vec{q}^{(k)} = \sum_{T \in \mathcal{T}_j} \sum_{E \in \partial T} q_{T,E}^{(k)} \vec{\varphi}_{T,E}.
\end{aligned} \tag{8.74}$$

The following quantities are of particular interest. Define the local matrix

$$B_T = \left[B_{T,EE'} \right]_{E,E' \in \partial T}, \tag{8.75}$$

where

$$B_{T,EE'} = \int_T \vec{\varphi}_{T,E} \cdot \vec{\varphi}_{T,E'} \, \mathrm{d}\vec{x} \tag{8.76}$$

on each element $T \in \mathcal{T}_j$. The quantity

$$z_{T,E} = \int_T \nabla z \vec{\varphi}_{T,E} \, \mathrm{d}\vec{x}, \tag{8.77}$$

is also useful. Let

$$\omega_E = \{T \in \mathcal{T}_j \mid \partial T \cap E \neq \emptyset\}. \tag{8.78}$$

Using the representations of the functions in (8.74) and integrating the equations in (8.73) exactly gives the system of equations

$$\begin{aligned} \theta(\psi_T^{(k)}) + \frac{\delta t}{|T|} \sum_{E \in \partial T} q_{T,E}^{(k)} &= \theta(\psi_T^{(k-1)}), \quad \forall T \in \mathcal{T}_j, \\ \sum_{E' \in \partial T} B_{T,EE'} q_{T,E'}^{(k)} &= K(\psi_T^{(k)})(\psi_T^{(k)} - \lambda_E^{(k)} - z_{T,E}), \quad \forall T \in \mathcal{T}_j, \quad E \in \partial T, \\ \sum_{T \in \omega_E} q_{T,E}^{(k)} &= 0, \quad \forall E \in \mathcal{E}_j \setminus \mathcal{E}_j^{(D)}. \end{aligned} \tag{8.79}$$

Considering the second equation in (8.79) over all edges on an element we get

$$B_T \begin{bmatrix} q_{T,E}^{(k)} \\ q_{T,E'}^{(k)} \\ q_{T,E''}^{(k)} \end{bmatrix} = K(\psi_T^{(k)}) \begin{bmatrix} \psi_T^{(k)} - \lambda_E^{(k)} - z_{T,E} \\ \psi_T^{(k)} - \lambda_{E'}^{(k)} - z_{T,E'} \\ \psi_T^{(k)} - \lambda_{E''}^{(k)} - z_{T,E''} \end{bmatrix}, \tag{8.80}$$

and hence

$$\begin{bmatrix} q_{T,E}^{(k)} \\ q_{T,E'}^{(k)} \\ q_{T,E''}^{(k)} \end{bmatrix} = K(\psi_T^{(k)}) B_T^{-1} \begin{bmatrix} \psi_T^{(k)} - \lambda_E^{(k)} - z_{T,E} \\ \psi_T^{(k)} - \lambda_{E'}^{(k)} - z_{T,E'} \\ \psi_T^{(k)} - \lambda_{E''}^{(k)} - z_{T,E''} \end{bmatrix}. \tag{8.81}$$

The 3×3 matrix B_T^{-1} can be calculated as

$$B_T^{-1} = \frac{1}{\det(B_T)} \hat{B}_T^\top, \tag{8.82}$$

for $\hat{B}_T$ the cofactor matrix of B_T. For the two-dimensional case explicit values are given by

$$\det(B_T) = \sum_{E \in \partial T} |E|^4 \left(|E|^2 - \sum_{\substack{E' \in \partial T \\ E' \neq E}} |E'|^2 \right) - 6 \prod_{E \in \partial T} |E|^2 , \tag{8.83}$$

$$B^{-1}_{T,EE'} = \frac{16\,|T|}{\det(B_T)} \left(\delta_{EE'} \sum_{\substack{\tilde{E} \in \partial T \\ \tilde{E} \neq E}} \left|\tilde{E}\right|^2 \left(\left|\tilde{E}\right|^2 - |E|^2 \right) + \right.$$

$$\left. (1 - \delta_{EE'}) \left(\sum_{\tilde{E} \in \partial T} (\delta_{E\tilde{E}} + \delta_{E'\tilde{E}} + 0.5) \left|\tilde{E}\right|^4 + |E|^2 \, |E'|^2 \right) - \sum_{\substack{\tilde{E}, \hat{E} \in \partial T \\ \tilde{E} \neq \hat{E}}} \left(\left|\tilde{E}\right|^2 - \left|\hat{E}\right|^2 \right) \right), \tag{8.84}$$

$$\beta_T \equiv \sum_{E' \in \partial T} B^{-1}_{T,EE'} = 48\,|T| \left(\sum_{E'' \in \partial T} |E''|^2 \right)^{-1} , \tag{8.85}$$

$$z_{T,E} = \frac{1}{3}(z_E - z_O) \tag{8.86}$$

for z_E the z-coordinate of the mid-point of edge E, and z_O the z-coordinate of the vertex of element T not incident to edge E. Finally we note that

$$\sum_{E \in \partial T} z_{T,E} = 0. \tag{8.87}$$

From (8.81) the representation of $q_{T,E}$ can be written as

$$q_{T,E}^{(k)} = K(\psi_T^{(k)}) \sum_{E' \in \partial T} B_{T,EE'}^{-1}(\psi_T^{(k)} - \lambda_{E'}^{(k)} - z_{T,E'}). \tag{8.88}$$

Substituting (8.88) into the left-hand side of the first equation of (8.79) gives

$$\begin{aligned} &\theta(\psi_T^{(k)}) + \frac{\delta t}{|T|} \sum_{E \in \partial T} K(\psi_T^{(k)}) \sum_{E' \in \partial T} B_{T,EE'}^{-1}(\psi_T^{(k)} - \lambda_{E'}^{(k)} - z_{T,E'}) \\ &= \theta(\psi_T^{(k)}) + \frac{\delta t K(\psi_T^{(k)})}{|T|} \sum_{E' \in \partial T} (\psi_T^{(k)} - \lambda_{E'}^{(k)} - z_{T,E'}) \left(\sum_{E \in \partial T} B_{T,EE'}^{-1} \right) \\ &= \theta(\psi_T^{(k)}) + \frac{\delta t K(\psi_T^{(k)}) \beta_T}{|T|} (3\psi_T^{(k)} - \sum_{E \in \partial T} \lambda_E^{(k)}). \end{aligned} \tag{8.89}$$

Including the right-hand side, and rearranging in terms of λ gives

$$\begin{aligned} \sum_{E \in \partial T} \lambda_E^{(k)} &= 3\psi_T^{(k)} + \frac{|T| \left(\theta(\psi_T^{(k)}) - \theta(\psi_T^{(k-1)}) \right)}{\delta t K(\psi_T^{(k)}) \beta_T} \\ &= F(\psi_T^{(k)}). \end{aligned} \tag{8.90}$$

Equation (8.90) is a *local* function which relates the Lagrange multipliers to the pressure head on each element. This is useful, as the pressure may be recovered, given the Lagrange multipliers, by the relation

$$\psi_T^{(k)} = F^{-1} \left(\sum_{E \in \partial T} \lambda_E^{(k)} \right), \tag{8.91}$$

which can be solved using a Newton iteration for the single equation (8.90). Note that the invertibility of F may impose a restriction on the spatial and temporal step sizes (see [13,157]), which means that the time steps we are able to use for this formulation may be much smaller than for the continuous P^1 elements presented in the previous section.

To obtain the reduced nonlinear formulation, the representation of the flux unknown

(8.88) and the pressure head (8.91) are substituted into the final equation in (8.79) to give

$$\sum_{T\in\omega_E} K\left(F^{-1}(\sum_{E''\in\partial T}\lambda_{E''}^{(k)})\right)\sum_{E'\in\partial T} B_{T,EE'}^{-1}\left(F^{-1}(\sum_{E''\in\partial T}\lambda_{E''}^{(k)})-\lambda_{E'}^{(k)}-z_{T,E'}\right)=0, \tag{8.92}$$

for all $E\in\mathcal{E}_j$, as the system of equations to solve for the Lagrange multipliers $\lambda^{(k)}=[\lambda_E^{(k)}]$. Let

$$G(\lambda^{(k)})=[g_E(\lambda^{(k)})]_{E\in\mathcal{E}_j\setminus\mathcal{E}_j^{(D)}}, \tag{8.93}$$

where $g_E(\lambda^{(k)})$ is defined as the left hand side of (8.92). The final step in the discretisation is to linearise (8.92). In order to do this we require the derivative of $F^{-1}(\sum_{E\in\partial T}\lambda_E^{(k)})$ with respect to $\lambda_{E'}^{(k)}$. Note that

$$\frac{\partial}{\partial\lambda_{E'}^{(k)}}\sum_{E\in\partial T}\lambda_E^{(k)}=1,\quad E'\in\partial T. \tag{8.94}$$

Then, using the representation of $\psi_T^{(k)}$ in (8.91)

$$\begin{aligned}\frac{\partial}{\partial\lambda_{E'}^{(k)}}\sum_{E\in\partial T}\lambda_E^{(k)}&=\frac{\partial}{\partial\lambda_{E'}^{(k)}}F(F^{-1}(\sum_{E\in\partial T}\lambda_E^{(k)}))\\&=\frac{\partial}{\partial\psi_T^{(k)}}F(\psi_T^{(k)})\frac{\partial\psi_T^{(k)}}{\partial\lambda_{E'}^{(k)}},\end{aligned} \tag{8.95}$$

which implies that

$$\frac{\partial}{\partial\lambda_{E'}^{(k)}}\psi_T^{(k)}=\left(\frac{\partial}{\partial\psi_T^{(k)}}F(\psi_T^{(k)})\right)^{-1}. \tag{8.96}$$

The derivative of $F(\psi_T^{(k)})$ is given by

$$\frac{\partial}{\partial\psi_T^{(k)}}F(\psi_T^{(k)})=3+\frac{|T|}{\delta tK(\psi_T^{(k)})\beta_T}\left(\theta'(\psi_T^{(k)})-\frac{(\theta(\psi_T^{(k)})-\theta(\psi_T^{(k-1)}))}{K(\psi_T^{(k)})}K'(\psi_T^{(k)})\right). \tag{8.97}$$

Writing

$$g_E(\lambda^{(k)}) = \sum_{T\in\omega_E} K(\psi_T^{(k)}) \sum_{\tilde{E}\in\partial T} B^{-1}_{T,E\tilde{E}} \left(\psi_T^{(k)} - \lambda_{\tilde{E}}^{(k)} - z_{T,\tilde{E}}\right), \tag{8.98}$$

the entries in the Jacobian matrix

$$G'(\lambda^{(k)}) = \left[\frac{\partial g_E(\lambda^{(k)})}{\partial \lambda_{E'}^{(k)}}\right]_{E,E'\in\mathcal{E}_j\setminus\mathcal{E}_j^{(D)}}, \tag{8.99}$$

are then given by

$$\begin{aligned}
\frac{\partial g_E(\lambda^{(k)})}{\partial \lambda_{E'}^{(k)}} &= \sum_{T\in\omega_E\cap\omega_{E'}} K'(\psi_T^{(k)}) \frac{\partial \psi_T^{(k)}}{\partial \lambda_{E'}^{(k)}} \sum_{\tilde{E}\in\partial T} B^{-1}_{T,E\tilde{E}} \left(\psi_T^{(k)} - \lambda_{\tilde{E}}^{(k)} - z_{T,\tilde{E}}\right) + \\
&\quad \sum_{T\in\omega_E\cap\omega_{E'}} K(\psi_T^{(k)}) \sum_{\tilde{E}\in\partial T} B^{-1}_{T,E\tilde{E}} \left(\frac{\partial \psi_T^{(k)}}{\partial \lambda_{E'}^{(k)}} - \delta_{E'\tilde{E}}\right) \\
&= \sum_{T\in\omega_E\cap\omega_{E'}} \left(K'(\psi_T^{(k)}) \frac{\partial \psi_T^{(k)}}{\partial \lambda_{E'}^{(k)}} \sum_{\tilde{E}\in\partial T} B^{-1}_{T,E\tilde{E}} \left(\psi_T^{(k)} - \lambda_{\tilde{E}}^{(k)} - z_{T,\tilde{E}}\right) + \right. \\
&\quad \left. K(\psi_T^{(k)}) \left[-B^{-1}_{T,EE'} + \frac{\partial \psi_T^{(k)}}{\partial \lambda_{E'}^{(k)}} \sum_{\tilde{E}\in\partial T} B^{-1}_{T,E\tilde{E}} \right] \right) \\
&= \sum_{T\in\omega_E\cap\omega_{E'}} K(\psi_T^{(k)}) \left(- B^{-1}_{T,EE'} + \right. \\
&\quad \left. \left[\frac{K'(\psi_T^{(k)})}{K(\psi_T^{(k)})} \sum_{\tilde{E}\in\partial T} B^{-1}_{T,E\tilde{E}} (\psi_T^{(k)} - \lambda_{\tilde{E}}^{(k)} - z_{T,\tilde{E}}) + \beta_T \right] / \frac{\partial F(\psi_T^{(k)})}{\partial \psi_T} \right),
\end{aligned} \tag{8.100}$$

which concludes the discretisation of the mixed finite element formulation. To recap, the nonlinear system of equations to solve at each time step is given by

$$G(\lambda^{(k)}) = 0, \tag{8.101}$$

and the Newton iteration reads

$$\lambda^{(k,p+1)} = \lambda^{(k,p)} + \delta^{(k,p)} \tag{8.102}$$

where the correction $\delta^{(k,p)}$ is the solution of

$$G'(\lambda^{(k,p)})\delta^{(k,p)} = -G(\lambda^{(k,p)}). \tag{8.103}$$

The pressure head $\psi^{(k)}$ and the flux $\vec{q}^{(k)}$ can be recovered from the Lagrange multipliers using (8.91) and (8.88). Note that the pressure is required to be calculated whenever the residual for the Lagrange multipliers is calculated. Hence, every time the nonlinear residual is calculated, the solution of nonlinear equation (8.91) is required on every element, which is an expensive operation. It is clear that the number of times this operation is performed should be minimised in an efficient solution algorithm. It is therefore expected that the Newton-MG iteration will be much more efficient (per V-cycle) than the FAS iteration, as only a single residual calculation is required. Results in Subsection 8.3.3 demonstrate this behaviour.

Before presenting results for the mixed finite element formulation of Richards equation a description of the multigrid method used in the solution must be given. The method given so far is valid for piecewise linear continuous finite elements on triangles, but not for the space of Lagrange multipliers on edges of the mesh. In this case the grid transfer operators and the coarse grid operator must be changed in order to ensure that the method works correctly.

8.3.2 Multigrid for the Mixed Formulation

For previous problems a piecewise linear continuous basis has been used, and the space of piecewise linear continuous functions was required to be transferred between grids. The transfer operator introduced in Subsection 4.1.2 is no longer valid for the space of Lagrange multipliers on the edges of the elements. Therefore, before a multigrid method can be implemented, appropriate grid transfer operators need to be defined, as well as coarse grid operators. The subject of this subsection is the presentation of these grid transfer and coarse grid operators.

Let a hierarchy of nested grids Ω_j, $j = 1, \ldots, J$ be defined (see (4.2)). With each grid we associate the space M_j (see Equation (8.71)) of functions that are piecewise constant on the edges of Ω_j, $j = 1, \ldots, J$. We only have information regarding the functions on the edges, so that the transfer function used for the P_1 functions over entire elements is no longer valid, and a different transfer operator is required. A detailed discussion of the transfer operators used can be found in [25], and a summary of the operators follows. Our presentation is simplified by considering only the case that a regular refinement of coarse

grid triangles of equal size is used, in a two-dimensional setting.

The grid transfer operator used here is designed for nonconforming finite element methods where the functions are defined on each element, rather than just the edges. However, in the linear case, the equivalence of non-conforming and mixed finite element methods condensed with Lagrange multipliers is known [50]. Therefore, the transfer operator for the non-conforming method [25] is applicable also in the case of the Lagrange multipliers. This means that the transfer operator is valid when used with the linear multigrid method applied to the inner iteration of a Newton method. To the best of our knowledge there exists no literature detailing whether the transfer functions are valid also for nonlinear problems, but we assume that the transfer operator will also be valid for the nonlinear FAS. Results in Subsection 8.3.3 suggest that the transfer operator remains valid for FAS.

Let

$$P_{j-1}^{j} : M_{j-1} \to M_j$$

be the transfer operator from a coarse to a fine space. As in the case of the linear interpolation operator derived in Subsection 4.1.2 the transfer operator should be a projection in the L^2 inner product. A mesh-dependent L^2 inner product on $M_j \oplus M_{j-1}$ is defined by [25]

$$(u, v)_j = \sum_{T \in \mathcal{T}_j} Q_T(uv), \tag{8.104}$$

where Q_T is the second order accurate Gaussian quadrature on a triangle [54] given by

$$\int_T u \, \mathrm{d}\vec{x} \approx Q_T(u) = \frac{|T|}{3} \sum_{i=1}^{3} u(\vec{m}_i), \tag{8.105}$$

for $\vec{m}_i$ the mid-points of the edges on triangle T. In two-dimensions, if u and v are linear on each element, (8.104) coincides with the L^2 norm. The condition that P_{j-1}^{j} should satisfy is

$$(P_{j-1}^{j} v_{j-1}, w_j)_j = (v_{j-1}, w_j)_j, \quad \forall v_{j-1} \in M_{j-1}, w_j \in M_j. \tag{8.106}$$

For a fine grid edge that is not also on the coarse grid (see point A in Figure 8.7) the

transfer operator simply gives an average of neighbouring coarse grid edges, *i.e.* for edge A in Figure 8.7

$$P_{j-1}^{j} v_{j-1}(\vec{x}_A) = \frac{1}{2}(v_{j-1}(\vec{x}_1) + v_{j-1}(\vec{x}_2)). \tag{8.107}$$

For a fine grid edge that lies on a coarse grid edge (see edge B in Figure 8.7) the transfer operator is defined as

$$P_{j-1}^{j} v_{j-1}(\vec{x}_B) = v_{j-1}(\vec{x}_1) + \frac{1}{4}(v_{j-1}(\vec{x}_2) - v_{j-1}(\vec{x}_3) + v_{j-1}(\vec{x}_4) - v_{j-1}(\vec{x}_5)), \tag{8.108}$$

for elements of equal size.

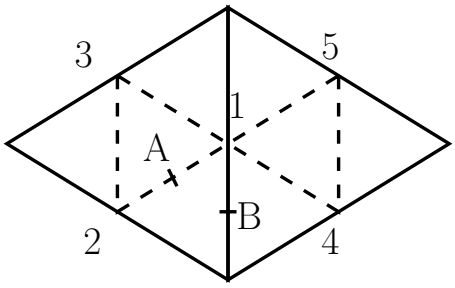


Figure 8.7: Numbers represent coarse-grid edges, and letters represent fine-grid edges

The restriction R_j^{j-1} is taken as the transpose of P_{j-1}^{j} with respect to $(\,\cdot\,,\,\cdot\,)_j$, *i.e.*

$$(P_{j-1}^{j} v_{j-1}, w_j)_j = (v_{j-1}, R_j^{j-1} w_j)_j. \tag{8.109}$$

The calculation of this is simple but tedious. There is only one case to consider, which is that of edge A in Figure 8.8. The action of the restriction operator at A is given by

$$\begin{aligned} R_j^{j-1} v_j(\vec{x}_A) = v_j(\vec{x}_1) + v_j(\vec{x}_2) + \frac{1}{2}\left\{v_j(\vec{x}_3) + v_j(\vec{x}_4) + v_j(\vec{x}_5) + v_j(\vec{x}_6)\right\} + \\ \frac{1}{4}\left\{v_j(\vec{x}_7) - v_j(\vec{x}_8) + v_j(\vec{x}_9) - v_j(\vec{x}_{10}) + v_j(\vec{x}_{11}) - v_j(\vec{x}_{12}) + v_j(\vec{x}_{13}) - v_j(\vec{x}_{14})\right\}. \end{aligned} \tag{8.110}$$

The coarse grid operators for the linear multigrid iteration are Galerkin coarse grid operators. Note that, for the grid transfer operators given above, the number of non-zeros per row in the coarse grid matrices will grow. To prevent this, any contribution from a non-zero entry that is outside of the sparsity pattern of the operator on the finest grid is

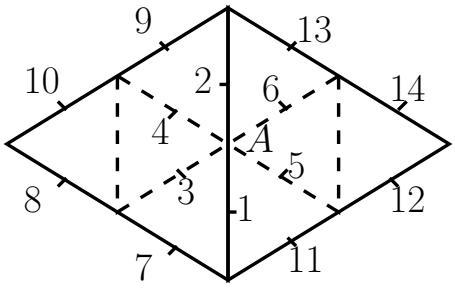


Figure 8.8: Numbers represent fine-grid edges and letters represent coarse grid edges

added to the diagonal of the current row in the matrix, as per [13]. For an FAS iteration the problem is re-discretised on each coarse grid, where the grid transfer functions are taken as the ones given above.

In the next subsection we compare a Newton-MG iteration with an FAS iteration for the mixed finite element formulation of the Richards equation. Although previous results have also considered a preconditioned GMRES iteration we do not do this here. This is because results are sufficient to show the superiority of a Newton type iteration over an FAS iteration without requiring a better solver. If a multigrid-preconditioned GMRES iteration were to be used, we expect to see the same advantages as for the previous problems (see Chapter 7 and Section 8.1).

The next subsection investigates the execution time per V-cycle of the algorithms as well as the robustness of the nonlinear multigrid methods with respect to an adaptive time step.

8.3.3 Execution Time and Robustness

The formulation of the Richards equation outlined above, whilst locally mass conservative, is much more challenging for nonlinear multigrid methods to solve. Previous research has not yet considered infiltration into dry soils [13]. The problem given in Equation (8.46) is too challenging for either nonlinear method to be able to solve due to the steepness of the wetting front entering into the soil. An unreasonably small timestep must be taken in order for the methods to converge. The problem we consider here takes the same form as in the previous section, apart that the pressure head in the bulk of the

soil is not as low, meaning that the infiltration front is less steep. Specifically we use

$$\begin{gathered}\frac{\partial}{\partial t}\theta(\psi) - \nabla\cdot(K(\psi)\nabla(\psi+z)) = 0, \quad \vec{x}\in\Omega = (0,100)\times(-100,0),\\ K(\psi)\nabla(\psi+z)\cdot\vec{\nu} = 0, \quad x = 0,100, \quad \text{or} \quad z = -100,\\ \psi = 0, \quad z = 0,\\ \psi = -100, \quad t = t_0, \quad \vec{x}\in\Omega.\end{gathered} \tag{8.111}$$

Initially we consider the infiltration of water into a single homogeneous soil. The soil defined on the domain is soil type A (Clay) from Table 8.7. As in previous sections we are interested in the execution time per V-cycle of each iteration. The theory from Chapter 6 is no longer valid for the current problem, as the analytic derivative, rather than a numerical approximation, is taken. As such we are left to measure the execution time empirically. However, from previous discussion it is expected that the execution time of the FAS iteration will be much larger than the execution time of the Newton-MG iteration.

Table 8.10 gives the time required to perform one-hundred nonlinear iterations for Newton-MG and FAS for varying numbers of linear inner iterations and nonlinear smoothing iterations. The finest grid in the multigrid hierarchy is 256^2 grid, and the coarsest is an 8^2 grid. Recall that p and ν in the first column heading stand for the the number of linear multigrid iterations performed per Newton iteration, and the number of pre- and post-smoothing iterations per FAS V-cycle, respectively. The dash in the final column indicates that the method did not converge. This is the case even for very small time steps, and is caused by the fact that a single nonlinear smooth on the fine grid is not sufficient to remove high frequency components from the error, so that the correction calculated on the coarse grid is inaccurate, regardless of the number of grids in the hierarchy. This is even the case for a two grid iteration. Increasing the number of nonlinear smoothing iterations gives a convergent FAS iteration, for which the execution times are given.

p/ν	Newton-MG	FAS
1	16.74	-
2	20.07	72.97
3	22.69	91.01
4	25.07	110.85

Table 8.10: Time required (s) to perform 100 nonlinear iterations (Newton or FAS) on a 256^2 grid for the mixed finite element formulation of the Richards equation.

It is clear from Table 8.10 that the Newton iteration is *much* quicker than the FAS for this problem. The framework in Chapter 6 shows that for a more complex nonlinear

problem the relative efficiency of the Newton iteration compared to the FAS iteration will increase, since the work involved in the linear multigrid iteration will be small compared to a more expensive nonlinear residual calculation. We again compare the execution time of a single V-cycle when three linear multigrid iterations are performed as part of the Newton iteration, and three pre- and post-smoothing iterations are performed as part of the FAS iteration. Using the values from Table 8.10 the measured ratio is

$$\frac{91.01}{100} \times \frac{300}{22.69} \approx 12.0 \tag{8.112}$$

per V-cycle, which is considerably higher than for any of the previous problems considered (see Chapter 7 and Subsections 8.1.3 and 8.2.3). Results later in this subsection suggest that, for the current problem, 4 pre- and post-smoothing iterations give an improved performance of an FAS iteration. Three linear iterations per Newton iteration still give a good convergence of the Newton iteration. In this case the ratio of execution time per V-cycle is given by

$$\frac{110.85}{100} \times \frac{300}{22.69} \approx 14.7. \tag{8.113}$$

The convergence of an FAS iteration must be much faster than for a Newton-MG iteration in order for the method to be more computationally efficient. In order to assess this we consider the growth of execution time with respect to decreasing mesh spacing, and also the robustness with respect to a time step parameter.

Figure 8.9 gives the growth in execution time for the iterations when a fixed time step is used, which is scaled with the square of the mesh spacing. The results are summarised in Table 8.11. The results show that the growth of the execution time of a Newton-MG iteration scales linearly with the decrease in time step size and quadratically with the decrease in mesh spacing. Whereas for previous problems the qualitative behaviour of the FAS iteration was the same, in this case we see a super-linear growth in execution time.

Using 3 pre- and post-smooths gives a much more pronounced deterioration in the execution time, and the average number of V-cycles required per time step increases dramatically. No timing is shown for the 512^2 grid, as the number of V-cycles required per iteration grew to > 40. The iteration was stopped after 123 time steps, at which point 3768 V-cycles had been performed in ~ 13200 seconds. Extrapolating, the execution time could be expected in the order of 27000s, which would be ~ 34 times slower than

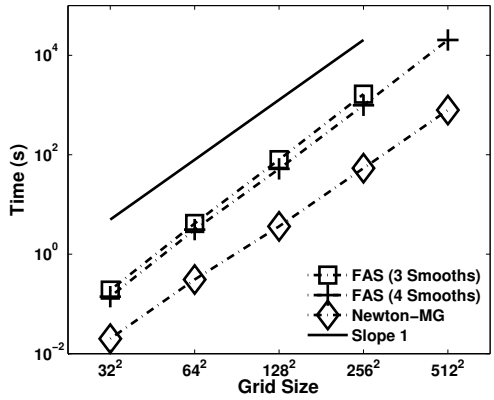


Figure 8.9: Growth of execution time as grid spacing h is scaled successively by a factor $1/2$, and the time step $\delta t \propto h^2$.

using a Newton-MG iteration. The growth in execution time is not so pronounced when using 4 pre- and post-smooths, although a super-linear increase in time is still observed (see Table 8.11). On the 512^2 grid the iteration requires a factor of

$$\frac{20341.22}{791.52} \approx 25.7 \tag{8.114}$$

times longer to execute. Hence, for a fixed time step, a Newton-MG iteration is much superior to the FAS iteration. Note that inspection of Table 8.11 shows that more than three times as many nonlinear V-cycles are performed for the FAS iteration than for the Newton iteration. Using the ratio of execution times per V-cycle given in (8.113) would suggest that the ratio in (8.114) should be ~ 45. However, the ratio (8.113) only considers the execution time of the solver. There is extra computational time required in setting up the time dependent problem, which is common to both iterations. Hence the increase in performance is not as large as (8.113) suggests.

		Newton-MG		FAS (3 Smooths)		FAS (4 Smooths)	
Grid Size	Time Steps	Time	V-Cycles	Time	V-Cycles	Time	V-Cycles
32^2	1	0.02	12	0.19	8	0.14	5
64^2	4	0.31	52	4.14	56	3.12	32
128^2	16	3.62	111	79.72	282	51.93	152
256^2	64	53.63	399	1658.56	1658	991.01	841
512^2	256	791.52	1392	-	-	20341.22	4641

Table 8.11: Number of V-cycles required to reach a target time ($t = 4\text{e-}4$) using a fixed time step scaled with the square of the mesh spacing h^2.

We now turn our attention to the robustness of the methods with respect to a time step

parameter by considering an adaptive time stepping strategy. In order to make the problem more challenging multiple soils are defined on the domain, with the same arrangement as in Figure 8.3. The properties for the two soils is given in Table 8.7. The adaptive time-stepping strategy is the same as outlined in Algorithm 8.1. The same parameters in the adaptive time stepping are used as given in Table 8.8, except that for Newton-MG the number of iterations below which the time step is increased is given by $\eta_s = 5$ and the number of iterations above which the time step is decreased is given by $\eta_l = 12$. For the FAS iteration $\eta_s = 4$ and $\eta_l = 10$. The maximum number of iterations to perform per time step is kept at $\eta_{\max} = 20$. A target time of 0.5 days is set for the problem given in (8.111), with initial time step 1e-6, and minimum time step 1e-6.

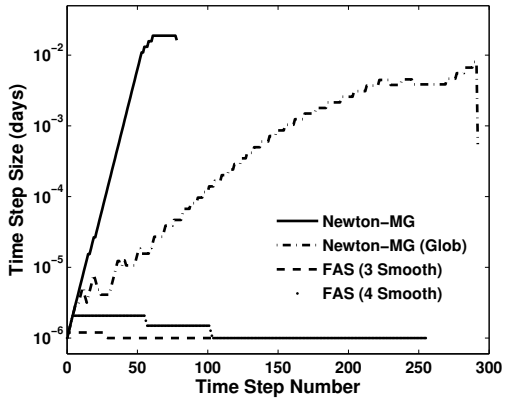


Figure 8.10: Progression of time step size for different nonlinear multigrid methods for the mixed finite element formulation of the Richards equation (8.111).

Figure 8.10 shows the progression of the size of the time step for each of the iterations. The advantage of the use of a Newton iteration is immediately apparent. The size of the time step for the Newton-MG iterations is much ($\sim$ 1e4 times) larger than that for the FAS iterations. Note that the FAS iterations fail to converge within 20 iterations even for the smallest time step at a very small time, so the execution is terminated. Increasing the number of smooths performed per FAS V-cycle increases the robustness of the method, but the iteration still fails to converge within the maximum number of iterations even at the minimum time step. This could be due to the fact that the linearisation is not symmetric in the unsaturated zone. Results from the previous section (see Subsection 8.2.3) show that an advantage can be obtained in the performance of the FAS iteration when an approximate linearisation is taken. However, the increase in performance was small (see Figure 8.5b). A substantial increase in the robustness is required for the method to become competitive with the Newton-MG iteration, which likely will not occur by changing the linearisation.

In contrast to the FAS, the Newton-MG permits a large time-step, and a Newton-MG iteration reaches the target time of 0.5 days in a small number of time steps. Interestingly, we see that the global Newton iteration performs worse for this problem than the standard implementation. This is explained similarly to the previous section, whereby the non-global method gives a correction at the first Newton iteration which increases the nonlinear residual, but the convergence for subsequent iterations is very good. For the stabilised method small corrections are made in order to keep the convergence of the Newton iteration monotonic. In this case the overall convergence is worse, as the corrections calculated require some time to move the current approximation into a regime of rapid convergence, as observed for the uncorrected method. This demonstrates that the choice of globalisation method may be improved upon.

We note that the results in this section are given solely to compare the execution times and robustness of FAS and Newton iterations. The detail is sufficient to demonstrate the superiority of the Newton-MG iteration over the FAS iteration in the case of a complex nonlinear problem, and for a discretisation method other than the continuous piecewise linear basis. As expected, the Newton-MG iteration performs better than the FAS iteration. This is in terms of execution time as well as robustness with respect to a time step parameter. In particular the results are in line with the prediction that the Newton iteration will perform more favourably in comparison to an FAS iteration the more complex a nonlinear problem becomes. This does not mean that the Newton-MG iteration is a good solver for the mixed formulation of the Richards equation. Newton's iteration will also fail to converge when the target time in Figure 8.10 is increased to 1 day. From inspection of the approximation this appears to be due to the fact that the wetting front propagating through a fine soil (type A in Figure 8.3) from above meets a faster moving steep wetting front moving through soil B (see Figure 8.3) moving from the right. This creates a curved wetting front, and at some point the front has a high curvature. It seems as though the meshes used are not apt at capturing this high curvature, and so convergence is not obtained. In order for the method to be useful from a soil physics perspective some adaptive mesh refinement, as in [13], could be used to ensure that fine level detail is resolved accurately. Improving the solution procedure by introduction of an adaptive mesh refinement procedure, as well as an improved Newton globalisation procedure, and improved inner linear iteration (for example preconditioned GMRES) would be an interesting investigation for future work.

8.4 Summary

In this chapter results have been presented to demonstrate that, for a time-dependent nonlinear parabolic operator with nonlinearity in the diffusion coefficient (where the nonlinearity depends on the solution function rather than its gradient), the Newton-MG iteration is again preferable to the FAS iteration. In Section 8.1 a problem was presented in which, for small time steps, the FAS iteration is competitive in terms of computational efficiency when compared to Newton-MG iteration. However, when considering robustness with respect to a time-step parameter it was shown that a Newton iteration may easily be made more robust by implementing a preconditioned GMRES iteration, at very little extra computational cost.

A more complicated problem was considered in Sections 8.2 and 8.3 – the Richards equation. In Section 8.2 the Richards equation is given as a single equation for a single unknown function. For this more complicated nonlinear problem it was found that there was a relative speed-up of a Newton-MG iteration compared to an FAS iteration, as predicted using the framework derived in Chapter 6. As well as being more computationally efficient a Newton iteration is much more robust with respect to a time-step parameter. This is demonstrated through the use of an adaptive time-stepping procedure, where a larger robust time-step is selected for an inexact Newton iteration compared to an FAS iteration. An advantage is also observed in the use of a global iteration for Newton's method and/or the use of a preconditioned GMRES iteration as an inner iteration. No stabilisation method is known for the FAS iteration. As in Section 7.7, it is found that a Newton iteration is better able to deal with a discontinuity in material properties on the domain in Subsection 8.2.3.

Finally, in Section 8.3, a more complicated discretisation of the Richards equation is presented, in which a small system of equations is solved. In this example the advantage of Newton-MG is made very clear. The execution time is over an order of magnitude faster for a Newton-MG iteration than for an FAS iteration, even at small time scales, when a homogeneous soil type is used. When using heterogeneous soils on the domain the FAS iteration is found to be unable to converge even for small time steps (see Subsection 8.3.3). The Newton iteration, on the other hand, is much more robust in the case of heterogeneous soils, and an adaptive time-stepping procedure finds a robust time step four orders of magnitude larger for Newton-MG than for FAS.

Chapter 9

Conclusions and Future Work

This chapter briefly summarises the results and conclusions found in Chapters 6 to 8, and gives a discussion of possible future work. The results and discussion from each of the chapters can be summarised in the following sentence:

> In a finite element setting, given sufficient computational resources, Newton-MG is a more robust and more efficient (*i.e*. superior) algorithm to FAS.

As is suggested in the above comment, sufficient computational resources need to be available for Newton's method to be feasible. The main advantage of an FAS iteration over a Newton iteration is that the large Jacobian matrix does not need to be stored, and so the memory requirements are smaller. Hence, for a very large problem an FAS iteration may be feasible when a Newton iteration is not. Except for this (large) advantage, results suggest that Newton-MG will be a superior algorithm in terms of:

(i) *Computational efficiency*: In Chapter 6 the execution time of the iterations per V-cycle is estimated through the development of a novel framework which allows a direct comparison of Newton-MG and FAS methods. The predictions gained suggest that a Newton-MG iteration will be more efficient per V-cycle relative to an FAS iteration, and that this relative efficiency is likely to increase the more complex a nonlinear problem is. Results are given in Chapters 7 and 8 supporting these statements. When applied to a model problem, in the best case an FAS iteration is comparable to Newton-MG in terms of running time, and in the worst case is a factor of 25-30 times slower (see Subsection 8.3.3).

(ii) *Robustness*: Chapter 7 shows the robustness of the iterations applied to a non-time-dependent elliptic problem. For approximations close to the exact discrete solution both methods converge in a predictable manner, with convergence of a Newton-MG iteration being superior to that of an FAS iteration. Further from the solution, robust convergence is more likely to be observed for a Newton iteration. In contrast, the convergence of an FAS iteration is difficult to predict for approximations far from the exact discrete solution, and no (general) stabilisation techniques are known for the FAS iteration. Robustness is also shown with respect to a discontinuous coefficient on the domain, where the convergence of the Newton iteration can be predicted with the use of linear multigrid convergence theory. Robust convergence is observed even in the case that the coefficient is not resolved on coarse grid levels (see Section 7.7) for a Newton-MG iteration, whereas the rate of convergence deteriorates rapidly for an FAS iteration in such cases. Chapter 8 investigates the robustness of the methods applied to time-dependent problems. The problems considered include a non-symmetric term in the linearisation, and it is shown that this has a negative impact of the convergence of both Newton-MG and FAS iterations. This can be justified using theory for linear multigrid iterations in the case of a Newton-MG iteration, but there is no theory to justify why this should be the case for FAS. Using heuristics from the linear theory, it is predicted that the use of a linear iterative solver for non-symmetric systems (in this case a multigrid preconditioned GMRES iteration) will increase the robustness of a Newton iteration, which is observed. No variant of an FAS iteration is known to stabilise the convergence in the case that the linearisation also has non-symmetric components. Theoretical discussion in Subsection 4.2.3 and results in Section 7.6 support a supposition that the asymptotic convergence of an FAS iteration depends on the convergence of a Newton iteration.

(iii) *Flexibility*: Globalisation techniques and the use of varied linear iterative solvers as part of the Newton iteration are investigated in Sections 7.6 and 7.7 and Chapter 8. These modifications of the Newton iteration lead to a more robust nonlinear iteration, and improved convergence results. The modifications are simple to apply, and are well researched. In contrast, there are no general stabilisation or globalisation techniques known for an FAS iteration. Discussion in Chapter 6 mentions the large additional cost in the FAS iteration using a known nonlinear smoother other than the pointwise nonlinear Jacobi iteration. This demonstrates that an FAS iteration is inflexible, and unless the standard implementation works well, it is not known how the method should be improved in an efficient manner.

The extensive results presented allow for us to confidently say that a Newton-MG iteration is better than an FAS iteration in a finite element setting. The results presented here are not exhaustive, but are varied enough to be able to be confident that the iterations will display similar convergence behaviours for other problems. Whilst for completeness it would be useful to have more results to strengthen the conclusion even further, this would not constitute interesting further work. The work in this thesis can be considered self-contained, and direct extension for similar PDEs would not add much qualitative information. However, the results give a good basis for a project involving the efficient solution of some complex nonlinear discrete PDEs. For example, as noted in Chapter 8, the solution of the Richards equation in mixed form (see Subsection 8.3.1) is an interesting problem which is receiving recent attention. It is found that although a Newton iteration is superior to an FAS iteration for the solution of the mixed finite element formulation that the solution method is substandard. In the future it would be interesting to make a Newton iteration more efficient and more robust for the mixed formulation of the Richards equation. This would be done through the use of adaptive mesh refinement in conjunction with the use of a linear iteration for non-symmetric systems in the inner iteration of a Newton iteration. In particular, results presented here show that it would be interesting to use a multigrid-preconditioned GMRES iteration in which the symmetric part of the Jacobian matrix is used as a preconditioner. It is expected that an improvement in the robustness of the method would be brought about by using the preconditioned GMRES iteration, and the use of an adaptively refined mesh would allow for fine level detail to be captured, where necessary.

Bibliography

[1] E.L. Allgower, K. Böhmer, A. Potra, and W.C. Rheinboldt. A Mesh-Independence Principle for Operator Equations and Their Discretizations. *SIAM Journal on Numerical Analysis*, 23(1):160–169, 1986.

[2] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' Guide*. SIAM Publishing, 1999.

[3] L. Armijo. Minimization of functions having Lipschitz-continuous first partial derivatives. *Pacific Journal of Mathematics*, 204:126–136, 1966.

[4] D.N. Arnold, R.S. Falk, and R. Winther. Preconditioning in $H(\mathrm{div})$ and Applications. *Mathematics of Computation*, 66(219):957–984, July 1997.

[5] Helenbrook. B., D. Mavriplis, and H. Atkins. Analysis of "p"-multigrid for continuous and discontinuous finite element discretizations. In *16th AIAA Copmutational Fluid Dynamics Conference*, June 2003.

[6] M.J. Baines, M.E. Hubbard, P.J. Jimack, and A.C. Jones. Scale-Invariant Moving Finite Elements for Nonlinear Partial Differential Equations in Two Dimensions. *Applied Numerical Mathematics*, 56:230–252, 2006.

[7] N.S. Bakhvalov. On the convergence of a relaxation method with natural constraints on the elliptic operator. *USSR Computational Mathematics and Mathematical Physics*, 6(5):101–135, 1966.

[8] R. Bank and C. Douglas. Sparse Matrix Multiplication Package (SMMP). *Advances in Computational Mathematics*, 1(1):127–137, 1993.

[9] R. Bank, T. DuPont, and H. Yserentant. The hierarchical basis multigrid method. *Numerische Mathematik*, 42:427–458, 1988.

[10] R.E. Bank and C.C. Douglas. Sharp Estimates for Multigrid Rates of Convergence with General Smoothing and Acceleration. *SIAM Journal on Numerical Analysis*, 22(4):617–633, August 1985.

[11] G.I. Barenblatt. *Scaling, self-similarity, and intermediate asymptotics: dimensional analysis and intermediate asymptotics*, volume 14 of *Cambridge Texts in Applied Mathematics*. Cambridge University Press, 1996.

[12] J.W. Barrett and W.B. Liu. Finite Element Approximation of the p-Laplacian. *Mathematics of Computation*, 61(204):523–537, 1993.

[13] M. Bause and P. Knabner. Computation of variably saturated subsurface flow by adaptive mixed hybrid finite element methods. *Advances in Water Resources*, 27:565–581, 2004.

[14] M. Benzi, M.A. Olshanskii, and Z. Wang. Modified augmented Lagrangian preconditioners for the incompressible Navier-Stokes equations. *International Journal for Numerical Methods in Fluids*, 66(4):486–508, 2011.

[15] M.S. Berger. *Nonlinearity and Functional Analysis*. Pure and Applied Mathematics. Academic Press, 1977.

[16] M. Bern, D. Eppstein, and J. Gilbert. Provably Good Mesh Generation. *Journal of Computer and Systems Sciences*, 48:384–409, 1994.

[17] J. Bey and G. Wittum. Downwind numbering: Robust multigrid for convection-diffusion problems. *Applied Numerical Mathematics*, 23(1):177–192, 1997.

[18] M.L. Bittencourt, C.C. Douglas, and R.A. Feijoo. Adaptive non-nested multigrid methods. *Engineering Computations*, 19(1-2):158–176, 2002.

[19] L.S. Blackford, J. Choi, A. Cleary, E. D'Azevedoo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R.C. Whaler. *ScaLAPACK Users' Guide*. SIAM Publications, 1997.

[20] F.A. Bornemann and P. Deuflhard. The cascadic multigrid method for elliptic problems. *Numerische Mathematik*, 75(2):135–152, 1996.

[21] E.T. Bouloutas. *Improved numerical methods for modeling flow and transport processes in partially saturated porous media*. PhD thesis, Massachusetts Institute of Technology, Dept. of Civil Engineering, 1989.

[22] K.J. Brabazon, M.E. Hubbard, and P.K. Jimack. Nonlinear Multigrid Methods for Second Order Differential Operators with Nonlinear Diffusion Coefficient. *Computers & Mathematics with Applications*, 68:1619–1634, 2014.

[23] D. Braess. *Finite Elements: Theory, fast solvers, and applications in solid mechanics*. Cambridge University Press, second edition, 2001.

[24] D. Braess and W. Hackbusch. A New Convergence Proof for the Multigrid Method Including the V-Cycle. *SIAM Journal on Numerical Analysis*, 20(5):967–975, October 1983.

[25] D. Braess and R. Verfürth. Multigrid methods for nonconforming finite element methods. *SIAM Journal on Numerical Analysis*, 27:979–986, 1990.

[26] J.H. Bramble, D.Y. Kwak, and J.E. Pasciak. Uniform Convergence of Multigrid V-Cycle Iterations for Indefinite and Nonsymmetric Problems. *SIAM Journal on Numerical Analysis*, 31(6):1746–1763, December 1994.

[27] J.H. Bramble, J. Pasciak, and J. Xu. The Analysis of Multigrid Algorithms with Nonnested Spaces or Noninherited Quadratic Forms. *Mathematics of Computation*, 56:1–34, 1991.

[28] J.H. Bramble and J.E. Pasciak. New Convergence Estimates for Multigrid Algorithms. *Mathematics of Computation*, 49(180):311–329, Oct. 1987.

[29] J.H. Bramble and J.E. Pasciak. New Estimates for Multilevel Algorithms Including the V-cycle. *Mathematics of Computation*, 60(202):447–471, April 1993.

[30] J.H. Bramble, J.E. Pasciak, J. Wang, and J. Xu. Convergence Estimates for Multigrid Algorithms without Regularity Assumptions. *Mathematics of Computation*, 57(195):23–45, July 1991.

[31] J.H. Bramble, J.E. Pasciak, J. Wang, and J. Xu. Convergence Estimates for Product Iterative Methods with Applications to Domain Decomposition. *Mathematics of Computation*, 57(195):1–21, July 1991.

[32] A. Brandt. Multilevel Adaptive Solutions to Boundary Value Problems. *Mathematics of Computation*, 31(138):333–390, 1977.

[33] A. Brandt. Rigorous local mode analysis of multigrid. In *Preliminary Proc. 4th Copper Mountain Conf. on Multigrid Methods*, Copper Mountain, Colorado, April 1989.

[34] A. Brandt. Rigorous Quantitative Analysis of Multigrid, I: Constant Coefficients Two-Level Cycle with L_2-Norm. *SIAM Journal on Numerical Analysis*, 31(6):1695–1730, 1994.

[35] J. Brannick, M. Brezina, R. Falgout, T. Manteuffel, S. McCormick, J. Ruge, B. Sheehan, J. Xu, and L. Zikatanov. Extending the applicability of multigrid methods. *Journal of Physics: Conference Series*, 46:443–452, 2006.

[36] S.C. Brenner. Convergence of nonconforming multigrid methods without full elliptic regularity. *Mathematics of Computation*, 68:25–53, 1999.

[37] S.C. Brenner. Convergence of the multigrid V-cycle algorithm for second-order boundary value problems without full elliptic regularity. *Mathematics of Computation*, 71:507–525, 2002.

[38] S.C. Brenner. Convergence of Nonconforming V-Cycle and F-Cycle Multigrid Algorithms for Second Order Elliptic Boundary Value Problems. *Mathematics of Computation*, 73:1041–1066, 2003.

[39] S.C. Brenner and L.R. Scott. *The Mathematical Theory of Finite Element Methods*. Number 15 in Texts in Applied Mathematics. Springer, 3rd edition, 2008.

[40] M. Brezina, A. Doostan, T. Manteuffel, S. McCormick, and J. Ruge. Smoothed aggregation algebraic multigrid for stochastic PDE problems with layered materials. *Numerical Linear Algebra with Applications*, 21(2):239–255, 2014.

[41] F. Brezzi and M. Fortin. *Mixed and Hybrid Finite Element Methods*. Springer Series in Computational Mathematics. Springer-Verlag, 1991.

[42] W.L. Briggs, V.E. Henson, and S.F. McCormick. *A Multigrid Tutorial*. Society for Industrial and Applied Mathematics, 2nd edition, 2000.

[43] R.H. Brooks and A.T. Corey. *Hydraulic Properties of Porous Media*. Number 3 in Colorado State University Hydrology Papers. Colorado State University, 1964.

[44] J. Brown, B. Smith, and A. Ahmadia. Achieving Textbook Multigrid Efficiency for Hydrostatic Ice Sheet Flow. *SIAM Journal on Scientific Computing*, 35(2):B359–B375, 2013.

[45] P.N. Brown, P.S. Vassilevski, and C.S. Woodward. On Mesh-Independent Convergence of an Inexact Newton-Multigrid Algorithm. *SIAM Journal on Scientific Computing*, 25(2):570–590, 2003.

[46] R.L. Burden and J.D. Faires. *Numerical Analysis*. Thomson Brooks/Cole, 8th edition, 2005.

[47] C. Canuto, M.Y. Hussaini, A. Quarteroni, and T.A. Zang. *Spectral Methods - Fundamentals in Single Domains*. Springer Series in Scientific Computation. Springer, 2006.

[48] M.A. Celia, E.T. Bouloutas, and R.L. Zarba. A General Mass-Conservative Numerical Solution for the Unsaturated Flow Equation. *Water Resources Research*, 26(7):1483–1496, July 1990.

[49] J. Chen. Multigrid method and multilevel additive preconditioner for mixed element method for non-self-adjoint and indefinite problems. *Applied Mathematics and Computation*, 119:229–247, 2001.

[50] Z. Chen. Equivalence between and multigrid algorithms for nonconforming and mixed methods for second-order elliptic problems. *Journal of Numerical Mathematics*, 4:1–33, 1996.

[51] Z. Chen, D.Y. Kwak, and Y.J. Yon. Multigrid Algorithms for Nonconforming and Mixed Methods for Nonsymmetric and Indefinite Problems. *SIAM Journal on Scientific Computing*, 19:502–515, 1998.

[52] P.G. Ciarlet. *The Finite Element Method for Elliptic Problems*. North-Holland Publishing Company, 1978.

[53] R.D. Cook, D.S. Malkus, M.E. Plesha, and R.J. Witt. *Concepts and Applications of Finite Element Analysis*. John Wiley & Sons, 4 edition, 2002.

[54] G.R. Cowper. Gaussian quadrature formulas for triangles. *International Journal for Numerical Methods in Engineering*, 7(3):405–408, 1973.

[55] P. Daskalopoulos and R. Hamilton. Regularity of the free boundary for the porous medium equation. *Journal of the Mathematical Society*, 11:895–965, 1998.

[56] T.A. Davis. *Direct Methods for Sparse Linear Systems*. SIAM Publications, 2006.

[57] J.W. Demmel, J. Gilbert, and X.S. Li. SuperLU Users' Guide. Technical report, University of California at Berkeley, 1997.

[58] P. Deuflhard. *Newton Methods for Nonlinear Problems: Affine Invariance and Adaptive Algorithms*. Number 35 in Springer Series in Computational Mathematics. Springer Berlin / Heidelberg, 2004.

[59] P. Deuflhard. A Short History of Newton's Method. *Documenta Mathematica*, Extra Volume: Optimization Stories:25–30, 2012.

[60] T. Dickopf and R. Krause. A Study of Prolongation Operators Between Non-nested Meshes. In Y.Q. Huang, R. Kornhuber, O. Widlund, and J.C. Xu, editors, *Domain Decomposition Methods in Science and Engineering XIX*, volume 78 of *Lecture Notes in Computational Science and Engineering*, pages 343–350, 2011.

[61] L. Diening and C. Kreuzer. Linear Convergence of an Adaptive Finite Element Method for the p-Laplacian Equation. *SIAM Journal on Numerical Analysis*, 46(2):614–638, 2008.

[62] F. Dobrian and A. Pothen. Oblio: A Sparse Direct Solver Library for Serial and Parallel Computations. Technical report, 2000.

[63] M. Donatelli, M. Semplice, and S. Serra-Capizzano. Analysis of Multigrid Preconditioning for Implicit PDE Solvers for Degenerate Parabolic Equations. *SIAM Journal on Matrix Analysis and Applications*, 32(4):1125–1148, 2011.

[64] M. Donatelli, M. Semplice, and S. Serra-Capizzano. AMG preconditioning for nonlinear degenerate parabolic equations on nonuniform grids with application to monument degradation. *Applied Numerical Mathematics*, 68:1–18, 2013.

[65] V. Druskin and M. Zaslavsky. On convergence of Krylov subspace approximation of time-invariant self-adjoint dynamical systems. *Linear Algebra and its Applications*, 436(10):3883–3903, 2012.

[66] M.A. Dumett, P. Vassilevski, and C.S. Woodward. A Multigrid Method for Nonlinear Unstructured Finite Element Elliptic Equations. Technical report, Lawrence Livermore National Laboratories, 2002.

[67] Y. Efendiev, J. Galvis, S. Ki Kang, and R.D. Lazarov. Robust Multiscale Iterative Solvers for Nonlinear Flows in Highly Heterogeneous Media. *Numerical Mathematics - Theory Methods and Applications*, 5(3):359–383, 2012.

[68] Y. Efendiev, J. Galvis, R. Lazarov, and J. Willems. Robust domain decomposition preconditioners for abstract symmetric positive definite bilinear forms. *ESAIM: Mathematical Modelling and Numerical Analysis*, 46(5):1175–1199, 2012.

[69] H. Elman, D. Silvester, and A. Wathen. *Finite Elements and Fast Iterative Solvers with applications in incompressible fluid dynamics*. Oxford University Press, 2005.

[70] Y.A. Erlangga and R. Nabben. On a multilevel Krylov method for the Helmholtz equation preconditioned by shifted Laplacian. *Electronic Transactions on Numerical Analysis*, 31:403–424, 2008.

[71] L.C. Evans. *Partial Differential Equations*. American Mathematical Society, 1998.

[72] R. Fedorenko. A relaxing method for solving elliptic difference equations. *USSR Computational Mathematics and Mathematical Physics*, 1:1092–1096, 1961.

[73] B. Fornberg. Calculation of Weights in Finite Difference Formulas. *SIAM Review*, 40(3):685–691, 1998.

[74] P.H. Gaskell, P.K. Jimack, M. Sellier, and H.M. Thompson. Efficient and accurate time-adaptive multigrid simulation of droplet spreading. *International Journal for Numerical Methods in Fluids*, 45(11):1161–1186, 2004.

[75] P.H. Gaskell, P.K. Jimack, M. Sellier, and H.M. Thompson. Flow of evaporating, gravity-driven thin liquid films over topography. *Physics of Fluids*, 18:013601, 2006.

[76] L. Ge and F. Sotiropoulos. A numerical method for solving the 3D unsteady incompressible Navier-Stokes equations in curvilinear domains with complex immersed boundaries. *Journal of Computational Physics*, 225:1782–1809, 2007.

[77] W.C. Gear. *Numerical Initial Value Problems in Ordinary Differential Equations*. Prentice-Hall, 1971.

[78] G.H. Golub and C.F. Van Loan. *Matrix Computations*. Johns Hopkins studies in the mathematical sciences. Johns Hopkins University Press, 3rd edition, 1996.

[79] G.H. Golub and C.F. Van Loan. *Matrix Computations*. The John Hopkins University Press, 4 edition, 2013.

[80] C. Gräser and R. Kornhuber. Multigrid Methods for Obstacle Problems. *Journal of Computational Mathematics*, 27(1):1–44, 2009.

[81] C. Gräser, U. Sack, and O. Sander. Truncated nonsmooth Newton multigrid methods for convex minimization problems. In M. Bercovier, M. Gander, R. Kornhuber, and O. Widlund, editors, *Domain Decomposition Methods in Science and Engineering XVIII*, volume 70 of *LNCSE*, pages 129–136. Springer, 2009.

[82] J.R. Green, P.K. Jimack, A.M. Mullis, and J. Rosam. An Adaptive, Multilevel Scheme for the Implicit Solution of Three-Dimensional Phase-Field Equations. *Numerical Methods for Partial Differential Equations*, 27:106–120, 2011.

[83] D.H. Griffel. *Applied Functional Analysis*. Dover Publications, 3rd edition, 2002.

[84] L.J. Guo, H. Huan, D.R. Gaston, C.J. Permann, D. Andrs, G.D. Redden, C. Lu, D.T. Fox, and Y. Fujita. A parallel, fully coupled, fully implicit solution to reactive transport in porous media using the preconditioned Jacobian-Free Newton-Krylov Method. *Advances in Water Resources*, 53:101–108, 2013.

[85] W. Hackbusch. *Multi-Grid Methods and Applications*, volume 4 of *Springer Series in Computational Mathematics*. Springer-Verlag, 1985.

[86] W. Hackbusch. Comparison of Different Multi-Grid Variants for Nonlinear Equations. *Zeitschrift für Angewandte Mathematik und Mechanic*, 72(2):148–151, 1992.

[87] W. Hackbusch and A. Reusken. On Global Multigrid Convergence for Nonlinear Problems. In W. Hackbusch, editor, *Robust Multi-Grid Methods*, Notes on Numerical Fluid Dynamics, pages 105–113. Vieweg, Braunschweig, 1988.

[88] W. Hackbusch and A. Reusken. Analysis of a Damped Nonlinear Multilevel Method. *Numerische Mathematik*, 55:225–246, 1989.

[89] W. Hackbusch and S. Sauter. A New Finite Element Approach for Problems Containing Small Geometric Detail. *Archivum Mathematicum (Brno)*, 34:105–117, 1998.

[90] M. Heil. An efficient solver for the fully coupled solution of large-displacement fluid-structure interaction problems. *Computer Methods in Applied Mechanics and Engineering*, 193(1-2):1–23, 2004.

[91] V.E. Henson. Multigrid Methods For Nonlinear Problems: An Overview. In *Proceedings of the Society of Photo-Optical Instrumentation Engineers (SPIE)*, pages 36–48, 2003.

[92] N. Higham. *Accuracy and Stability of Numerical Algorithms*. SIAM, 2nd edition, 2002.

[93] D. Hillel. *Introduction to Environmental Soil Physics*. Elsevier Academic Press, 2nd edition, 2004.

[94] T.K. Huckle and C. Kravvaritis. Compact Fourier Analysis for Multigrid Methods Based on Block Symbols. *SIAM Journal on Matrix Analysis and Applications*, 33(1):73–96, 2012.

[95] A. Iserles. *A First Course in the Numerical Analysis of Differential Equations*. Cambridge texts in applied mathematics. Cambridge University Press, 1996.

[96] R.W. Jeppson. Axisymmetric infiltration in soils. I. Numerical techniques for solution. *Journal of Hydrology*, 23:111–130, 1974.

[97] A.C. Jones. *A projected multigrid method for the solution of non-linear finite element problems on adaptively refined grids*. PhD thesis, University of Leeds, 2005.

[98] J.E. Jones and P. Vassilevski. AMGe based on element agglomeration. *SIAM Journal on Scientific Computing*, 23:109–133, 2001.

[99] J.E. Jones and C.S. Woodward. Newton-Krylov-Multigrid Solvers for Large-Scale, Highly Heterogeneous, Variably Saturated Flow Problems. *Advances in Water Resources*, 24(7):763–774, 2000.

[100] G. Jouvet and C. Gräser. An adaptive Newton multigrid method for a model of marine ice sheets. *Journal of Computational Physics*, 252:419–437, 2013.

[101] S.H. Ju and K.J.S. Kung. Mass types, element orders and solution schemes for the Richards equation. *Computers & Geosciences*, 23(2):175–187, 1997.

[102] G. Juncu, A Nicola, and C. Popa. Nonlinear Multigrid Methods for Numerical Solution of the Variably Saturated Flow Equation in Two Space Dimensions. *Transport in Porous Media*, 91:35–47, 2012.

[103] L. Kantorovich. On Newton's Method. *Trudy Matematicheskogo Instituta Imeni V. A. Steklova*, 28:104–144, 1949. (Russian).

[104] J. Karátson. Characterizing Mesh Independent Quadratic Convergence of Newton's Method for a Class of Elliptic Problems. *SIAM Journal on Mathematical Analysis*, 44(3):1279–1303, 2012.

[105] C.T. Kelley. *Iterative Methods for Linear and Nonlinear Equations*. SIAM, 1995.

[106] P. Kerfriden, P. Gosselet, S. Adhikari, and S.P.A. Bordas. Bridging proper orthogonal decomposition methods and augmented Newton-Krylov algorithms: An adaptive model order reduction for highly nonlinear mechanical problems. *Computer methods in applied mechanics and engineering*, 200(5-8):850–866, 2011.

[107] T. Kim, J.E. Pasciak, and P.S. Vassilevski. Mesh-independent convergence of the modified inexact Newton method for a second order non-linear problem. *Numerical Linear Algebra with Applications*, 13:23–47, 2006.

[108] D.A. Knoll and D.E. Keyes. Jacobian-free Newton-Krylov methods: a survey of approaches and applications. *Journal of Computational Physics*, 193:357–397, 2004.

[109] R. Kornhuber. Monotone multigrid methods for elliptic variational inequalities I. *Numerische Mathematik*, 96:167–184, 1994.

[110] R. Kornhuber. Monotone multigrid methods for elliptic variational inequalities II. *Numerische Mathematik*, 72:481–499, 1996.

[111] J. Kouatchou. Comparison of time and spatial collocation methods for the heat equation. *Journal of Computational and Applied Mathematics*, 150:129–141, 2003.

[112] J.D. Lambert. *Numerical Methods for Ordinary Differential Systems*. John Wiley & Sons, 1991.

[113] Y.C. Lee and P.H. Gaskell. Multi-level adaptive solution of anisotropic porous media flow with strong discontinuous jumps in permeability. *Journal of Mechanical Engineering Science*, 225(4):853–868, 2011.

[114] R.J. LeVeque. *Finite-Volume Methods for Hyperbolic Problems*. Cambridge Texts in Applied Mathematics. Cambridge University Press, 2002.

[115] X.Y. Li. Generalized block diagonal and block triangular preconditioners for non-symmetric indefinite linear systems. *Journal of Computational Analysis and Applications*, 17(3):445–458, 2014.

[116] F.R. Lin, S.W. Yang, and X.Q. Jin. Preconditioned iterative methods for fractional diffusion equation. *Journal of Computational Physics*, 256:109–117, 2014.

[117] S.P. MacLachlan and L.N. Olson. Theoretical bounds for algebraic multigrid performance: review and analysis. *Numerical Linear Algebra with Applications*, 21(2):194–220, 2014.

[118] S.P. MacLachlan and C.W. Oosterlee. Local Fourier analysis for multigrid with overlapping smoothers applied to systems of PDEs. *Numerical Linear Algebra with Applications*, 18(4):751–774, 2011.

[119] C.H. Marchi, L.K. Araki, A.C. Alves, R. Suero, F.F.T. Goncalves, and M.A.V. Pinto. Repeated Richardson extrapolation applied to the two-dimensional Laplace equation using triangular and square grids. *Applied Mathematical Modelling*, 37:4661–4675, 2013.

[120] D.J. Mavriplis. Multigrid Approaches to Non-Linear Diffusion Problems on Unstructured Meshes. *Numerical Linear Algebra with Applications*, 8(8):499–512, 2001.

[121] D.J. Mavriplis. An Assessment of Linear versus Non-Linear Multigrid Methods for Unstructured Mesh Solvers. *Journal of Computational Physics*, 175:302–325, 2002.

[122] R. Maxwell. A terrain-following grid transform and preconditioner for parallel, large-scale, integrated hydrologic modelling. *Advances in Water Resources*, 53:109–117, 2013.

[123] S.F. McCormick. Multigrid methods for variational problems: general theory for the V-cycle. *SIAM Journal on Numerical Analysis*, 22:634–643, 1985.

[124] S.A. Melchior, V. Legat, P. Van Dooren, and A.J. Wathen. Analysis of preconditioned iterative solvers for incompressible flow problems. *International Journal for Numerical Methods in Fluids*, 68(3):269–286, 2012.

[125] C.T. Miller, C.N. Dawson, M.W. Farthing, T.Y. Hou, J. Huang, C.E. Kees, C.T. Kelley, and H.P. Langtangen. Numerical simulation of water resources problems: Models, methods, and trends. *Advances in Water Resources*, 51:405–437, 2013.

[126] R.J. Millington and J.P. Quirk. Permeability of porous solids. *Transactions of the Faraday Society*, 57:1200–1206, 1961.

[127] J. Molenaar. Multigrid Methods for Fully Implicit Oil Reservoir Simulation. Technical report, TWI, Delft University of Technology, 1995.

[128] Y. Mualem. A New Model for Predicting the Hydraulic Conductivity of Unsaturated Porous Media. *Water Resources Research*, 12(3):513–522, June 1976.

[129] MUMPS: MUltifrontal Massively Parallel Solver. http://mumps.enseeiht.fr/, 1999–2011.

[130] A.A. Muresan and Y. Notay. Analysis of aggregation-based multigrid. *SIAM Journal on Scientific Computing*, 30:1082–1103, 2008.

[131] J.D. Murray. *Mathematical Biology: An Introduction*. Springer, 3rd edition, 2002.

[132] I. Mysovskikh. On convergence of Newton's Method. *Trudy Mat Inst Steklova*, 28:145–147, 1949. (Russian).

[133] A. Napov and Y. Notay. When does two-grid optimality carry over to the V-cycle? *Numerical Linear Algebra with Applications*, 17:273–290, 2010.

[134] A. Napov and Y. Notay. Smoothing factor, order of prolongation and actual multigrid convergence. *Numerische Mathematik*, 118:457–483, 2011.

[135] Nédélec. Mixed finite elements in $\mathbb{R}^3$. *Numerische Mathematik*, 35:315–341, 1980.

[136] Y. Notay. An aggregation-based algebraic multigrid method. *Electronic Transactions on Numerical Analysis*, 37:123–146, 2010.

[137] Y. Notay and P.S. Vassilevski. Recursive Krylov-based multigrid cycles. *Numerical Linear Algebra with Applications*, 15(5):473–487, 2008.

[138] M. Olshanskii and A. Reusken. Convergence analysis of a multigrid method for a convection-dominated model problem. *SIAM Journal on Numerical Analysis*, 42:1261–1291, 2004.

[139] J.M. Ortega and W.C. Rheinboldt. *Iterative Solution of Nonlinear Equations in Several Variables*. Computer Science and Applied Mathematics. Academic Press, 1970.

[140] P. Oswald. On function spaces related to finite element approximation theory. *Zeitschrift für Analysis und ihre Anwendungen*, 9:43–64, 1990.

[141] P. Oswald. On discrete norm estimates related to multilevel preconditioners in the finite element method. In *Proceedings of the International Conference on the Constructive Theory of Functions*, Varna, 1991.

[142] H. Park, D.A. Knoll, R.M. Rauenzahn, C.K. Newman, J.D. Densmore, and A.B. Wollaber. An Efficient and Time Accurate, Moment-Based Scale-Bridging Algorithm for Thermal Radiative Transfer Problems. *SIAM Journal on Scientific Computing*, 35(5):S18–S41, 2013.

[143] C. Pechstein and R. Scheichl. Analysis of FETI methods for multiscale PDEs - Part II: Interface variation. *Numerische Mathematik*, 118(3):485–529, 2011.

[144] C. Pechstein and R. Scheichl. Weighted Poincaré Inequalities and Applications in Domain Decomposition. In Y. Huang, R. Kornhuber, O. Widlund, and J. Xu, editors, *Domain Decomposition Methods in Science and Engineering XIX*, volume 78 of *Lecture Notes in Computational Science and Engineering*, pages 197–204. Springer Berlin / Heidelberg, 2011.

[145] F. Perini, E. Galligani, and R.D. Reitz. A study of direct and Krylov iterative sparse solver techniques to approach linear scaling of the integration of chemical kinetics with detailed combustion mechanisms. *Combustion and Flame*, 161(5):1180–1195, 2014.

[146] P.O. Persson and J. Peraire. Newton-GMRES Preconditioning for Discontinuous Galerkin Discretization of the Navier-Stokes Equations. *SIAM Journal on Scientific Computing*, 30(6):2709–2733, 2008.

[147] M.H. Protter and H.F. Weinberger. *Maximum principles in differential equations*. Prentice-Hall, 1967.

[148] P. A. Raviart and J. M. Thomas. A mixed finite element method for 2-nd order elliptic problems. In I. Galligani and E. Magenes, editors, *Mathematical aspects of the finite element method*, number 606 in Lecture Notes in Mathematics, pages 292–315. Springer, New York, 1977.

[149] A. Reusken. Convergence of the Multigrid Full Approximation Scheme for a Class of Elliptic Mildly Nonlinear Boundary Value Problems. *Numerische Mathematik*, 52:251–277, 1988.

[150] A. Reusken. Convergence of the Multilevel Full Approximation Scheme including the V-Cycle. *Numerische Mathematik*, 53:663–686, 1988.

[151] J. Rosam, P.K. Jimack, and A.M. Mullis. An Adaptive, Fully Implicit, Multigrid Phase-Field Model for the Quantitative Simulation of Non-isothermal Binary Alloy Solidification. *Acta Mathematica*, 56:4559–4569, 2008.

[152] B.P. Rynne and M.A. Youngson. *Linear Functional Analysis*. Springer Undergraduate Maths Series. Springer-Verlag, 2000.

[153] Y. Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, 2nd edition, 2003.

[154] D. Sármány, M.E. Hubbard, and M. Ricchiuto. Unconditionally stable space-time discontinuous residual distribution for shallow-water flows. *Journal of Computational Physics*, 253:86–113, 2013.

[155] R. Scheichl. *Iterative Solution of Saddle Point Problems Using Divergence-free Finite Elements with Applications to Groundwater Flow*. PhD thesis, University of Bath, 2002.

[156] R. Scheichl, P.S. Vassilevski, and L.T. Zikatanov. Multilevel Methods for Elliptic Problems with Highly Varying Coefficients on Non-aligned Coarse Grids. *SIAM Journal on Numerical Analysis*, 50(3):1675–1694, 2012.

[157] E. Schneid. *Hybrid-Gemischte Finite-Elemente-Diskretisierung der Richards-Gleichung*. PhD thesis, University of Erlangen-Nuremberg, 2000. (In German).

[158] M. Semplice. Preconditioned Implicit Solvers for Nonlinear PDEs in Monument Conservation. *SIAM Journal on Scientific Computing*, 32(5):3071–3091, 2010.

[159] V.P. Sergiievskyi, W. Hackbusch, and M.V. Fedorov. Multigrid Solver for the Reference Interaction Site Model of Molecular Liquids Theory. *Journal of Computational Chemistry*, 32(9):1982–1992, 2011.

[160] V.V. Shaidurov. *Multigrid Methods for Finite Elements*. Kluwer Academic Publishers, 2nd edition, 1995.

[161] J. Shewchuk. Delaunay refinement algorithms for triangular mesh generation. *Computational Geometry - Theory and Applications*, 22:21–74, 2002.

[162] J. R. Shewchuk. What Is a Good Linear Finite Element? Interpolation, Conditioning, Anisotropy and Quality Measures. (Preprint), December 2002.

[163] J. Shin, D. Jeong, and J. Kim. A conservative numerical method for the Cahn-Hilliard equation in complex domains. *Journal of Computational Physics*, 230:7441–7455, 2011.

[164] V. Simoncini and D.B. Szyld. Recent computational developments in Krylov subspace methods for linear systems. *Numerical Linear Algebra with Applications*, 14(1):1–59, 2007.

[165] B.F. Smith, P.E. Bjørstad, and W. Gropp. *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations*. Cambridge University Press, 1996.

[166] G.D. Smith. *Numerical Solutions of Partial Differential Equations: Finite Difference Methods*. Oxford Applied Mathematics and Computing Science Series. Oxford University Press, 3rd edition, 1985.

[167] L. Stals. Comparison of Non-Linear Solvers for the Solution of Radiation Transport Equations. *Electronic Transactions on Numerical Analysis*, 15:78–93, 2003.

[168] G. Strang. *Linear Algebra and its Applications*. Harcourt Brace Jovanovich, 3rd edition, 1988.

[169] G. Strang and G. Fix. *An Analysis of the Finite Element Method*. Prentice-Hall Series in Automatic Computation. Prentice-Hall, 1973.

[170] W. Subber and S. Loisel. Schwarz preconditioners for stochastic elliptic PDEs. *Computer Methods in Applied Mechanics and Engineering*, 272:34–57, 2014.

[171] J. Summers. Slow power computation by 64-bit glibc. `http://entropymine.com/imageworsener/slowpow/`, 2011. Accessed: 2014-07-10.

[172] S.H. Teng and C.W. Wong. Unstructured mesh generation: Theory, practice and perspectives. *International Journal of Computational Geometry and Applications*, 10(3):227–266, 2000.

[173] F. Tisseur. Newton's Method in Floating Point Arithmetic and Iterative Refinement of Generalized Eigenvalue Problems. *SIAM Journal on Matrix Analysis and Applications*, 22(4):1038–1057, 2001.

[174] D. Titley-Peloquin, J. Pestana, and A.J. Wathen. GMRES convergence bounds that depend on the right-hand-side vector. *IMA Journal of Numerical Analysis*, 341(2):462–479, 2014.

[175] S.K. Tomar. Robust algebraic multilevel preconditioning in H(curl) and H(div). *Computers & Mathematics with Applications*, 66:1024–1046, 2013.

[176] A. Toselli and O. Widlund. *Domain Decomposition Methods - Algorithms and Theory*, volume 34 of *Springer Series in Computational Mathematics*. Springer Berlin / Heidelberg, 2005.

[177] U. Trottenberg, C. Oosterlee, and A. Schüller. *Multigrid*. Academic Press, 2001.

[178] Valgrind. Valgrind User Manual. `http://valgrind.org/docs/`, 2014. Accessed: 2014-09-30.

[179] M.T. van Genuchten. A Closed-form Equation for Predicting the Hydraulic Conductivity of Unsaturated Soils. *Soil Science Society of America Journal*, 44(5):892–898, 1980.

[180] P. Vassilevski and U.M. Yang. Reducing communication in algebraic multigrid using additive variants. *Numerical Linear Algebra with Applications*, 21:275–296, 2014.

[181] P.S. Vassilevski. *Multilevel block factorization preconditioners: Matrix-based analysis and algorithms for solving finite element equations*. Springer, New York, 2008.

[182] E. Vecharynski and A.V. Knyazev. Absolute Value Preconditioning for Symmetric Indefinite Linear Systems. *SIAM Journal on Scientific Computing*, 35(2):A696–A718, 2013.

[183] H.K. Versteeg and W. Malalasekera. *An introduction to Computational Fluid Dynamics - The Finite Volume Method*. Longman Scientific & Technical, 1995.

[184] H.-J. Vogel, I. Cousin, I. Ippisch, and P. Bastian. The dominant role of structure for solute transport in soil: experimental evidence and modelling of structure and transport in a field experiment. *Hydrology and Earth System Sciences*, 10:495–506, 2006.

[185] T. Vogel and M. Cislerova. On the reliability of Unsaturated Hydraulic Conductivity Calculated from the Moisture Retention Curve. *Transport in Porous Media*, 3:1–15, 1988.

[186] T. Vogel, M. T. van Genuchten, and M. Cislerova. Effect of the shape of the soil hydraulic functions near saturation on variably-saturated flow predictions. *Advances in Water Resources*, 24:133–144, 2001.

[187] W.L. Wan and T.F. Chan. A phase error analysis of multigrid methods for hyperbolic equations. *SIAM Journal on Scientific Computing*, 25(3):857–880, 2003.

[188] S. Weill, E. Mouche, and J. Patin. A generalized Richards equation for surface/subsurface flow modelling. *Journal of Hydrology*, 366:9–20, 2009.

[189] M. Weiser, A. Schiela, and P. Deuflhard. Asymptotic mesh independence of Newton's method revisited. *SIAM Journal on Numerical Analysis*, 42:1830–1845, 2005.

[190] R. Wienands and W. Joppich. *Practical Fourier Analysis for Multigrid Methods*. Chapman and Hall/CRC, 2005.

[191] R. Wienands and C.W. Oosterlee. On Three-Grid Fourier Analysis for Multigrid. *SIAM Journal on Numerical Analysis*, 23(2):651–671, 2001.

[192] S.M. Wise, J.S. Lowengrub, and V. Cristini. An adaptive multigrid algorithm for simulating solid tumor growth using mixture models. *Mathematical and Computer Modelling*, 53:1–20, 2011.

[193] H.J. Wu and Z.M. Chen. Uniform convergence of multigrid V-cycle on adaptively refined finite element meshes for second order elliptic problems. *Science in China*, 39:1405–1429, 2006.

[194] S.L. Wu and C.X. Li. A splitting iterative method for the discrete dynamic linear systems. *Journal of Computational and Applied Mathematics*, 267:49–60, 2014.

[195] D. Xie. New Nonlinear Multigrid Analysis. In *Seventh Copper Mountain Conference on Multigrid Methods*, pages 793–808, 1995.

[196] J. Xu. Iterative Methods by Space Decomposition and Subspace Correction. *SIAM Review*, 34(4):581–613, December 1992.

[197] J. Xu and Y.R. Zhu. Uniform convergent multigrid methods for elliptic problems with strongly discontinuous coefficients. *Mathematical Models & Methods in Applied Sciences*, 18(1):77–105, 2008.

[198] J.C. Xu. The single-grid multilevel method and its applications. *Inverse Problems and Imaging*, 7(3):987–1005, 2013.

[199] X. Xu, H. Chen, and R.H.W. Hoppe. Optimality of local multilevel methods on adaptively refined meshes for elliptic boundary value problems. *Journal of Numerical Mathematics*, 18:59–90, 2010.

[200] A. Younes, M. Fahs, and B. Belfort. Monotonicity of the cell-centred triangular MPFA method for saturated and unsaturated flow in heterogeneous porous media. *Journal of Hydrology*, 504:132–141, 2013.

[201] H. Yserentant. On the convergence of multi-level methods for strongly nonuniform families of grids and any number of smoothing steps per level. *Computing*, 30:305–313, 1983.

[202] H. Yserentant. Old and new convergence proofs for multigrid methods. *Acta Numerica*, 2:285–326, 1993.

[203] C.T. Zamfirescu. Survey of two-dimensional acute triangulations. *Discrete Mathematics*, 313(1):35–49, 2013.

[204] S. Zhang and J. Xu. Optimal Solvers for Fourth-Order PDEs Discretized on Unstructured Grids. *SIAM Journal on Numerical Analysis*, 52:282–307, 2014.

[205] J. Zhao. Convergence of V-cycle and F-cycle multigrid methods for the biharmonic problem using the Morley element. *Electronic Transactions on Numerical Analysis*, 17:112–132, 2004.

[206] Y. Zhu. Domain decomposition preconditioners for elliptic equations with jump coefficients. *Numerical Linear Algebra with Applications*, 15(2-3):27–289, 2008.

[207] Y.R. Zhu. Analysis of a multigrid preconditioner for Crouziex-Raviart discretisation of elliptic partial differential equation with jump coefficients. *Numerical Linear Algebra with Applications*, 21(1):24–38, 2014.

[208] O.C. Zienkiewicz and K. Morgan. *Finite Elements and Approximation*. John Wiley and Sons, 1983.

[209] W. Zulehner. Nonstandard norms and robust estimates for saddle point problems. *SIAM Journal on Matrix Analysis and Applications*, 32(2):536–560, 2011.