Fast Iterative Methods for Solving Large Systems Arising from Variational Models in Image Processing

Thesis submitted in accordance with the requirements of the University of Liverpool for the degree of Doctor in Philosophy by Joseph David Savage.

May 2006

Contents

Acknowledgements

I would like to thank my supervisor Dr Ke Chen for his support throughout my time as a PhD student and for giving me the opportunity to undertake this project in the first place. I would also like to thank several other postgrads and postdocs, Dr Stuart Hawkins, Dr Martyn Hughes, Dr Anwar Hussein and Dr Kamal Shanazari, who offered me advice on various aspects of studying for a PhD. I would specifically like to thank Dr Hughes for allowing me to use his latex template for writing this thesis. I also acknowledge the useful input of Prof. Li Wei-Guo, who worked in collaboration with myself and Dr Chen during a one year visit to Liverpool from Dong-ying in China.

Funding from the EPSRC and the support of the Department of Mathematical Sciences at the University of Liverpool is acknowledged. I also acknowledge the use of image generation MATLAB codes provided to my supervisor by Hao Min Zhou (while at UCLA) and some 1-d data and programs provided by Antonio Marquina (Universitat de Valencia). Use was also made of the resources available at `http://www.math.montana.edu/~vogel`, `http://www.cs.uiowa.edu/~dstewart/classes/22m174-2000/22m174.html` and `http://www.siam.org/books/fr23` when writing MATLAB codes.

Last but not least I would like to thank my family for their continuing support throughout this process.

Abstract

This thesis is concerned with the development of fast iterative methods for the numerical solution of nonlinear partial differential equations, which result from the application of Tikhonov regularization to image restoration problems.

The majority of the work is concerned with image denoising using the total variation regularization functional, well known for its edge capturing properties. A nonlinear multigrid (FAS) method with a new smoother is developed. Numerical results showing the effectiveness of the method over nonlinear multigrid with other smoothers and over other existing iterative methods are presented. Unfortunately the nonlinear multigrid method does not perform well when a perturbing parameter β is taken too small. A method which is more robust with respect to β is the fixed point method with algebraic multigrid linear solver, a technique for accelerating this method by recycling algebraic multigrid setup data is also presented.

In later chapters an attempt is made to extend the use of the nonlinear multigrid method to several other denoising models, which use alternative regularization functionals designed to reduce the staircasing effect seen in images denoised using the total variation regularization functional. The nonlinear multigrid method is shown to be particularly effective in one of the cases. Also presented are iterative methods for solving the equations associated with the deblurring and denoising of an image with total variation regularization. Results showing the possible advantage of these methods in problems with heavy noise and moderate blur are presented.

Publications

1. J. Savage, K. Chen, *On Multigrids for Solving a Class of Improved Total Variation Based Staircasing Reduction Models*, To appear in the proceedings of the International Conference on PDE-Based Image Processing and Related Inverse Problems, CMA Oslo, Springer-Verlag, 2006.

2. J. Savage, K. Chen, *An Accelerated Algebraic Multigrid Algorithm for Total-Variation Denoising*, Submitted to BIT, 2006.

3. K. Chen, J. Savage, *On Two New and Nonlinear Iterative Methods for Total-Variation Based Image Deblurring*, in Proceedings of the Fifth UK Conference on Boundary Integral Methods, K. Chen, ed., 2005.

4. J. Savage, K. Chen, *An improved and accelerated Nonlinear Multigrid Method for Total-Variation Denoising*, International Journal of Computer Mathematics., **82**, No 8, August 2005, pp.1001-1015.

5. J. Savage, K. Chen, *Accelerated Iterative Methods for Solving an Inverse Problem Arising from Image Restoration*, in Proceedings of the Fourth UK Conference on Boundary Integral Methods, S. Amini, ed., 2003.

List of Figures

List of Tables

List of Algorithms

List of Abbreviations

- AMG=Algebraic Multigrid
- BCCB=Block Circulant with Circulant Blocks
- BTTB=Block Toeplitz with Toeplitz Blocks
- DFT=Discrete Fourier Transform
- FAS=Full Approximation Scheme
- FFT=Fast Fourier Transform
- FP=Fixed Point
- GMG=Geometric Multigrid
- G-S=Gauss-Seidel
- PCG=Preconditioned Conjugate Gradient
- PDE=Partial Differential Equation
- PSNR=Peak Signal to Noise Ratio
- SNR=Signal to Noise Ratio
- SOR=Successive Over Relaxation
- SPD=Symmetric Positive Definite
- TV=Total Variation

Chapter 1

Introduction

A picture paints a thousand words is an often used phrase which alludes to the power that images have. In today's modern society images are all around us, they have the power to spark controversy across the world, to diagnose illnesses and to catch criminals. We are all it seems, constantly being captured on CCTV, speed cameras and mobile phones and possessing the wrong type of image can land you in a lot of trouble. It is not surprising then that there is a large amount of mathematical research being carried out which relates in some way to images. In image registration (or matching) a reference image is mapped into a target image, in medical imaging for example the images could be two images of the same part of the body obtained using two different scanning techniques or two scans taken at two different times, registration can also be used to map two images of the same scene taken at different angles into each other. In image segmentation the task is to pick out features of interest in an image from the rest of the image (the background), in image tracking the task is to track a particular image feature across several scenes, for example tracking the movement of an individual or vehicle on CCTV footage. In image recognition two images are compared and the task is to determine whether they are of the same thing, e.g fingerprint matching, facial recognition software.

A precursor to many of these applications and an important application in itself, is the task of restoring degraded images. The degradation in the image usually results from two phenomena, the first is blurring which can occur because of imperfections in the recording device or light having to pass through some medium e.g the earths atmosphere in astronomical imaging. The operator which causes this blurring effect is usually known. The second is the phenomena of random noise which can be added during the transmission of the image or be due to problems with the recording device. There are several different types of noise which may occur, each requiring different restoration techniques. The image restoration problem is an example of an inverse problem and is often also ill-posed. It is important to note here that the original image will never be restored exactly and some loss of information is inevitable.

Research in the field of image restoration, broadly falls in to two categories, the design of new models for accurately restoring degraded images and the efficient solution of the resulting equations (obviously many works encompass both of these). The work in this thesis falls into the latter category and is concerned mainly with the popular Tikhonov regularization approach to restoring blurred and noisy images using the total-variation regularization functional proposed originally by Rudin Osher and Fatemi [78], which is well known for its excellent edge recovering properties. In Chapter 3 a more thorough introduction to this method is given but for the purpose of introducing the research problem I give a very brief introduction here. The blurred noisy image is modeled by $z = \mathcal{K}u + n$ where, the blurring operator denoted $\mathcal{K}$ is assumed to be either the identity (no blurring) or a fredholm integral operator of the first kind and the noise to be additive Gaussian white noise. The problem is to minimize the functional

$$J(u) = \int_{\Omega} \alpha\sqrt{|\nabla u|^2 + \beta} + 1/2(\mathcal{K}u - z)^2 dxdy. \tag{1.1}$$

The Euler-Lagrange equation which gives the solution to the variational problem is:

$$-\alpha\nabla.\left(\frac{\nabla u}{\sqrt{|\nabla u| + \beta}}\right) + \mathcal{K}^*(\mathcal{K}u - z) = 0. \tag{1.2}$$

Due to the discrete nature of images the equation is discretized using finite difference methods. The equation is highly nonlinear and the number of unknowns can be large (typically 256^2-1024^2), the construction of fast iterative methods for solving the discrete equation has therefore been an active area of research over the last decade or so [7, 17, 18, 19, 20, 24, 25, 29, 32, 34, 39, 48, 55, 61, 71, 78, 79, 93, 94, 95, 96, 97] and is the main focus of this thesis.

Multigrid methods [12, 23, 44, 43, 66, 92, 102, 103] based on the recursive application of error smoothing and coarse grid correction have been demonstrated to be efficient solvers for a wide range of (linear and nonlinear) partial differential equations discretized on regular domains (grids) and can also be applied more generally via the black box algebraic multigrid methods [9, 11, 80, 92, 99]. The primary aim of my work was to develop a nonlinear multigrid method for solving the discrete Euler-Lagrange equation of the total-variation problem in the pure denoising ($\mathcal{K} = I$) case but I have also proposed a technique for accelerating the existing fixed point method, when algebraic multigrid is used as the linear solver, extended the work on nonlinear multigrid to other denoising models and proposed an alternative iterative solver for the deblurring problem.

The thesis is organized as follows:

- **Chapter 2** covers various background material which is useful in the chapters that follow, it includes:
  - Useful preliminary definitions and theorems.

- A brief discussion of inverse problems and regularization needed for the introduction of denoising techniques in chapter 3.
- A discussion of the discretization of partial differential equations (PDEs) on regular domains using finite difference methods.
- A review of stationary iterative methods for solving discrete linear systems and nonlinear analogues.
- An introduction to geometric multigrid methods as iterative solvers for discrete elliptic PDEs including algorithms for linear and nonlinear multigrid methods.
- An introduction to the black box algebraic multigrid methods for linear problems.

- **Chapter 3** is an introduction to various aspects of image restoration using the Total Variation (TV) regularization functional, it includes:
  - An introduction to the restoration problem.
  - An introduction to regularization techniques used in image restoration, including the Total Variation regularization method.
  - A brief review of some of the mathematical analysis of Total-Variation regularization.
  - A review of the properties of Total-Variation denoising.
  - Details of the discretization scheme that I use when solving the TV denoising problem and a review of alternative discretization schemes.
  - A review of the iterative solvers currently available for solving the TV denoising problem.

- **Chapter 4** details our attempts to develop a nonlinear multigrid method for solving the discrete TV denoising problem, it includes:
  - A discussion of the various smoothers that we have used, including our new smoother FPGS.
  - The algorithm for the nonlinear multigrid method being used.
  - A comparison of FPGS against other smoothers, which demonstrates its advantage.
  - An investigation into the effect of applying Krylov acceleration to the nonlinear multigrid method.
  - A comparison of nonlinear multigrid with FPGS against other iterative solvers, showing its advanatage in certain situations.
  - Details of the disadvantage of the nonlinear multigrid method, poor performance for small β.

- **Chapter 5** details a method for reducing the overall cost of the fixed point method when algebraic multigrid (AMG) is used as the inner linear solver, it includes:
  - A brief review of linear solvers used within the fixed point method, including a demonstration of the better convergence properties of AMG vs geometric multigrid for small β.
  - A discussion of our idea to recycle AMG setup data, within the fixed point method including preliminary tests which motivate the final algorithm.
  - Tests showing the effectiveness of the new method, in reducing the overall cost of the fixed point method.
- **Chapter 6** is an extension of the work on nonlinear multigrid in chapter 4 to other denoising models, which reduce the staircasing effect, it includes:
  - A review of staircase-reducing models, focusing on 4 models in particular.
  - A discretization scheme for the models in question and algorithms for the nonlinear multigrid method and other iterative solvers which are tested.
  - Details of the implemntation of the nonlinear multigrid method for each of the four models and comparison with other iterative solvers.
  - A focus on one particular model, which gives good quality reconstructions and a demonstration of the effectiveness of nonlinear multigrid for this model.
- **Chapter 7** details some of our recent work on image deblurring, it includes:
  - An introduction to some topics specific to the TV deblurring problem, including the discrete fourier transform, fast fourier transform and Toeplitz and Circulant matrices.
  - Details of the discretization of the TV deblurring problem and a review of iterative methods for solving the discrete problem.
  - A proposal for an alternative deblurring model, which is still under development.
  - Details of two alternative iterative solvers for the TV deblurring problem, which are shown to potentially have an advantage over the fixed point method when the level of blurring is relatively low and the level of noise relatively high.
- **Chapter 8** covers possible future research directions.
- **Appendix A** covers various topics from optimization, which although useful throughout the discussion are somewhat peripheral to my own work, it includes:
  - A definition of constrained and unconstrained optimization and local and global minima.

- Conditions for the existance of minima of functionals on Hilbert spaces.
- Discussion of the optimality conditions for unconstrained problems based on the Frechet and Gateaux derivatives.
- Statement of the KKT optimality conditions for constrained problems on $\mathbb{R}^n$.
- A review of the steepest descent and conjugate gradient optimization methods for quadratic and more general nonlinear functionals on $\mathbb{R}^n$.
- A discussion of multilevel techniques used in optimization.

- **Appendix B** details some cost estimates for the AMG method of chapter 5 implemented in MATLAB.

All experiments presented in the thesis were run in MATLAB on a Sun Fire 880.

Chapter 2

Mathematical Preliminaries and Multigrid Methods

Main Reference Material: [2]-[4],[9, 11, 12, 14, 23, 35, 43, 44, 45, 47, 54, 57, 66, 72, 80, 81, 92, 99, 100, 103, 105]

This chapter introduces various material which will be useful throughout the rest of the thesis. The first two sections introduce some useful definitions and concepts related to normed spaces and operators. The material is meant mainly for reference, a thorough introduction to the topics presented can be found, for example, in [54, 105] from which much of the material has been taken.

§2.3 gives a brief introduction to ill-posed inverse problems and the regularization techniques which are used to approximately solve such problems, this material will be useful in the introduction of image restoration techniques in Chapter 3.

The Main aim of the rest of the chapter is to introduce multigrid methods as iterative solvers for discrete (linear and nonlinear) elliptic partial differential equations (PDEs). §2.4 introduces some techniques and notation associated with the discretization of continuous PDEs on regular domains, §2.5 introduces the stationary iterative methods, which are the building blocks of multigrid methods, §2.6 gives a brief introduction to the principles on which multigrid methods are based and details algorithms for linear and nonlinear multigrid schemes and finally §2.7 covers black box algebraic multigrid methods which can be applied more generally to discrete linear problems. Nonlinear multigrid methods will be applied to the discrete Euler-Lagrange equation of the Total-Variation denoising problem in Chapter 4 and to other denoising problems in Chapter 6, linear (geometric) multigrid methods and

algebraic multigrid methods have been employed within the fixed point method (§3.6.2) as linear solvers and the acceleration of fixed point with algebraic multigrid will be investigated in Chapter 5.

2.1 Normed Spaces

Definition 2.1.1 (Vector Space)
A vector or linear space over a field F (usually the real or complex numbers) is a set S together with the operations of vector addition and scalar multiplication such that, the following conditions hold:

1. *Closure of vector addition: If $u \in S$ and $v \in S$, then $u + v \in S$.*
2. *Commutativity of addition: $u+v=v+u$*
3. *Associativity of addition: $(u+v)+w=u+(v+w)$*
4. *Existence of additive inverse: For each $u \in S$, there exists $(-u) \in S$ such that $u + (-u) = 0$*
5. *Closure of Scalar multiplication: If $\lambda \in F$ and $u \in S$ then $\lambda u \in S$.*
6. *Associativity of scalar multiplication: If $u \in S$, $\lambda, \theta \in F$ then $\lambda(\theta u) = (\lambda\theta)u$.*
7. *Scalar multiplication is distributive: If $u, v \in S$, $\lambda, \nu \in F$ then $(\lambda + \theta)u = \lambda u + \theta v$ and $\lambda(u + v) = \lambda u + \lambda v$.*

A subset of a vector space V which is also a vector space under the same operators of addition and scalar multiplication is called a subspace of V, it is said to be a proper subspace if it is not V itself and a non-trivial subspace if it contains non zero elements.

Example
Examples of vector spaces are

- The spaces $\mathbb{R}^n$ and $\mathbb{C}^n$.
- The space $C^l(\Omega)$ of all functions on the domain $\Omega \subset \mathbb{R}^n$ whose partial derivatives of order up to l are continuous on Ω and the space $C_0^l(\Omega)$ of all functions in $C^l(\Omega)$ with compact support.
- The space $L^p(\Omega)$ of all Lebesgue measurable functions on $\Omega \subset \mathbb{R}^n$ for which $\int_\Omega |u(\mathbf{x})|^p d\mathbf{x} < \infty$. The precise definition of a Lebesgue measurable function will not be covered here it can be found in, for example [2]. Strictly speaking the elements of $L^p(\Omega)$ are not individual functions but equivalence classes of functions which are equal almost everywhere in Ω, but I will ignore this distinction here.

- The Sobolev Space $W^{m,p}$ of functions $u \in L^p$ whose weak partial derivatives up to order m also belong to L^p, see [2] for more details.

■

Definition 2.1.2 (Linear Dependance)
A set of vectors $\{v_1, .., v_n\}$ *is said to be linearly dependant if there is a set of scalars* $c_1, .., c_n$ *not all zero such that* $\sum_{i=1}^{n} c_i v_i = 0$. *If a set of vectors are not linearly dependant they are said to be linearly independent.*

Definition 2.1.3 (Linear Combination)
A linear combination of a finite set of vectors $v_1, .., v_n$ *is a vector of the form* $a_1 v_1 +, .., + a_n v_n$ *where the* a_j *are scalars.*

Definition 2.1.4 (Space Spanned by a Subspace)
If $Y = \{y_1, .., y_n\}$ *is a set of vectors in a vector space* V, *then the set of all linear combinations of elements of* Y *is called the subspace spanned by* Y *and is denoted* $span\{y_1, .., y_n\}$.

Definition 2.1.5 (Finite Basis)
A set $\{y_1, .., y_n\}$ *is called a finite basis for a vector space* V *if it is linearly independent and* $V = span\{y_1, .., y_n\}$.

Example
Any linearly independent set of n vectors in $\mathbb{R}^n$ is a basis for $\mathbb{R}^n$. ■

Definition 2.1.6 (Dimension of a Vector Space)
A vector space is said to be n-dimensional if it has a finite basis of n elements. A vector space with no finite basis is said to be infinite dimensional.

An operator or a mapping from one vector space U into another V (or from a subset of U into a subset of V) $A : U \rightarrow V$ is just a rule which given an element $u \in U$ produces $A(u) \in V$. The simplest type of operator is a functional which maps a vector space to the real or complex numbers. A functional mapping $\mathbb{R}^n$ to $\mathbb{R}$ is usually called a function.

Definition 2.1.7 (Norm)
A norm on a vector space S *is a real-valued functional* $\|.\|$ *on* S *such that*

1. $\|u\| > 0$ *if* $u \neq 0$.
2. $\|\lambda u\| = |\lambda| \|u\|$ *for all scalars* λ *and vectors* u.
3. $\|u + v\| \leq \|u\| + \|v\|$ *for all* $u, v \in S$

A seminorm is defined similarly to above except that axiom 1 is replaced by $\|u\| \geq 0$, it is therefore possible for a seminorm to be equal to zero for some $u \neq 0$.

Definition 2.1.8 (Normed Space)
A Normed space is a vector space S provided with a norm $\|.\|_S$.

A special type of normed space is an inner product space.

Definition 2.1.9 (Inner Product)
An inner product on the vector space S is a functional $(.,.)_S$ on $S \times S$ which satisfies:

1. $(u,v)_S = \overline{(v,u)}_S$ *for all* $u, v \in S$.
2. $(\lambda u, v)_S = \lambda(u,v)_S$
3. $(u+v, w)_S = (u,w) + (v,w)$
4. $(u,u)_S > 0$ *when* $u \neq 0$.

Any inner product induces a norm, which is defined as follows $\|u\| = (u,u)^{1/2}$.
Example
The classic example of a norm is the Euclidean norm on $\mathbb{R}^n$ defined by $\|\mathbf{x}\|_2 = \sqrt{\sum_{i=1}^n x_i^2}$ this is sometimes also written $|\mathbf{x}|$ and is induced by the Euclidean inner product $(\mathbf{x},\mathbf{y})_2 = \mathbf{y}^T\mathbf{x}$. An example of a normed space, which is not an inner product space is the space of all bounded continuous complex valued functions on $\Omega \subset \mathbb{R}^n$ together with the supremum norm defined by $\|f\|_\infty = \sup_{\mathbf{x}\in\Omega} |f(\mathbf{x})|$. ∎

Theorem 2.1.1 The Cauchy Schwarz Inequality
For any inner product $(.,.)$

$$|(u,v)| \leq \|u\|\|v\|. \tag{2.1}$$

2.1.1 Sequences and Convergence

Definition 2.1.10 (Cauchy Sequence)
A sequence $\{v_k\}$ in a normed space is said to be a cauchy sequence if for all $\epsilon > 0$ there exits a K such that any $k > K$ and any $l > K$ implies that $\|v_k - v_l\| < \epsilon$.

If a sequence is a cauchy sequence then we can force the terms to be as close to each other as we choose by taking k sufficiently large.

Definition 2.1.11 (Convergent Sequence)
A sequence $\{v_k\}$ in a normed space is said to converge to v (denoted $v_k \to v$ as $k \to \infty$ or $\lim_{k\to\infty} v_k = v$) if for all $\epsilon > 0$ there exits a K such that $k > K$ implies that $\|v_k - v\| < \epsilon$.

If a sequence converges to v then we can force the terms to be as close to v as we choose by taking k sufficiently large.

Definition 2.1.12 (Weak Convergence)
A sequence $\{v_k\}$ in an inner product space V is said to converge weakly to v (denoted $v_k \rightharpoonup v$ as $k \to \infty$) if for all $w \in V$, the sequence (v_k, w) converges to (v, w).

Strong convergence implies weak convergence since by the Cauchy Schwartz inequality $|(v,w) - (v_k,w)| \leq \|v - v_k\|\|w\|$ and $\|v - v_k\| \to 0$ as $k \to \infty$ if v_k converges to v.

2.1.2 Open and Closed Sets

Definition 2.1.13 (Open Set)
A subset S of a normed space N is said to be open if for each $u \in S$ there exists a $\delta > 0$ such that $\|u - v\| < \delta$ for all $v \in S$.

This definition says that any point in an open set can be surrounded by a ball centered at that point which lies entirely in S.
Example
An example of an open set in $\mathbb{R}^n$ is the set $\{\mathbf{x} | \|\mathbf{x} - \mathbf{a}\|_2 < r\}$ for some $\mathbf{a} \in \mathbb{R}^n$ and $r \in \mathbb{R}$. ∎

Definition 2.1.14 (Limit Point)
v is said to be a limit point of a subset S of a normed space N if there exists a sequence $\{v_n\}$ of elements of S such that v_n converges to v and v is not a member of the sequence $\{v_n\}$.

Definition 2.1.15 (Closed Set)
A subset S of a normed space N is said to be closed if all of it's limit points are contained in S.

It can be shown that S is closed if and only if its complement $N - S$ is open. Subsets S can be both closed and open, the obvious example being the normed space N itself, or neither.

Definition 2.1.16 (Closure)
The closure of a subset S of a normed space N is the union of S and all its limit points and is denoted $\bar{S}$.

2.1.3 Banach and Hilbert Spaces

A normed space in which every Cauchy sequence converges (a complete normed space) is known as a Banach space, similarly a complete inner product space is known as a Hilbert space, see [105] for more details.
Example
Two relevant examples of Hilbert spaces are the space $\mathbb{R}^n$ together with the Euclidean inner product and the space $L^2(\Omega)$ together with the inner product defined by $(f,g)_{L^2(\Omega)} = \int_\Omega f(\mathbf{x})g(\mathbf{x})d\mathbf{x}$. ∎

Definition 2.1.17 (Orthogonal Set)
A set of vectors $\{x_i\}$ in an inner product space is called an orthogonal set if $(x_i, x_j) = 0$ for $i \neq j$ and no $x_i = 0$.

A set of vectors is called orthonormal if in addition $(x_i, x_i) = 1$ for all i. It is clear that any finite orthogonal set is linearly independent since if we were to take the inner product of both sides of $\sum_{i=1}^{n} c_i x_i = 0$ with x_j for any j we would have $c_j(x_j, x_j) = 0$ and hence $c_j = 0$.

Definition 2.1.18 (Orthogonal Basis)
An orthogonal basis for an inner product space S is a (possibly infinite) orthogonal set $\{e_n\}$ such that for any $u \in S$ there exists scalars c_n such that $u = \sum_n c_n e_n$.

If $u = \sum_n c_n e_n$ then we can see from the orthogonality of the e_n that $c_l = (u, e_l)/(e_l, e_l)$ or just (u, e_l) if the e_n are orthonormal. Furthermore if the basis is infinite, these coefficients will also minimize the distance $\|u - \sum_{n=1}^{N} c_n e_n\|$ between u and the truncated sum involving N terms. In some books a Hilbert space is defined more strongly than above as a complete inner product space with an orthogonal basis.

Definition 2.1.19 (Orthogonal Complement)
If S is a subset of a Hilbert space H, then the orthogonal complement of S is defined to be the set $S^{\perp} = \{u \in H | (u, v) = 0,$ for all $v \in S\}$.

If U is a subspace of a Hilbert space H with an orthogonal basis, then any element $y \in H$ can be written uniquely as $y = u + v$, where $u \in U$ and $v \in U^{\perp}$, this is sometimes written $H = U + U^{\perp}$.

2.2 Operators

Recall from above that an operator $A : X \rightarrow Y$ is a rule which takes any element of a subset X of a vector space V and produces an element of a subset Y of a vector space U. The domain of A denoted $D(A)$ is just X , the image of $x \in X$ under A is $A(x)$, and the range of A denoted $R(A)$ is $\{A(x) | x \in X\}$, clearly $R(A) \subset Y$. The Null space of an operator A denoted $N(A)$ is all the elements of X which are mapped to 0 by A i.e $\{x \in X | A(x) = 0\}$.

Definition 2.2.1 (Identity Operator)
The identity operator I mapping a vector space U into itself is the operator which maps every element of U to itself.

Definition 2.2.2 (Inverse Operator)
The operator $A : U \rightarrow V$ is invertible if for each $v \in V$ there is one and only one $u \in U$ such that $A(u) = v$. The Inverse operator of A is denoted $A^{-1} : V \rightarrow U$.

If the inverse of an operator exists then clearly $A^{-1}A = AA^{-1} = I$.

2.2.1 Linear Operators

Definition 2.2.3 (Continuous Operator)
An operator $A : S \to M$, where S is a subset of N and N and M are normed spaces, is said to be continuous at $\bar{u} \in S$ if for any $\epsilon > 0$ there exists a $\delta > 0$ such that $\|A(u) - A(v)\| < \epsilon$ for all $v \in N$ such that $\|u - v\| < \delta$. A is continuous on S if it is continuous at all points in S.

Definition 2.2.4 (Linear Operator)
An operator $A : V \to W$, where V and W are vector spaces is linear if $A(av_1 + bv_2) = aA(v_1) + bA(v_2)$ for all $v_1, v_2 \in V$, and all scalars a, b.

Example
A linear operator mapping $\mathbb{R}^n$ to $\mathbb{R}^m$ is defined by a matrix of A size $m \times n$, then given $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{y} = A\mathbf{x} \in \mathbb{R}^m$. ∎

Theorem 2.2.1 *If a linear operator $A : N \to M$ is continuous at a single point in N it is continuous on M.*

Definition 2.2.5 (Operator Norm)
The norm of an operator $A : N \to M$, where N and M are normed spaces is defined as follows

$$\|A\| = \sup\{\|A(u)\|/\|u\| : u \in N, u \neq 0\}. \tag{2.2}$$

Definition 2.2.6 (Bounded Operator)
An Operator $A : N \to M$, where N and M are normed spaces is bounded if there exits a number θ such that $\|A(u)\| \leq \theta\|u\|$ for all $u \in N$ i.e. $\|A\| \leq \theta$.

Theorem 2.2.2 *A linear operator is bounded if and only if it is continuous.*

Theorem 2.2.3 *The set of all bounded linear operators from N to M, where N, M are normed spaces is itself a normed space under the norm of definition 2.2.5 and the operators of addition and scalar multiplication defined respectively by $(A+B)u = Au+Bu$ and $(cA)u = c(Au)$ for bounded linear operators A and B, scalar c and $u \in N$. This normed space is denoted $B(N, M)$ and furthermore is a Banach space if M is a Banach space.*

Definition 2.2.7 (Dual)
The dual of a vector space V is the set of all linear functionals $f : V \to \mathbb{R}$ and is denoted V^*.

Definition 2.2.8 (Bilinear Mapping)
A bilinear mapping $B : U \times U \to V$, where U and V are vector spaces is a rule which given any two elements u_1, u_2 of U produces $B(u_1, u_2) \in V$ such that

$$B(au_1 + bu_2, u_3) = aB(u_1, u_3) + bB(u_2, u_3) \tag{2.3}$$

$$B(u_1, au_2 + bu_3) = aB(u_1, u_2) + bB(u_1, u_3) \tag{2.4}$$

for all scalars a, b and all $u_1, u_2, u_3 \in U$.

Example
A bilinear mapping from $\mathbb{R}^n \times \mathbb{R}^n$ into $\mathbb{R}$ is defined by an $n \times n$ matrix A, and $B(\mathbf{u}, \mathbf{v}) = \mathbf{v}^T A \mathbf{u}$
■

2.2.2 Compact Operators

Definition 2.2.9 (Compact Set)
A subset S of a normed space N is compact if every infinite sequence of elements of S has a subsequence which converges to an element of S.

Definition 2.2.10 (Relatively Compact Set)
A subset S of N is relatively compact if every sequence in S has a subsequence converging to an element of N.

Definition 2.2.11 (Bounded Set)
A subset S of a normed space N is called bounded if there is a number θ such that $\|x\| \leq \theta$ for all $x \in S$.

Definition 2.2.12 (Compact Operator)
An operator is compact on a normed space if it maps every bounded set into a relatively compact set.

Any bounded operator on a finite dimensional space is compact, since bounded operators map bounded sets into bounded sets and a bounded set in a finite dimensional space is relatively compact.

2.2.3 Operators on Hilbert Spaces

Theorem 2.2.4 Reisz Representation Theorem *If $f : H \rightarrow \mathbb{R}$ is a continuous linear functional and H is a Hilbert space, then there exists a unique $v \in H$ such that $f(u) = (u, v)$ for all $u \in H$.*

Definition 2.2.13 (Eigenvectors and Eigenvalues)
If $A : V \rightarrow V$ is a linear operator mapping a vector space V into itself, then $v \in V$ is said to be an eigenvector of A if $Av = \lambda v$. The number λ is known as an eigenvalue corresponding to the eigenvector v.

Theorem 2.2.5 *Let $A \in B(H1, H2)$, where $H1$ and $H2$ are Hilbert spaces, then there exists a unique operator $A^* \in B(H2, H1)$ called the adjoint of A such that $(Au, v)_{H2} = (u, A^*v)_{H1}$ for all $u \in H1$ and $v \in H2$.*

If $H1 = H2$ and $A^* = A$ then A is said to be self-adjoint. If $A : V \to H$ is an operator on a dense subspace V of H then A is symmetric if $(u, Av) = (Av, u)$ for all $u, v \in V$. Symmetry is a generalization of self-adjointness and any bounded self-adjoint operator on H is symmetric.

Theorem 2.2.6

If A is a self adjoint operator on a Hilbert space, then all it's eigenvalues are real and it's eigenvectors form an orthogonal set.

Proof

If $Av = \lambda v$, then $\lambda\|v\|^2 = \lambda(v, v) = (Av, v)$, since A is self-adjoint this equal to $(v, Av) = (v, \lambda v) = \bar{\lambda}\|v\|^2$ i.e $\lambda = \bar{\lambda}$ and hence λ is real. If in addition $Au = \sigma u$, then since $(v, Au) = (Av, u)$, $\sigma(v, u) = \lambda(v, u)$ and so $(v, u) = 0$ when $\lambda \neq \sigma$. ∎

Theorem 2.2.7 *Let A be a compact self-adjoint operator on a Hilbert space H, then H has a basis (e_n) consisting of orthonormal eigenvectors of A. If H is infinite dimensional then the corresponding eigenvalues λ_n tend to zero as $n \to \infty$ and if $y = \sum_n c_n e_n$ then $Ay = \sum_n \lambda_n c_n e_n$.*

This is equivalent to saying that the eigenvectors of A form a complete set, a complete set of vectors being a set such that the only vector orthogonal to all members of the set is the zero vector.

Definition 2.2.14 (Positive Definite Operator)

An operator A on an inner product space is called positive definite if $(u, Au) > 0$ for all $u \neq 0$ and positive semidefinite if $(u, Au) \geq 0$ for all u, similar definitions exist for negative definite and negative semidefinite operators.

Theorem 2.2.8

The eigenvalues of a positive definite operator are positive.

Proof

If $Av = \lambda v$ and $(v, Av) > 0$ then $\bar{\lambda}(v, v) > 0$ hence λ is real and positive. ∎

Theorem 2.2.9

If the eigenvectors of an operator A form a complete orthogonal set, and all the eigenvalues of A are positive then A is positive definite.

Definition 2.2.15 (Strongly Positive Operator)

A symmetric operator A on an inner product space V is said to be strongly positive if $(v, Av) \geq \theta(v, v)$ for all $v \in V$. In this case θ is said to be a lower bound for A.

Theorem 2.2.10 *If A is a symmetric operator with a complete set of eigenvectors, then A is strongly positive if there exists a $\theta > 0$ such that all eigenvalues of A are greater than or equal to θ.*

From this result it is clear that for operators on a finite dimensional space who's eigenvectors form a complete set, strong positivity is equivalent to positive definiteness. From Theorem 2.2.7 it is also clear that compact operators on infinite dimensional Hilbert spaces are not strongly positive.

2.3 Inverse and Ill-Posed Problems and Regularization

An Inverse Problem is a problem in which one is given the output data from a system and wishes to find the unknown input which created this data as opposed to the direct problem in which one generates the output from the input. As an example in image processing problems seen later, one is given a blurred image contaminated with random noise and wishes to find the true image from this data and knowledge of the blurring operator. In reality inverse problems are often ill-posed and small errors in the data can lead to very large errors in the solution, regularization methods to find an approximation to the solution are then required. In this section I give a very brief introduction to the concepts of ill-posedness and regularization, focusing on compact operators. For a more thorough treatment of the topics of ill-posed problems and regularization see [47, 56, 97].

2.3.1 Ill-Posed Problems and the Generalized Inverse

Definition 2.3.1 (Well-Posedness)

If $K : H_1 \rightarrow H_2$ where H_1 and H_2 are Hilbert Spaces then the problem $K(u) = z$ is said to be well-posed, in the sense of Hadamard, if

1. *For each $z \in H_2$ there exists a $u \in H_1$ such that $K(u) = z$ holds.*
2. *u is unique.*
3. *The solution is stable with respect to perturbations in the data, that is given $u^* \in H_1$ and $z^* \in H_2$ such that $K(u^*) = z^*$ then for every $\epsilon > 0$ there exists a $\delta > 0$ such that $\|K(u) - z^*\| < \delta$ implies that $\|u - u^*\| < \epsilon$.*

A Problem that is not well-posed is said to be ill-posed.

The first condition is equivalent to the requirement that the range of K, R(K) is equal to H_2. In the case of linear operators, the second condition is equivalent to requiring that $N(K) = \{0\}$ where $N(K)$ is the null space of K i.e if $Ku = z$ then $K(u+v) = z$ if and only if $v = 0$. If the first two conditions hold then the inverse of K exists and the third condition says that the inverse should be continuous. From now on we will focus on linear operators only.

In cases where $z \notin R(K)$ one may still be interested in finding some solution that approximately solves the problem, similarly if the solution to the problem is not unique one may want to find some specific solution, this can be achieved via the generalized inverse (or pseudoinverse) of the problem.

Definition 2.3.2 (The Generalized Inverse)
The Generalized inverse $K^\dagger$ of $K \in B(H_1, H_2)$ is defined as the unique linear extension of $\tilde{K}^{-1}$ to the domain of $K^\dagger$

$$D(K^\dagger) = R(K) + R(K)^\perp \tag{2.5}$$

with

$$N(K^\dagger) = R(K)^\perp \tag{2.6}$$

where

$$\tilde{K} = K|_{N(K)^\perp} : N(K)^\perp \to R(K). \tag{2.7}$$

To see that $K^\dagger$ is well defined consider the following; The domain of $\tilde{K}$ is $N(K)^\perp$, therefore if $w \in D(\tilde{K})$, $(w, v) = 0$ for all $v \in H_1$ such that $Kv = 0$. Given that $N(\tilde{K}) = \left\{w \in D(\tilde{K}) | Kw = 0\right\}$ any $w \in N(\tilde{K})$ must satisfy $(w, w) = 0$. This is only possible if $w = 0$, hence $N(\tilde{K}) = \{0\}$, furthermore since H_1 can be written as $N(K) + N(K)^\perp$ and any $w \in N(K)$ by definition satisfies $Kw = 0$ we see that the range of $\tilde{K}$ is the same as the range of K, hence $\tilde{K}^{-1}$ exists.

Now $K^\dagger$ is required to be the linear extension of $\tilde{K}^{-1}$ to the domain of $K^\dagger = R(K) + R(K)^\perp$ and any $z \in R(K) + R(K)^\perp = H_2$ can be uniquely defined by $z = z_1 + z_2$ with $z_1 \in R(K)$ and $z_2 \in R(K)^\perp$. Since we also require that $N(K^\dagger) = R(K)^\perp$ we must have $K^\dagger z = \tilde{K}^{-1} z_1$. Obviously if $z \in R(K)$, then $KK^\dagger z = z$.

The generalized inverse can be characterised via the idea of least squares solutions of $Ku = z$.

Definition 2.3.3 (Least Squares Solution)
Let $K \in B(H_1, H_2)$ where H_1 and H_2 are Hilbert spaces, then $u^* \in H_1$ is called a least squares solution of $Ku = z$ if

$$\|Ku^* - z\|_{H_2} = \inf\{\|Ku - z\|_{H_2} | u \in H_1\}. \tag{2.8}$$

$u^* \in H_1$ is called the best approximate solution of $Ku = z$ if u is a least squares solution of $Ku = z$ and

$$\|u^*\|_{H_1} = \inf\{\|v\| | v \text{ is a least squares solution of } Ku = z\}. \tag{2.9}$$

With this definition we can introduce the following theorem, the proof of which can be found in [47].

Theorem 2.3.1 *If $z \in D(K^\dagger)$ then $Ku = z$ has a unique best approximate solution given by*

$$u^\dagger = K^\dagger z \tag{2.10}$$

and the set of all least squares solutions is given by $u^\dagger + N(K)$.

2.3.2 Compact Operators and Singular Systems

Recall that a compact operator (Definition 2.2.12) is a bounded linear operator which maps any bounded set into a relatively compact set. An example of a compact operator which is of particular interest here is the Fredhlom first kind integral operator mapping $L^2(\Omega) \to L^2(\Omega)$ defined by

$$(Ku)(\mathbf{x}) = \int_\Omega k(\mathbf{x},\mathbf{y})u(\mathbf{x})dy, \qquad \mathbf{x} \in \Omega. \tag{2.11}$$

where $\int_\Omega \int_\Omega k(\mathbf{x},\mathbf{y})^2 dxdy < \infty$.

It can be shown that for any compact operator K on infinite-dimensional Hilbert spaces $Ku = z$ is an ill-posed problem. If $R(K)$ is infinite dimensional, this is because the first and third conditions on a well-posed problem are violated, if $R(K)$ has finite dimension then the second condition is violated.

For any compact operator K, K^*K is a self adjoint compact operator. From the eigendecomposition of K^*K (see Theorem 2.2.7) a singular system for K defined as below, can be found.

Definition 2.3.4 (Singular System)
A singular system for a compact linear operator $K : H_1 \to H_2$ is a countable set of triplets $\{u_j, \sigma_j, v_j\}_j$ with the following properties

1. *The right singular vectors v_j form an orthonormal basis for $N(K)^\perp$*
2. *The left singular vectors u_j form an orthonormal basis for the closure of $R(K)$*
3. *The singular values σ_j are positive real numbers and are in nonincreasing order $\sigma_1 \geq \sigma_2 \geq ... > 0$.*
4. *For each j, $Kv_j = \sigma_j u_j$.*
5. *For each j, $K^*u_j = \sigma_j v_j$.*

In addition if $R(K)$ is infinite dimensional then $\lim_{j\to\infty} \sigma_j = 0$.

Using the singular system we can write

$$Ku = \sum_j \sigma_j (u, v_j)_{H_1} u_j \tag{2.12}$$

and

$$K^* z = \sum_j \sigma_j (z, u_j)_{H_2} v_j, \tag{2.13}$$

furthermore it can be shown that the generalized inverse has the following representation

$$K^\dagger z = \sum_j \frac{(z, u_j)_{H_2}}{\sigma_j} v_j. \tag{2.14}$$

If the range of K is infinite dimensional then the sums in all three cases will be infinite.

From the representation of the generalized inverse in (2.14) we can see that small errors in z can be amplified by the application of the generalized inverse due to the division by small singular values (particularly if $R(K)$ is infinite dimensional). For example if we take $\hat{z} = z + \delta u_l$ for some l and $\delta > 0$ and small, then we see that $\|\hat{z} - z\|_{H_2} = \|\delta u_l\|_{H_2} = \delta$ (since $(u_l, u_l)_{H_2} = 1$), on the other hand we have

$$K^\dagger \hat{z} = \sum_j \frac{(z + \delta u_l, u_j)_{H_2}}{\sigma_j} v_j = K^\dagger z + \frac{\delta}{\sigma_l} v_l \tag{2.15}$$

and hence $\|K^\dagger \hat{z} - K^\dagger z\|_{H_1} = \frac{\delta}{\sigma_l}$, as l goes to infinity this error becomes infinitely large. In order to prevent this happening we can filter the singular values i.e we can replace $1/\sigma_j$ in (2.14) by $\phi(\sigma_j^2)/\sigma_j$, two such filters are the truncated singular value decomposition (TSVD) filter function defined as follows

$$\phi(\sigma^2) = \begin{cases} 1 \text{ if } \sigma^2 > \alpha \\ 0 \text{ if } \sigma^2 \leq \alpha \end{cases} \tag{2.16}$$

and the Tikhonov filter function defined by

$$\phi(\sigma^2) = \frac{\sigma^2}{\sigma^2 + \alpha}. \tag{2.17}$$

These filtering schemes are examples of regularization schemes. In each case we see that $\phi(\sigma^2)$ depends on α, which is a positive constant known as the regularization parameter. The choice of the regularization parameter will be important in determining the quality of the solution, choose α too small and to many small singular values remain, choose α too large and the solution will be inaccurate because too many large singular values have been filtered out. In the following the precise definition of a regularization scheme is given.

2.3.3 Regularization Schemes

In the following it is assumed that for the equation $K(u) = z$ where $K : H_1 \to H_2$ there exists an operator R_* that assigns to each $v \in R(K)$ a unique vector $R_*(v) \in H_1$ such that $K(R_*(v)) = v$. In the linear case discussed above R_* is the generalized inverse. A regularization scheme is given by a family of operators $R_\alpha : H_2 \to H_1$, where α lies in an index set I. The regularization scheme converges to R_* if the following two conditions hold:

1. For each $\alpha \in I$, R_α is continuous.

2. Given any $z \in R(K)$, for any sequence $\{z_n\} \subset H_2$ that converges to z there is a sequence $\{\alpha_n\} \subset I$ such that $R_{\alpha_n}(z_n)$ converges to $R_*(z)$ as $n \to \infty$.

In the case that R_α is the linear operator defined by

$$R_\alpha z = \sum_j \frac{\phi_\alpha(\sigma_j^2)(z, u_j)_{H_2}}{\sigma_j} v_j, \tag{2.18}$$

we see from the orthogonality of the v_j that

$$\|R_\alpha z\|_{H_1} = \sqrt{\sum_j \left(\frac{\phi_\alpha(\sigma_j^2)(z, u_j)_{H_2}}{\sigma_j} \right)^2} \leq \left(\sup_j \frac{\phi_\alpha(\sigma_j^2)}{\sigma_j} \right) \|z\|_{H_2}. \tag{2.19}$$

For both the TSVD filter and the Tikhonov filter $\sup_{\sigma>0} \phi_\alpha(\sigma^2)/\sigma = 1/\sqrt{\alpha}$ (in the TSVD case this is trivial in the Tikhonov case it can be seen using simple calculus) and so R_α is bounded and thus continuous for every α. To see that the second condition holds for these two schemes let $\{z_n\}$ be a sequence that converges to z with $\|z_n - z\| < \delta_n$ where $\delta_n \to 0$ as $n \to \infty$, we have

$$\begin{aligned} \|R_{\alpha_n} z_n - R_* z\|_{H_1} &= \|R_{\alpha_n} z_n + R_{\alpha_n} z - R_{\alpha_n} z - R_* z\|_{H_1} \\ &\leq \|R_{\alpha_n} z - R_* z\|_{H_1} + \|R_{\alpha_n} z_n - R_{\alpha_n} z\|_{H_1} \\ &\leq \|R_{\alpha_n} z - R_* z\|_{H_1} + \|R_{\alpha_n}\| \|z_n - z\|_{H_2} \\ &< \|R_{\alpha_n} z - R_* z\|_{H_1} + \|R_{\alpha_n}\| \delta_n. \end{aligned} \tag{2.20}$$

Since R_α is bounded and $R_\alpha \to R_*$ as $\alpha \to 0$ by taking $\{\alpha_n\} = \{\delta_n\}$ we will have $R_{\alpha_n} z_n$ converging to $R_* z$ as $n \to \infty$.

2.3.4 Generalized Tikhonov Regularization

The Tikhonov regularization scheme introduced above can be alternatively written in the following form

$$\min_u J(u) \qquad J(u) = 1/2 \|Ku - z\|_{H_2}^2 + \alpha/2 \|u\|_{H_1}^2. \tag{2.21}$$

To see this let us first find the first variation (§A.3.2)

$$\frac{d}{dt} J(u + tv)|_{t=0} = (Ku - z, Kv)_{H_2} + \alpha(u, v)_{H_1}. \tag{2.22}$$

Using the adjoint of K this can be rewritten

$$\frac{d}{dt} J(u + tv)|_{t=0} = (K^*(Ku - z), v)_{H_1} + \alpha(u, v)_{H_1}. \tag{2.23}$$

If u is the minimum of $J(u)$ then the first variation is zero for any v (§A.3.3) i.e

$$K^* K u + \alpha u = K^* z. \tag{2.24}$$

Using the singular system we can write this in the form

$$\sum_j (\sigma_j^2 + \alpha)(u, v_j)_{H_1} v_j = \sum_j \sigma_j (z, u_j)_{H_2} v_j. \tag{2.25}$$

Taking the inner product of both sides with v_l for a given l gives us

$$(\sigma_l^2 + \alpha)(u, v_l)_{H_1} = \sigma_l (z, u_l)_{H_2}. \tag{2.26}$$

and so

$$u = \sum_j (u, v_j)_{H_1} v_j = \sum_j \frac{\sigma_j (z, u_j)_{H_2}}{\sigma_j^2 + \alpha} v_j, \tag{2.27}$$

which is the same as what we had above.

Written in the form (2.21) Tikhonov regularization can be viewed as a scheme for finding a solution which balances between the need for a solution for which Ku is a good fit to z and one that has minimal H_1 norm, with the value of the regularization parameter α determining how much weight is given to each requirement. In a more general form of Tikhonov regularization, the first term in $J(u)$ can be any fidelity term $\rho(Ku, z)$ which some how measures how closely Ku matches z and the second term a general penalty term $R(u)$ which penalizes against certain artifacts in the solution. Generalized Tikhonov regularization is used in the image processing problems seen in Chapter 3.

2.4 Discrete PDEs and Notation

In many situations one has to solve a discrete version of a continuous partial differential equation, because the equation can not be solved analytically or because data is only known at a certain number of discrete locations. A continuous linear boundary value problem in d dimensions is denoted by

$$L^\Omega u(x_1, .., x_d) = f^\Omega(x_1, .., x_d) \text{ for } (x_1, .., x_d) \in \Omega. \tag{2.28}$$

$$L^\Gamma u(x_1, .., x_d) = f^\Gamma(x_1, .., x_d) \text{ for } (x_1, ..x_d) \in \Gamma. \tag{2.29}$$

where Ω is a bounded and open domain in $\mathbb{R}^d$ and Γ is its boundary.

Example

One such example would be Poisson's equation in two dimensions with Dirichlet boundary conditions

$$-\Delta u(x, y) = f^\Omega(x, y) \text{ in } \Omega \tag{2.30}$$

$$u(x, y) = f^\Gamma(x, y) \text{ on } \Gamma \tag{2.31}$$

■

I introduce this example here as it is the classic example of an equation to which multigrid methods can be applied successfully and will be used to illustrate various important concepts in the next two sections.

Similarly a continuous Nonlinear boundary value problem is defined by

$$N^{\Omega}(u(x_1,..,x_d)) = f^{\Omega}(x_1,..,x_d) \text{ for } (x_1,..,x_d) \in \Omega. \tag{2.32}$$

$$L^{\Gamma}u(x_1,..,x_d) = f^{\Gamma}(x_1,..,x_d) \text{ for } (x_1,..x_d) \in \Gamma. \tag{2.33}$$

Example

An example of a nonlinear equation which will be seen in later chapters is the equation

$$-\alpha\nabla.\left(\frac{\nabla u(x,y)}{\sqrt{|\nabla u(x,y)|+\beta}}\right) + u(x,y) = z(x,y), \tag{2.34}$$

with Neumann boundary condition $\frac{\partial u(x,y)}{\partial n} = 0$ on Γ. ∎

There are various ways that a continuous PDE can be discretized, for example using the finite element method or the finite volume method. In our work the domain $\Omega \in \mathbb{R}^2$ is usually rectangular and the values of f known at uniformally distributed points in the domain therefore the most natural discretization method to use is the finite difference method. From now on we restrict our discussion to 2 dimensions, although it easy to extend to higher dimensions. Assuming that $\Omega = (a,b) \times (c,d)$ is rectangular we impose a cartesian grid (or mesh) with grid spacing $h = (b-a)/n$ in the x direction and $k = (d-c)/m$ in the y direction. In a vertex-centered discretization grid points are placed at the vertices of the mesh so that there are $(n+1) \times (m+1)$ grid points including points on the boundary with grid point (i,j) located at $(x_i, y_j) = (ih, jk)$ for $0 \leq i \leq n$ and $0 \leq j \leq m$. In a cell-centered discretization grid points are placed at the centre of the grid cells so that there are $n \times m$ grid points (none lying on the boundary) and grid point (i,j) is located at $(x_i, y_j) = \left(a + \frac{2i-1}{2}h, c + \frac{2j-1}{2}k\right)$ for $1 \leq i \leq n$ and $1 \leq j \leq m$. The interior of the discrete grid is denoted Ω^h and the boundary by Γ^h. Figure 2.1 shows examples of vertex and cell-centered discretizations of a square domain.

Once the grid is in place the operators in the PDE can be approximated locally using Taylor series expansion, for example using the expansions

$$u(x+h,y) = u(x,y) + h\frac{\partial u}{\partial x}(x,y) + \frac{h^2}{2}\frac{\partial^2 u}{\partial x^2}(x,y) + \frac{h^3}{3!}\frac{\partial^3 u}{\partial x^3}(x,y) + \frac{h^4}{4!}\frac{\partial^4 u}{\partial x^4}(\epsilon_+, y) \tag{2.35}$$

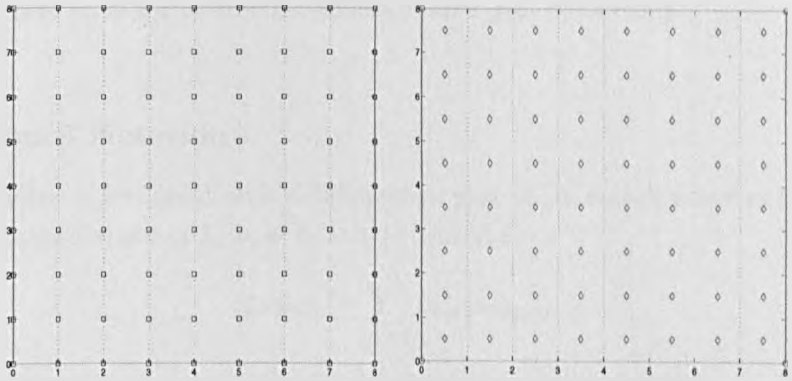
and

$$u(x-h,y) = u(x,y) - h\frac{\partial u}{\partial x}(x,y) + \frac{h^2}{2}\frac{\partial^2 u}{\partial x^2}(x,y) - \frac{h^3}{3!}\frac{\partial^3 u}{\partial x^3}(x,y) + \frac{h^4}{4!}\frac{\partial^4 u}{\partial x^4}(\epsilon_-, y), \tag{2.36}$$

with $x-h < \epsilon_- < x < \epsilon_+ < x+h$, the operator $\frac{\partial u}{\partial x}$ at the grid point (i,j) can be approximated in 3 ways, the first order forward and backward difference operators defined respectively by

$$\frac{\delta_x^+ u_{ij}}{h} = \frac{u_{i+1,j} - u_{ij}}{h} \text{ and } \frac{\delta_x^- u_{ij}}{h} = \frac{u_{i,j} - u_{i-1,j}}{h} \tag{2.37}$$

Figure 2.1: Vertex-centered (left) and cell-centered (right) discretizations of a square domain



or the second order central difference approximation

$$\frac{\delta_x^c u_{ij}}{2h} = \frac{u_{i+1,j} - u_{i-1,j}}{2h}, \tag{2.38}$$

where $u_{i,j} = u(x_i, y_j)$ is the value of u at the grid point (i, j). Approximations to higher order derivatives can be constructed in a similar way for example a second order approximation to $\frac{\partial^2 u}{\partial x^2}$ at (i, j) is given by

$$\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2}. \tag{2.39}$$

The discrete analogue of the continuous problem on the discrete domain is denoted by

$$L_h^\Omega u_h(x_1, .., x_d) = f_h^\Omega(x_1, .., x_d) \text{ for } (x_1, .., x_d) \in \Omega^h. \tag{2.40}$$

$$L_h^\Gamma u_h(x_1, .., x_d) = f_h^\Gamma(x_1, .., x_d) \text{ for } (x_1, ..x_d) \in \Gamma^h. \tag{2.41}$$

where u_h is a grid function on $\Omega^h \cup \Gamma^h$, L_h^Ω and L_h^Γ are operators on the space of grid functions and f_h^Ω and f_h^Γ are discrete analogues of f^Ω and f^Γ. Usually the boundary condition can be eliminated and (2.40) and (2.41) can be written simply as

$$L_h u_h = f_h. \tag{2.42}$$

Example

By way of an example consider Poisson's equation on the unit square with Dirichlet boundary condition. Assume that the domain is discretized using a vertex centered grid with $h = k = 1/n$ then at interior grid points not adjacent to the boundary a second order central difference approximation is given by

$$(L_h u_h)_{ij} = \frac{4u_{i,j} - u_{i+1,j} - u_{i-1,j} - u_{i,j+1} - u_{i,j-1}}{h^2} = f_{i,j}^\Omega. \tag{2.43}$$

At points adjacent to the right boundary, for example, $u_{i+1,j}$ will be replaced by the boundary value $f_{n,j}^\Gamma$, so we have

$$(L_h u_h)_{ij} = \frac{4u_{i,j} - u_{i-1,j} - u_{i,j+1} - u_{i,j-1}}{h^2} = f_{i,j}^\Omega + \frac{f_{n,j}^\Gamma}{h^2}. \tag{2.44}$$

Similar considerations give $L_h u_h$ at other points adjacent to the boundary, therefore we have $L_h u_h = f_h$ where u_h is a grid function on the interior grid points only. ∎

2.4.1 Stencil Notation

Let $p \in \mathbb{Z}^d$ define a grid point on a d-dimensional grid G. In stencil notation the left hand side of the discrete equation $L_h u_h = f_h$ at p is defined by:

$$(L_h u_h)_p = \sum_{q \in \mathbb{Z}^d} L_{p,q}(u_h)_{p+q}. \tag{2.45}$$

The stencil entry $L_{p,q}$ is non-zero when $L_h u_h$ at grid point $p \in G$ is dependant on the value of u_h at grid point $p+q$, the structure of an operator L is defined as all q such that there exists a $p \in G$ such that $L_{p,q}$ is non-zero and is denoted S_L. The stencil for L_h at p is displayed as an array containing all non-zero $L_{p,q}$ e.g a typical stencil in 2 dimensions has the form

$$\begin{bmatrix} 0 & L_{p,(0,1)} & 0 \\ L_{p,(-1,0)} & L_{p,(0,0)} & L_{p,(1,0)} \\ 0 & L_{p,(0,-1)} & 0 \end{bmatrix}$$

Example

Returning to the example we saw above at grid points not adjacent the boundary we can write

$$L_h u_h(x,y) = \frac{1}{h^2} \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} u_h(x,y) = f_h^{\Omega}(x,y) = f_h(x,y) \tag{2.46}$$

and, for example, at points adjacent to the right boundary

$$L_h u_h(x,y) = \frac{1}{h^2} \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & 0 \\ 0 & -1 & 0 \end{bmatrix} u_h(x,y) = f_h^{\Omega}(x,y) + \frac{1}{h^2} f_h^{\Gamma}(1,y) = f_h(x,y). \tag{2.47}$$

∎

2.4.2 Matrix Notation

It may sometimes also be useful to write $L_h u_h = f_h$ in matrix notation. This is done by stacking the grid function u_h into a vector $\mathbf{u}_h$, this is usually done in lexicographical order; u_h is stacked along rows of the grid starting at the bottom left point and ending at the top right. The right hand side vector is stacked in a similar manner into a vector $\mathbf{f}_h$. The discrete linear equation can then be written as $A_h \mathbf{u}_h = \mathbf{f}_h$.

Example

In the example of Poisson's equation considered previously we see that in general row l of A_h will have $a_{ll} = 4/h^2$ and $a_{ll-1} = a_{l,l+1} = a_{l,l+(n-1)} = a_{ll-(n-1)} = -1/h^2$ with all other entries in the row zero, with appropriate modifications for boundary points. $h^2 A_h$ is therefore the $(n-1) \times (n-1)$ block tridiagonal matrix with blocks of size $(n-1) \times (n-1)$ where the off diagonal blocks are the negative Identity and the diagonal blocks are tridiagonal with 4 on the diagonal and -1 on the off diagonals. ∎

2.4.3 Boundary Conditions

So far I have only mentioned Dirichlet Boundary conditions on a vertex centered grid, I briefly now describe how to deal with Neumann Boundary conditions and cell-centered grids.

Neumann for Vertex Centered Grids

Lets assume that we have a Neumann boundary condition $\frac{\partial u}{\partial n} = f^{\Gamma}(x, y)$ on the right boundary of a vertex centered grid. We assume that the equation $L_h^{\Omega} u(x, y) = f_h^{\Omega}(x, y)$ extends to the points on the right boundary. The equation at these grid points will involve 'ghost' grid points outside the domain. These 'ghost points' can be eliminated using the Neumann Boundary condition

$$\frac{u_{n+1,j} - u_{n-1,j}}{2h} = f_{n,j}^{\Gamma}. \tag{2.48}$$

Example

If we take the example of Poissons equation on the unit square then at the right boundary

$$L_h u_h(x, y) = \frac{1}{h^2} \begin{bmatrix} 0 & -1 & 0 \\ -2 & 4 & 0 \\ 0 & -1 & 0 \end{bmatrix} u_h(x, y) = f_h^{\Omega}(x, y) + \frac{2}{h} f_h^{\Gamma}(1, y) = f_h(x, y) \tag{2.49}$$

where u_h now includes points on the right boundary. ∎

Cell-Centered Grids

In the case of a cell-centered grid we have no points on the boundary, so in general the equation at interior points which are adjacent to the boundary will involve ghost points outside of the domain, which need to be eliminated using the boundary condition. If we have a Dirichlet boundary condition at the right boundary, for example we can write it as

$$\frac{1}{2}(u_{n,j} + u_{n+1,j}) = f_{n+1/2,j}^{\Gamma} \tag{2.50}$$

and use it to eliminate the ghost point value $u_{n+1,j}$ from $L_h^\Omega u_h(x,y) = f_h^\Omega(x,y)$. Similarly if we have a Neumann boundary condition at this right boundary we have

$$\frac{u_{n+1,j} - u_{n,j}}{h} = f^\Gamma_{n+1/2,j}. \tag{2.51}$$

2.4.4 Nonlinear Equations

Nonlinear PDEs are treated in much the same way as linear equations, the various operators in the equation are approximated locally on a discrete grid using the finite difference method. The discrete nonlinear equation is denoted by

$$N_h^\Omega\left(u_h(x_1,..,x_d)\right) = f_h^\Omega(x_1,..,x_d) \text{ for } (x_1,..,x_d) \in \Omega. \tag{2.52}$$

$$L_h^\Gamma u(x_1,..,x_d) = f_h^\Gamma(x_1,..,x_d) \text{ for } (x_1,..x_d) \in \Gamma. \tag{2.53}$$

Again the boundary conditions are usually eliminated and the discrete Nonlinear equation written simply as

$$N_h(u_h) = f_h. \tag{2.54}$$

It may be possible to write the Nonlinear equation in a stencil Notation, but some of the stencil entries will depend on u_h. Similarly if a matrix notation is used we will have something of the form $A_h(\mathbf{u}_h)\mathbf{u}_h = \mathbf{f}_h$.

The discretization of the specific nonlinear operator in (2.34) which is used in our work will be detailed in §3.5.

For more on finite difference methods for partial differential equations see for example [72, 88].

2.5 Basic Iterative Methods

This section introduces a class of iterative methods for solving a linear system of equations

$$A\mathbf{x} = \mathbf{b}. \tag{2.55}$$

Where $\mathbf{x}$ is of size N and A is an $N \times N$ matrix. These Iterative methods, which are known as the basic or stationary iterative methods start with some initial approximation $\mathbf{x}^{(0)}$ and generate a sequence $\left\{\mathbf{x}^{(k)}\right\}_{k=1}^{\infty}$ via the relation

$$\mathbf{x}^{(k)} = T\mathbf{x}^{(k-1)} + \mathbf{c}. \tag{2.56}$$

The iteration matrix T and the vector $\mathbf{c}$ come from a splitting $A = M - N$ of the matrix A where M is nonsingular. With this splitting the original system (2.55) can then be written

$$\begin{aligned} &(M-N)\mathbf{x} = \mathbf{b} \\ \iff\ &M\mathbf{x} = N\mathbf{x} + \mathbf{b} \\ \iff\ &\mathbf{x} = M^{-1}N\mathbf{x} + M^{-1}\mathbf{b} \\ \iff\ &\mathbf{x} = T\mathbf{x} + \mathbf{c} \end{aligned} \tag{2.57}$$

where $T = M^{-1}N = I - M^{-1}A$ and $\mathbf{c} = M^{-1}\mathbf{b}$.

Each application of an iterative method which updates $\mathbf{x}^{(k-1)}$ to $\mathbf{x}^{(k)}$ is known as an iteration or a relaxation sweep.

Remark 2.5.1 *I introduce the basic iterative methods in the context of solving a general linear system of equations, using for ease of presentation a matrix notation, however I mainly use them for solving discrete Linear PDEs discretized on structured grids and later I discuss the implementation of these methods in such cases, where a grid function notation may be more useful.*

2.5.1 The Jacobi Method

The first iterative method considered is the Jacobi Method. The Jacobi Method consists of solving the *ith* equation of $A\mathbf{x} = \mathbf{b}$ for x_i to get

$$x_i = \sum_{\substack{j=1 \\ j \neq i}}^{N} \left(\frac{-a_{ij}x_j}{a_{ii}} \right) + \frac{b_i}{a_{ii}} \text{ for } i = 1, .., N. \tag{2.58}$$

Then given $\mathbf{x}^{(k-1)}$ for $k \geq 1$, $x_i^{(k)}$ is generated by:

$$x_i^{(k)} = \sum_{\substack{j=1 \\ j \neq i}}^{N} \left(\frac{-a_{ij}x_j^{(k-1)}}{a_{ii}} \right) + \frac{b_i}{a_{ii}}. \tag{2.59}$$

Note that we require $a_{ii} \neq 0$ for each $i = 1, .., N$. If one or more $a_{ii} = 0$ and the system is nonsingular then a reordering can be performed so that no a_{ii} equals 0. To write $A\mathbf{x} = \mathbf{b}$ in the form $\mathbf{x} = T\mathbf{x} + \mathbf{c}$ we write A as $A = D - L - U$ where D is a diagonal matrix whose diagonal is the same as that of A, $-L$ is the strictly lower triangular part of A and $-U$ is the strictly upper triangular part of A. We then have

$$\begin{aligned} &(D - L - U)\mathbf{x} = \mathbf{b} \\ &\iff D\mathbf{x} = (L + U)\mathbf{x} + \mathbf{b} \\ &\iff \mathbf{x} = D^{-1}(L + U)\mathbf{x} + D^{-1}\mathbf{b} \end{aligned} \tag{2.60}$$

i.e we use matrix splitting $A = M - N$ where $M = D$ and $N = L + U$. The matrix form of the Jacobi Method is then given by:

$$\mathbf{x}^k = T_J\mathbf{x}^{k-1} + \mathbf{c}_J. \tag{2.61}$$

where $T_J = D^{-1}(L + U)$ and $\mathbf{c}_J = D^{-1}\mathbf{b}$.

Algorithm for Jacobi Method

The Jacobi Algorithm for finding an approximate solution to $A\mathbf{x} = \mathbf{b}$ given an initial approximation $\mathbf{x}^{(0)}$ is given below (Algorithm 1). A maximum number of iterations MAX to be performed and a tolerance TOL to stop the algorithm must be specified.

Algorithm 1 Jacobi Method

$$\mathbf{x} = JAC(A, \mathbf{b}, \mathbf{x}^{(0)}, MAX, TOL)$$

1. Set $k = 1$, n =size($\mathbf{x}^{(0)}$)
2. While $k \leq MAX$ do steps 3-5
3. For $i = 1, ..., n$

$$\text{set } x_i = \frac{1}{a_{ii}} \left(- \sum_{\substack{j=1 \\ j \neq i}}^{n} \left(a_{ij} x_j^{(0)} \right) + b_i \right)$$

4. If $\left\| \mathbf{x} - \mathbf{x}^{(0)} \right\| < TOL$ then STOP.
5. Set $k = k + 1$, $\mathbf{x} = \mathbf{x}^{(0)}$
6. OUTPUT Maximum number of iterations exceeded STOP

Weighted Jacobi Method

In the Weighted Jacobi Method, given the current approximation $\mathbf{x}^{(k-1)}$ the new Jacobi iterates are computed using

$$x_i^* = \sum_{\substack{j=1 \\ j \neq i}}^{N} \left(\frac{-a_{ij} x_j^{(k-1)}}{a_{ii}} \right) + \frac{b_i}{a_{ii}} \tag{2.62}$$

for $i = 1, ..., N$ as before, however $\mathbf{x}^*$ is now just an intermediate value. The new approximation $\mathbf{x}^{(k)}$ is given by:

$$\mathbf{x}^{(k)} = (1 - \omega)\mathbf{x}_i^{(k-1)} + \omega \mathbf{x}^* \tag{2.63}$$

where ω is a weighting factor to be chosen. Of course when $\omega = 1$ we have the original Jacobi Method. In matrix form the weighted Jacobi Method is:

$$\mathbf{x}^{(k)} = ((1 - \omega)I + \omega T_J)\mathbf{x}^{(k-1)} + \omega D^{-1}\mathbf{b}, \tag{2.64}$$

which is equivalent to

$$\mathbf{x}^{(k)} = T_\omega \mathbf{x}^{(k-1)} + \mathbf{c}_\omega \tag{2.65}$$

where $T_\omega = (1-\omega)I + \omega D^{-1}(L+U)$ and $\mathbf{c}_\omega = \omega D^{-1}\mathbf{b}$. Weighting of the Jacobi method can be important, when it is used within a multigrid framework, see §2.6.4 for an example of this.

2.5.2 Gauss Seidel Method

When computing $x_i^{(k)}$ in the Jacobi Method we have already computed $x_1^{(k)}, \ldots, x_{i-1}^{(k)}$ which should be better approximations to $x_1, \ldots, x_{i-1}$ than $x_1^{(k-1)}, \ldots, x_{i-1}^{(k-1)}$. Therefore the Jacobi Method should be improved if we rewrite the equation for $x_i^{(k)}$ as

$$x_i^{(k)} = \frac{-\sum_{j=1}^{i-1}\left(a_{ij}x_j^{(k)}\right) - \sum_{j=i+1}^{N}\left(a_{ij}x_j^{(k-1)}\right) + b_i}{a_{ii}} \qquad i = 1, \ldots, N. \tag{2.66}$$

This is known as the Gauss-Seidel Method. Rewriting the above equation as

$$a_{ii}x_i^{(k)} + \sum_{j=1}^{i-1} a_{ij}x_j^{(k)} = -\sum_{j=i+1}^{N}\left(a_{ij}x_j^{(k-1)}\right) + b_i \tag{2.67}$$

we see that the matrix form of the Gauss-Seidel Method is

$$(D-L)\mathbf{x}^{(k)} = U\mathbf{x}^{(k-1)} + \mathbf{b}, \tag{2.68}$$

or equivalently

$$\mathbf{x}^{(k)} = T_{GS}\mathbf{x}^{(k-1)} + \mathbf{c}_{GS} \tag{2.69}$$

where $T_{GS} = (D-L)^{-1}U$ and $\mathbf{c}_{GS} = (D-L)^{-1}\mathbf{b}$. Gauss Seidel is therefore based on a matrix splitting with $M = D - L$ and $N = U$.

Algorithm for Gauss-Seidel

The algorithm for the Gauss-Seidel Method is the same as the algorithm for the Jacobi Method, except that step 3 is replaced by

For $i = 1, \ldots, n$

$$\text{set } x_i = \frac{-\sum_{j=1}^{i-1} a_{ij}x_j - \sum_{j=i+1}^{n} a_{ij}x_j^{(0)} + b_i}{a_{ii}}$$

Backward Gauss-Seidel

In each relaxation sweep of G-S the latest components of the approximation $\mathbf{x}^{(k)}$ are used to update other components of $\mathbf{x}^{(k)}$. The order in which the components are updated therefore becomes significant. A backward G-S method can be defined by calculating the components

of $\mathbf{x}^{(k)}$ in the order $N, N-1, \ldots, 2, 1$ instead of the order $1, \ldots, N$ used earlier (forward G-S) we then have

$$(D-U)\mathbf{x}^{(k)} = L\mathbf{x}^{(k-1)} + \mathbf{b}. \tag{2.70}$$

$$\mathbf{x}^{(k)} = (D-U)^{-1}L\mathbf{x}^{(k)} + (D-U)^{-1}\mathbf{b}. \tag{2.71}$$

A symmetric G-S iteration consists of a forward sweep followed by a backward sweep.

2.5.3 SOR Methods

In Successive Over Relaxation or SOR methods given values $x_i^{(k-1)}$ the intermediate values x_i^* are computed using Gauss Seidel and these values used to evaluate $\mathbf{x}^{(k)}$ as follows:

$$\mathbf{x}^{(k)} = \omega\mathbf{x}^* + (1-\omega)\mathbf{x}^{(k-1)} \tag{2.72}$$

where ω is a positive constant. If $0 < \omega < 1$ this is called under-relaxation and is used to obtain convergence when G-S does not converge. If $\omega > 1$ it is called over-relaxation and it is used to accelerate convergence of systems which are convergent by G-S. SOR is based on the matrix splitting

$$\omega A = (D - \omega L) - (\omega U + (1-\omega)D) \tag{2.73}$$

and can be defined by the recurrence

$$\mathbf{x}^{(k)} = T_{SOR}\mathbf{x}^{(k-1)} + \mathbf{c}_{SOR} \tag{2.74}$$

where $T_{SOR} = (D - \omega L)^{-1}(\omega U + (1-\omega)D)$ and $\mathbf{c}_{SOR} = \omega(D - \omega L)^{-1}\mathbf{b}$.

A backward SOR method can be defined analogously to the backward G-S method. A symmetric SOR or SSOR iteration consists of a forward SOR sweep followed by a backward SOR sweep.

2.5.4 Block Methods

Assume that the vector $\mathbf{x}$ is partitioned into several disjoint sub-vectors (not necessarily of equal size)

$$\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_s)^T. \tag{2.75}$$

Then $A\mathbf{x} = \mathbf{b}$ can be written in the block form

$$\begin{bmatrix} A_{11} & A_{12} & . & A_{1s} \\ A_{21} & A_{22} & . & A_{2s} \\ . & . & . & . \\ A_{s1} & A_{s2} & . & A_{ss} \end{bmatrix} \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ . \\ \mathbf{x}_s \end{pmatrix} = \begin{pmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ . \\ \mathbf{b}_s \end{pmatrix}. \tag{2.76}$$

where the block A_{pq} is of size $n_p \times n_q$ (n_p being the size of $\mathbf{x}_p$) and the vector $\mathbf{b}_p$ is size n_p. Assuming that the diagonal blocks are nonsingular the Jacobi and Gauss-Seidel methods

can easily be extended to the block level. In the Block Jacobi method for $i = 1, \ldots, s$, $\mathbf{x}_i$ is updated as follows:

$$\mathbf{x}_i^{(k)} = A_{ii}^{-1} \left(\sum_{\substack{j=1 \\ j \neq i}}^{s} -A_{ij}\mathbf{x}_j^{(k-1)} + \mathbf{b}_i \right). \tag{2.77}$$

Similarly in the Block Gauss-Seidel method $\mathbf{x}_i$ is updated as

$$\mathbf{x}_i^{(k)} = A_{ii}^{-1} \left(\sum_{j=1}^{i-1} -A_{ij}\mathbf{x}_j^{(k)} + \sum_{j=i+1}^{s} -A_{ij}\mathbf{x}_j^{(k-1)} + \mathbf{b}_i \right). \tag{2.78}$$

Obviously we now have to invert the matrix A_{ii} in order to update $\mathbf{x}_i$ and the larger the vectors $\mathbf{x}_i$ are, the more expensive each step of the method is likely to be, on the other hand the payoff may be faster convergence of the iterative method.

If we define D_B, U_B and L_B as block analogues of D, U and L then the block Jacobi method can be described by the recurrence

$$\mathbf{x}^{(k)} = D_B^{-1}(U_B + L_B)\mathbf{x}^{(k-1)} + D_B^{-1}\mathbf{b} \tag{2.79}$$

and similarly Block Gauss-Seidel by the recurrence

$$\mathbf{x}^{(k)} = (D_B - L_B)^{-1}U_B\mathbf{x}^{(k-1)} + (D_B - L_B)^{-1}\mathbf{b}. \tag{2.80}$$

2.5.5 Convergence

The methods considered in this section all define a sequence of iterates $\mathbf{x}^{(k)} = T\mathbf{x}^{(k-1)} + \mathbf{c}$, which upon convergence produce a solution of the original system $A\mathbf{x} = \mathbf{b}$. In the following it is shown that the iteration $\mathbf{x}^{(k)} = T\mathbf{x}^{(k-1)} + \mathbf{c}$ converges if and only if the spectral radius of T is less than one. First the definition of a convergent matrix is needed.

Definition 2.5.1 (Convergent Matrix)
A square matrix A is said to be convergent if $\lim_{k\to\infty} A^k = 0$.

The following theorem, the proof of which can be found in [81], is also needed.

Theorem 2.5.1 *A matrix A is convergent if and only if $\rho(A) < 1$.*

Finally we need the following lemma.

Lemma 2.5.1 *If the spectral radius $\rho(T) < 1$ then $(I-T)^{-1}$ exists and $(I-T)^{-1} = \sum_{j=0}^{\infty} T^j$*

Proof
If λ is an eigenvalue of T then $(1 - \lambda)$ is an eigenvalue of $(I - T)$. Since $\rho(T) < 1$, 1 is not an eigenvalue of T. Hence 0 is not an eigenvalue of $(I - T)$ and $(I - T)$ is not singular i.e

$(I-T)^{-1}$ exists.

For the second part of the proof let

$$S_m = I + T + T^2 + ... + T^m \tag{2.81}$$

then

$$(I-T)S_m = (I + T + T^2 + ... + T^m) - (T + T^2 + ... + T^{m+1}) = I - T^{m+1}. \tag{2.82}$$

Using Theorem 2.5.1, $\rho(T) < 1$ implies that T is convergent and

$$\lim_{m\to\infty}(I-T)S_m = \lim_{m\to\infty}(I - T^{m+1}) = I. \tag{2.83}$$

Thus

$$(I-T)^{-1} = \lim_{m\to\infty} S_m = \sum_{j=0}^{\infty} T^j. \tag{2.84}$$

and the proof is complete. ∎

The main theorem on convergence can now be proved.

Theorem 2.5.2 *For any $x^{(0)} \in \mathbb{R}^n$ the sequence $\{x^{(k)}\}_{k=0}^{\infty}$ defined by $x^{(k)} = Tx^{(k-1)} + c$ for each $k \geq 1$ converges to the unique solution of $x = Tx + c$ if and only if $\rho(T) < 1$.*

Proof

Assume that $\rho(T) < 1$, we have

$$\begin{aligned} \mathbf{x}^{(k)} &= T\mathbf{x}^{(k-1)} + \mathbf{c} \\ \mathbf{x}^{(k)} &= T(T\mathbf{x}^{(k-2)} + \mathbf{c}) + \mathbf{c} \\ \mathbf{x}^{(k)} &= T^2\mathbf{x}^{(k-2)} + (T-I)\mathbf{c} \\ &\vdots \\ \mathbf{x}^{(k)} &= T^k\mathbf{x}^{(0)} + (T^{k-1} + ... + T^2 + T + I)\mathbf{c}. \end{aligned} \tag{2.85}$$

Using (2.85), Theorem 2.5.1 and the fact that $\rho(T) < 1$ we have

$$\lim_{k\to\infty}\mathbf{x}^{(k)} = \lim_{k\to\infty}\left(\sum_{j=0}^{k-1} T^j\right)\mathbf{c}, \tag{2.86}$$

which by lemma 2.5.1 is equal to $(I-T)^{-1}\mathbf{c}$. The sequence $\{\mathbf{x}^{(k)}\}_{k=0}^{\infty}$ therefore converges to the unique solution of $\mathbf{x} = (I-T)^{-1}\mathbf{c}$ or $\mathbf{x} = T\mathbf{x} + \mathbf{c}$. Conversely assume that $\mathbf{x}^*$ is the unique solution of $\mathbf{x} = T\mathbf{x} + \mathbf{c}$. If $\mathbf{c} = \mathbf{0}$ then $\mathbf{x}^*$ is the unique solution of $\mathbf{x} = T\mathbf{x}$, now let $\mathbf{z} \in \mathbb{R}^n$ be an arbitrary vector and take intial guess $\mathbf{x}^{(0)} = \mathbf{x}^* - \mathbf{z}$, we have

$$\begin{aligned} \lim_{k\to\infty} T^k\mathbf{z} = \lim_{k\to\infty} T^k(\mathbf{x}^* - \mathbf{x}^{(0)}) &= \lim_{k\to\infty} T^{k-1}(T\mathbf{x}^* - T\mathbf{x}^{(0)}) \\ &= \lim_{k\to\infty} T^{k-1}(\mathbf{x}^* - \mathbf{x}^{(1)}) \\ &= \lim_{k\to\infty} T^{k-2}(\mathbf{x}^* - \mathbf{x}^{(2)}) \\ &\vdots \\ &= \lim_{k\to\infty}(\mathbf{x}^* - \mathbf{x}^{(k)}) = 0. \end{aligned} \tag{2.87}$$

Since $\mathbf{z} \in \mathbb{R}^n$ was arbitrary, the matrix T must be convergent, Theorem 2.5.1 then implies that $\rho(T) < 1$, which completes the proof. ∎

Corollary 2.5.1 *If $\|T\| < 1$ for any natural matrix norm and $\mathbf{c}$ is a given vector , then $\{\boldsymbol{x}^{(k)}\}_{k=0}^{\infty}$ defined by $\boldsymbol{x}^{(k)} = T\boldsymbol{x}^{(k-1)} + \mathbf{c}$ converges for any $\boldsymbol{x}^{(0)}$ to a vector $\boldsymbol{x}^*$ which is the unique solution of $\mathbf{x} = T\mathbf{x} + \mathbf{c}$ and the following bounds hold*

$$\|\boldsymbol{x}^* - \boldsymbol{x}^{(k)}\| \leq \|T\|^k \|\boldsymbol{x}^{(0)} - \boldsymbol{x}^*\|. \tag{2.88}$$

and

$$\|\boldsymbol{x}^* - \boldsymbol{x}^{(k)}\| \leq \frac{\|T\|^k}{1 - \|T\|} \|\boldsymbol{x}^{(1)} - \boldsymbol{x}^{(0)}\|. \tag{2.89}$$

Proof

The first part of the corollary follows from the fact that $\rho(T) < \|T\|$ for any matrix norm. To prove the first bound observe that

$$\begin{aligned} \mathbf{x}^* - \mathbf{x}^{(k)} &= T\mathbf{x}^* + \mathbf{c} - (T\mathbf{x}^{(k-1)} + \mathbf{c}) \\ &= T(\mathbf{x}^* - \mathbf{x}^{(k-1)}) \\ &= T^2(\mathbf{x}^* - \mathbf{x}^{(k-2)}) \\ &\vdots \\ &= T^k(\mathbf{x}^* - \mathbf{x}^{(0)}). \end{aligned} \tag{2.90}$$

we therefore have

$$\|\mathbf{x}^* - \mathbf{x}^{(k)}\| = \|T^k(\mathbf{x}^* - \mathbf{x}^{(0)})\| \leq \|T\|^k \|\mathbf{x}^{(0)} - \mathbf{x}^*\|. \tag{2.91}$$

To prove the second bound observe that

$$\mathbf{x}^{(0)} - \mathbf{x}^* = \mathbf{x}^{(0)} - \mathbf{x}^{(1)} + \mathbf{x}^{(1)} - \mathbf{x}^* = \mathbf{x}^{(0)} - \mathbf{x}^{(1)} + T(\mathbf{x}^{(0)} - \mathbf{x}^*). \tag{2.92}$$

Therefore

$$\begin{aligned} \|\mathbf{x}^{(0)} - \mathbf{x}^*\| &\leq \|\mathbf{x}^{(0)} - \mathbf{x}^{(1)}\| + \|T\| \|\mathbf{x}^{(0)} - \mathbf{x}\| \\ (1 - \|T\|)\|\mathbf{x}^{(0)} - \mathbf{x}^*\| &\leq \|\mathbf{x}^{(0)} - \mathbf{x}^{(1)}\| \\ \|\mathbf{x}^{(0)} - \mathbf{x}^*\| &\leq \tfrac{1}{1-\|T\|} \|\mathbf{x}^{(0)} - \mathbf{x}^{(1)}\|. \end{aligned} \tag{2.93}$$

Combining (2.93) with (2.91) gives the desired result. ∎

Convergence factor

Define the error in the approximation $\mathbf{x}^{(k)}$ to the solution of $A\mathbf{x} = \mathbf{b}$ as

$$\mathbf{e}^{(k)} = \mathbf{x}^* - \mathbf{x}^{(k)}. \tag{2.94}$$

where $\mathbf{x}^*$ is the actual solution. The general convergence factor for an iterative method is defined as

$$\rho = \lim_{k\to\infty} \left(\sup_{\mathbf{e}^{(0)}\in\mathbb{R}^n} \frac{\|\mathbf{e}^{(k)}\|}{\|\mathbf{e}^{(0)}\|} \right)^{1/k}. \tag{2.95}$$

Given that

$$\begin{aligned} \mathbf{e}^{(k)} &= \mathbf{x}^* - \mathbf{x}^{(k)} \\ &= T\mathbf{x}^* + \mathbf{c} - (T\mathbf{x}^{(k-1)} + \mathbf{c}) \\ &= T(\mathbf{x}^* - \mathbf{x}^{(k-1)}) \\ &= T\mathbf{e}^{(k-1)} \\ &\vdots \\ &= T^k\mathbf{e}^{(0)}, \end{aligned} \tag{2.96}$$

this is equivalent to

$$\begin{aligned} \rho &= \lim_{k\to\infty} \left(\sup_{\mathbf{e}^{(0)}\in\mathbb{R}^n} \frac{\|T^k\mathbf{e}^{(0)}\|}{\|\mathbf{e}^{(0)}\|} \right)^{1/k} \\ &= \lim_{k\to\infty} \left(\|T^k\| \right)^{1/k} = \rho(T). \end{aligned} \tag{2.97}$$

(using the fact that $\lim_{k\to\infty} \left(\|A^k\| \right)^{1/k} = \rho(A)$ for any matrix norm). Therefore the optimal iterative method is the one whose iteration matrix T has minimal spectral radius.

More Convergence Results

Below some useful theorems on convergence are stated without proofs.

Theorem 2.5.3 *If a matrix A has positive diagonal entries and all other entries negative or zero then only one of the following statements holds*

1. $0 \le \rho(T_{GS}) < \rho(T_J) < 1$
2. $1 < \rho(T_J) < \rho(T_{GS})$
3. $\rho(T_J) = \rho(T_{GS}) = 0$
4. $\rho(T_J) = \rho(T_{GS}) = 1$

where T_J and T_{GS} are the iteration matrices for Jacobi and Gauss-Seidel respectively.

This theorem implies that for such matrices if one of Jacobi or Gauss-Seidel converges then so does the other and similarly divergence of one implies divergence of the other. If both converge then G-S converges faster than Jacobi. For the next theorem we need to define a regular splitting of A.

Definition 2.5.2 (Regular Splitting)
$A = M - N$ is a regular splitting of A if M is nonsingular and M^{-1} and N are nonegative.

Theorem 2.5.4 *If M and N are a regular splitting of A and $T = M^{-1}N$ then $\rho(T) < 1$ if and only if A is nonsingular and A^{-1} is nonegative.*

Theorem 2.5.5 *If all the diagonal elements of A are non-zero then $\rho(T_{SOR}) \geq |\omega - 1|$ and hence SOR converges only when $0 < \omega < 2$.*

Theorem 2.5.6 *If A is positive definite i.e $\boldsymbol{x}^T A\boldsymbol{x} > 0$ for any $\boldsymbol{x}$ and $0 < \omega < 2$ then the SOR method converges for any initial guess $\boldsymbol{x}^{(0)}$.*

Theorem 2.5.7 *If A is positive definite and tridiagonal then $\rho(T_{GS}) = \rho(T_J)^2$ and the optimal ω for SOR is*

$$\omega = \frac{2}{1 + \sqrt{1 - \rho(T_J)^2}} \tag{2.98}$$

for which $\rho(T_{SOR}) = \omega - 1$.

2.5.6 Implementation

If we have a system of equations $A\mathbf{u} = \mathbf{f}$ arising from the discretization of a PDE using the finite difference method on a rectangular domain then the matrix A is likely to be well structured and sparse, which means storage of A will not usually be required. The updating of each entry of $\mathbf{u}$ will typically involve just a few other entries. To illustrate this the implementation of Jacobi and Gauss-Seidel methods is outlined for the case of Poissons equation with Dirichlet boundary conditions on the unit square introduced in §2.4. For ease of presentation I revert to a grid function notation.

Jacobi Method

In the weighted Jacobi Method if grid point (i, j) is not adjacent to the boundary $u_{i,j}$ is updated according to the equation

$$u_{i,j}^k = (1 - \omega)u_{i,j}^{k-1} + \omega \left[\frac{h^2 f_{i,j} + u_{i+1,j}^{k-1} + u_{i-1,j}^{k-1} + u_{i,j+1}^{k-1} + u_{i,j-1}^{k-1}}{4} \right], \tag{2.99}$$

where for example $u_{i+1,j}^{k-1}$ is the entry of the previous approximation u_h^{k-1} corresponding to the grid point $(i + 1, j)$. For points adjacent to the boundary appropriate modifications to (2.99) should be made.

Gauss Seidel Method

Unlike in the Jacobi Method the order in which entries of u_h are updated is significant when using the Gauss-Seidel method. Two different ordering schemes (corresponding to two different ways of stacking u_h into a vector) for Gauss Seidel are outlined below.

Lexicographic Ordering

A lexicographic ordering of the grid points involves ordering the points in increasing order from left to right and up the rows so that the approximation at the bottom left point $(1,1)$ is updated first followed by the approximation at the point $(2,1)$ and so on with the approximation at the top right point $(n-1, m-1)$ updated last. A Gauss-Seidel scheme used with lexicographic ordering is denoted GS-LEX and the entry u_h corresponding to grid point (i,j) (not adjacent to the boundary) is updated as follows

$$u_{i,j}^k = \frac{h^2 f_{i,j} + u_{i-1,j}^k + u_{i,j-1}^k + u_{i+1,j}^{k-1} + u_{i,J+1}^{k-1}}{4}. \tag{2.100}$$

Note that because of the lexicographic ordering entries corresponding to points to the left of and below (i,j) have already been updated whereas entries corresponding to points to the right of and above (i,j) have not.

Red Black Ordering

When a red-black ordering of the grid points is used the grid is coloured in a checkerboard fashion as shown in Figure 2.2, entries of u_h corresponding to the red points are updated first followed by entries of u_h corresponding to the black points. A Gauss Seidel scheme with

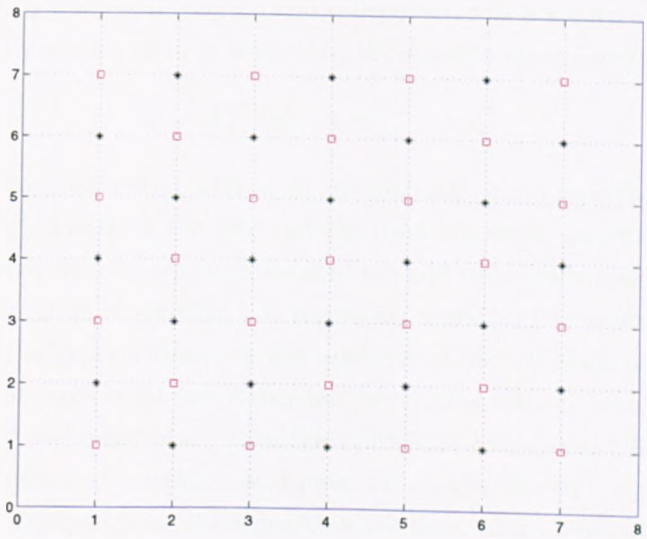


Figure 2.2: Red-Black ordering of grid points: red points are shown as squares, black points are shown as stars

red-black ordering of the grid points is denoted GS-RB. Entries of u_h corresponding to red grid points are updated by

$$u_{i,j}^k = \frac{h^2 f_{i,j} + u_{i-1,j}^{k-1} + u_{i,j-1}^{k-1} + u_{i+1,j}^{k-1} + u_{i,j+1}^{k-1}}{4} \tag{2.101}$$

and then entries corresponding to black points are updated by

$$u_{i,j}^k = \frac{h^2 f_{i,j} + u_{i-1,j}^k + u_{i,j-1}^k + u_{i+1,j}^k + u_{i,j+1}^k}{4}. \tag{2.102}$$

Because a five point approximation to the PDE is being used, the updating of each entry associated with a red point involves only entries associated to black points and vice versa. This means that after each sweep of GS-RB the residual $r_h = f_h - L_h u_h$ is zero at the black points. When each red point is updated using only black points and vice-versa, GS-RB has an advantage over GS-LEX in terms of paralell computing since all the entries of u_h corresponding to red points can be computed in paralell followed by all entries of u_h corresponding to black points. Note that because points are updated in different orders, one step of GS-LEX will not produce an identical result to one step of GS-RB with the same initial guess.

Line Relaxation

If u_h is stacked into a vector $\mathbf{u}$ lexicographically and $\mathbf{u}$ split into $(n-1)$ subvectors each of size $(n-1)$ then the subvector $\mathbf{u}_l$ will contain all the values of u_h corresponding to row l of the grid, hence performing a block Jacobi or Gauss-Seidel iteration on this block system is equivalent to relaxing a whole row of the grid collectively, this is known as x-line relaxation. In our example the updating of $\mathbf{u}_l$ is done using for example block Gauss-Seidel as follows

$$\mathbf{u}_l^k = A_{ll}^{-1}\left(\mathbf{u}_{l-1}^k + \mathbf{u}_{l+1}^{k-1} + h^2 \mathbf{f}_l\right), \tag{2.103}$$

where A_{ll} is a tridiagonal matrix with 4 on the diagonal and -1 on the off diagonals. If u_h is stacked along columns of the grid and the resulting vector partitioned as above the block relaxation methods relax whole columns of the grid collectively, this is known as y-line relaxation. A sweep of an alternating line relaxation consists of an x-line relaxation sweep followed by a y-line relaxation sweep. A line analogue of the red-black pointwise relaxation for line Gauss-Seidel is the zebra line relaxation; here either rows or columns of the grid are coloured alternately white and black, then the white lines are relaxed followed by the black lines, in most cases the approximation at a point on a white line will depend only on other points on that line and points on the adjacent black lines, hence a parallel implementation of zebra line Gauss-Seidel will be possible.

2.5.7 Local Nonlinear Relaxation Methods

If we have a discrete non-linear PDE $N_h(u_h) = f_h$ on a grid Ω^h which has in total N grid points then we have in general a system of non-linear equations

$$W_i(u_1, u_2, \ldots, u_N) = 0 \ , i = 1, \ldots, N. \tag{2.104}$$

Analogous to the linear case a non-linear Jacboi Iteration involves solving the *ith* equation for the *ith* unknown

$$W_i(u_1^k, u_2^k, .., u_{i-1}^k, u_i^{k+1}, u_{i+1}^k, .., u_N^k) = 0, \tag{2.105}$$

where k denotes the current approximation, $k+1$ denotes the new approximation and we start with some initial guess u^0. Similarly a non-linear Gauss-Seidel iteration is given by

$$W_i(u_1^{k+1}, u_2^{k+1}, \ldots, u_{i-1}^{k+1}, u_i^{k+1}, u_{i+1}^k, \ldots, u_N^k) = 0 \ , i = 1, \ldots, N, \tag{2.106}$$

where of course $u_1, .., u_{i-1}$ are known before u_i is updated. Both these methods will involve solving a non-linear equation in one unknown to update each u_i. This can be done by one step of Newton's method using the current approximation to u_i as initial guess

$$u_i^{k+1} = u_i^k - W_i(u_i^k)/C(u_i^k), \tag{2.107}$$

where

$$C(u_i^k) = \frac{\partial W_i}{\partial u_i}(u_i^k). \tag{2.108}$$

The resulting iterations are known as Jacobi-Newton and Gauss-Seidel-Newton respectively.

In the case where we have a semi-linear system of equations so that at each grid point we have

$$a_1 u_1 + .. + a_N u_N + W_i(u_1, \ldots, u_N) = 0, \tag{2.109}$$

where W is a non-linear equation, the Jacobi-Newton iteration is performed by substituting in u_j^k for $j \neq i$ and then replacing $W_i(u_i^{k+1})$ by

$$W_i(u_i^k) + C(u_i^k)(u_i^{k+1} - u_i^k). \tag{2.110}$$

u_i is then updated by

$$u_i^{k+1} = \frac{-(a_1 u_1^k + \ldots + a_{i-1} u_{i-1}^k + a_{i+1} u_{i+1}^k + \ldots + a_N u_N^k) - W_i(u_i^k) + C(u_i^k)}{a_i + C(u_i^k)}. \tag{2.111}$$

Alternatively we can simply substitute u_i^k into W_i in which case

$$u_i^{k+1} = \frac{-(a_1 u_1^k + \ldots + a_{i-1} u_{i-1}^k + a_{i+1} u_{i+1}^k + \ldots + a_N u_N^k) - W_i(u_1^k, \ldots, u_i^k, \ldots, u_p^k)}{a_i}, \tag{2.112}$$

this is known as the Jacobi-Picard iteration. A GS-Picard iteration is defined in a similar way.

Remark 2.5.2 *As in the case of linear PDEs, we expect that a discrete Nonlinear PDE at a particular grid point will be defined in terms of u at that grid point and a small number of neighbouring points.*

2.6 Multigrid Methods

Multigrid methods, first developed by A. Brandt in the 1970s have been shown to be fast efficient solvers for a range of linear and nonlinear elliptic PDEs discretized on structured and unstructured grids. In this section I give a short introduction to the concepts behind multigrid methods, using the classic example of Poisson's equation on the unit square to illustrate various key concepts and give the basic algorithms for linear and nonlinear multigrid methods. The discussion here is limited to equations defined on cartesian grids and to simple smoothers and grid transfer operators. For a more comprehensive introduction to multigrid methods see for example [12, 92, 102] and references therein and see the next section (§2.7) for a discussion of black box algebraic multigrid methods.

2.6.1 Basic Principles of Multigrid

In the following the two key ingredients of multigrid methods, error smoothing and coarse grid correction are introduced.

Smoothing

Many basic relaxation schemes like the ones seen in the previous section when used to solve discrete elliptic PDEs, discretized on cartesian grids are slow to converge, however they do (if applied appropriately) possess what is known as the smoothing property. These schemes are effective at removing the oscillatory fourier modes of the error in an approximation but may not be effective at removing the smooth modes of the error (leading to the aforementioned slow convergence) i.e they smooth the error while not necessarily reducing its size greatly. A smooth quantity can however be well approximated on a coarser grid, which leads us on to the second principle on which multigrid methods are built.

Coarse Grid Correction

Consider a linear system

$$A\mathbf{u} = \mathbf{f}. \tag{2.113}$$

If $\mathbf{v}$ is an approximation to the solution $\mathbf{u}$ then the error in the approximation $\mathbf{e}$ is defined by:

$$\mathbf{e} = \mathbf{u} - \mathbf{v}. \tag{2.114}$$

Applying A to both sides of (2.114) we get

$$A\mathbf{e} = \mathbf{f} - A\mathbf{v} = \mathbf{r} \tag{2.115}$$

where $\mathbf{r}$ is the residual. This is known as the residual equation. The residual equation gives us a way to relax directly on the error $\mathbf{e}$. Of course solving (2.115) is as expensive as

solving the original equation, however if A is replaced by some simpler approximation $\hat{A}$ an approximation of the error can be found relatively cheaply, used to correct $\mathbf{v}$ and then the process repeated until convergence. If for example we approximate A by its diagonal D we recover the Jacobi method.

Now lets imagine that we have discretized an elliptic PDE on a cartesian grid Ω^h with grid spacing (h, k) and we have (in grid function notation) the linear system

$$L_h u_h = f_h. \tag{2.116}$$

Given an approximation v_h one way to improve v_h would be to solve the residual equation on some coarser grid Ω^H with grid spacing (H, K) (we will use only standard coarsening i,e $(H, K) = (2h, 2k)$ in the following) i.e approximate L_h by a coarse grid analogue L_H. Overall the procedure would be

1. Transfer r_h to the coarse grid using a restriction operator $r_H = I_h^H r_h$
2. Solve $L_H e_H = r_H$
3. Transfer e_H back to the fine grid and correct the approximation $v_h \leftarrow v_h + I_H^h e_H$

Here L_H is usually just the original differential operator discretized on Ω^H. Given that Ω^H has less grid points it is of coarse cheaper to solve $L_H e_H = r_H$ than it is to solve the fine grid equation.

Remark 2.6.1 *A similar approach can be used for Nonlinear operators, using the Nonlinear residual equation, which is introduced later.*

Clearly this approach will only be effective if the error e_h can be well approximated on a coarser grid i.e it is 'smooth'. The combination of iterative methods which are slow to converge but nevertheless smooth the error, with coarse grid correction is the main idea behind multigrid methods.

The next three subsections define more precisely what is meant by coarse grids, restriction and interpolation operators and smoothness, before coarse grid correction is defined more precisely.

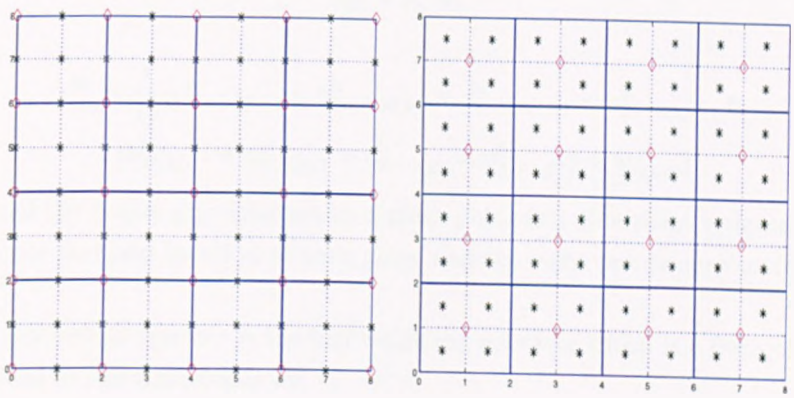
2.6.2 Coarsening

Given that multigrid methods are based on the use of coarse grids to accelerate iterative methods I describe in more detail what is meant by a coarse grid. I assume that we have a cartesian grid Ω^h with grid spacing (h, k) called the fine grid and construct a coarse analogue Ω^H with grid spacing (H, K).

Standard Coarsening

As mentioned above in the case of standard coarsening the coarse grid Ω^H is just Ω^{2h} i.e the grid with grid spacing $(2h, 2k)$. In the case of a vertex centered grid, if Ω^h has $(n+1)\times(m+1)$ grid points including boundary points then Ω^{2h} will have $(n/2+1)\times(m/2+1)$ grid points including boundary points and the coarse grid points will be a subset of the fine grid points, for example the coarse grid point $(1,1)$ located at $(a+2h, c+2k)$ is the same as the fine grid point $(2,2)$. If we have a cell-centered discretization then if the fine grid has $n\times m$ grid points the coarse grid will have $n/2\times m/2$ grid points and unlike in the verterx centered case the coarse grid points will not coincide with fine grid points. See Figure 2.3 for an example of fine and coarse vertex-centered and cell-centered grids.

Figure 2.3: Fine and coarse grids in the vertex centered case (left) and the cell-centered case (right). Coarse grid lines are full, additional fine grid lines are dashed. Stars are fine grid points, diamonds are coarse grid points in the cell-centered case and points which are both coarse and fine in the vertex centered case.



Other Coarsening

Other types of coarsening aside from standard coarsening can be used, for example the grid spacing can be doubled in just one direction e.g $(H, K) = (h, 2k)$ this is known as semi-coarsening. Semi-coarsening is used in anisotropic problems where pointwise smoothers smooth the error in only one direction.

2.6.3 Intergrid Transfers

We also need some way of transferring grid functions between grids. Transferring grid functions from a fine to a coarse grid is known as restriction and transferring grid functions from a coarse to a fine grid is called interpolation (or prolongation). In the following we consider

only the transfer between standard coarsened fine and coarse grids, in the vertex and cell-centered cases. An interpolation operator which transfers a grid function v_h from a grid Ω^{2h} to a grid Ω^h is denoted by I_{2h}^h and a restriction operator which transfers grid functions from a grid Ω^h to a grid Ω^{2h} is denoted by I_h^{2h}.

Restriction for Vertex-Centered Grids

The most obvious restriction operator is injection which is defined in 2 dimensions by

$$v_{2h} = I_h^{2h} v_h, \tag{2.117}$$

where

$$v_{i,j}^{2h} = v_{2i,2j}^h. \tag{2.118}$$

i.e the coarse grid function at a mesh point takes its value directly from the corresponding fine grid value. An alternative restriction operator is the full weighting operator which is defined in 2 dimensions by

$$v_{2h} = I_h^{2h} v_h, \tag{2.119}$$

where

$$\begin{aligned} v_{i,j}^{2h} = \frac{1}{16}[&v_{2i-1,2j-1}^h + v_{2i-1,2j+1}^h + v_{2i+1,2j-1}^h + v_{2i+1,2j+1}^h + \\ &2\left(v_{2i,2j-1}^h + v_{2i,2j+1}^h + v_{2i-1,2j}^h + v_{2i+1,2j}^h\right) + 4v_{2i,2j}^h]. \end{aligned} \tag{2.120}$$

i.e the value of the coarse grid function at a mesh point is a nine point weighted average of the value of the fine grid function at that point and the eight points surrounding it on the fine grid.

Another restriction operator is the half weighting operator which is a five point weighted average, defined in two dimensions by

$$v_{2h} = I_h^{2h} v_h, \tag{2.121}$$

where

$$v_{i,j}^{2h} = \frac{1}{8}\left[v_{2i,2j-1}^h + v_{2i,2j+1}^h + v_{2i-1,2j}^h + v_{2i+1,2j}^h + 4v_{2i,2j}^h\right]. \tag{2.122}$$

Stencil Notation

Before continuing I make a brief digression to introduce a stencil notation for restriction and interpolation operators. Using the notation introduced earlier (§2.4.1) we see that a restriction operator $R = I_h^{2h}$ which maps grid functions on Ω^h to grid functions on Ω^{2h} can be written in stencil notation as follows:

$$(Rv_h)_p = \sum_{q \in \mathbb{Z}^2} R_{p,q}(v_h)_{2p+q} \text{ for } p \in \Omega^{2h}, \tag{2.123}$$

where p is a grid point on the two-dimensional grid Ω^{2h} and $R_{p,q}$ ($q \in \mathbb{Z}^2$) defines the weight given to the value of v_h at the fine grid point $2p+q$, in the calculation of the value of v_{2h} at p. For example in the case of full weighting restriction defined above we have $R_{p,(0,0)} = 1/4$, $R_{p,(1,0)} = R_{p,(-1,0)} = R_{p,(0,1)} = R_{p,(-1,0)} = 1/8$ and $R_{p,(1,1)} = R_{p,(-1,1)} = R_{p,(1,-1)} = R_{p,(-1,-1)} = 1/16$ so the stencil is

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}_h^{2h} \tag{2.124}$$

similarly the stencil for injection is $[1]_h^{2h}$ and the stencil for half weighting is:

$$\frac{1}{8}\begin{bmatrix} 0 & 1 & 0 \\ 1 & 4 & 1 \\ 0 & 1 & 0 \end{bmatrix}_h^{2h} \tag{2.125}$$

If L is an operator mapping U to V where U is the space of grid functions on a grid G_1 and V is the space of grid functions on a grid G_2 its adjoint (or transpose) operator with respect to the Euclidean inner product, L^*, which maps V into U satisfies

$$(Lu, v)_2 = (u, L^*v)_2 \tag{2.126}$$

for all $u \in U$ and $v \in V$, where $(.,.)_2$ denotes the Euclidean inner product. The transpose of every restriction operator is a prolongation operator. If we consider

$$(Rv_h, v_{2h}) = \sum_{p\in\mathbb{Z}^2}\left[\sum_{q\in\mathbb{Z}^2} R_{p,q}(v_h)_{2p+q}\right](v_{2h})_p, \tag{2.127}$$

making the change of variables $s = 2p+q$ we have

$$(Rv_h, v_{2h}) = \sum_{p\in\mathbb{Z}^2}\sum_{s\in\mathbb{Z}^2} R_{p,s-2p}(v_h)_s(v_{2h})_p = \sum_{s\in\mathbb{Z}^2}(v_h)_s\sum_{p\in\mathbb{Z}^2} R_{p,s-2p}(v_{2h})_p = (v_h, R^*v_{2h}) \tag{2.128}$$

and we see that

$$(R^*v_{2h})_s = \sum_{p\in\mathbb{Z}^2} R_{p,s-2p}(v_{2h})_p \tag{2.129}$$

so a prolongation operator P can be written in stencil notation in terms of its transpose

$$(Pv_{2h})_p = \sum_{q\in\mathbb{Z}^2} P^*_{q,p-2q}(v_{2h})_q. \tag{2.130}$$

Given a rule to determine Pv_{2h} the value $P^*_{s,t}$ can be obtained by applying P to the grid function on Ω^{2h} which is zero everywhere except at the point s ,which I shall denote by δ^s_{2h}, then

$$(P\delta^s_{2h})_p = P^*_{s,p-2s} \text{ or } P^*_{s,t} = (P\delta^s_{2h})_{2s+t}. \tag{2.131}$$

Interpolation for Vertex-Centered Grids

The most commonly used interpolation operator is bilinear interpolation which is defined by

$$v_h = I_{2h}^h v_{2h}, \tag{2.132}$$

where

$$\begin{aligned} v_{2i,2j}^h &= v_{i,j}^{2h} \\ v_{2i+1,2j}^h &= \tfrac{1}{2}\left(v_{i,j}^{2h} + v_{i+1,j}^{2h}\right) \\ v_{2i,2j+1}^h &= \tfrac{1}{2}\left(v_{i,j}^{2h} + v_{i,j+1}^{2h}\right) \\ v_{2i+1,2j+1}^h &= \tfrac{1}{4}\left(v_{i,j}^{2h} + v_{i+1j}^{2h} + v_{i,j+1}^{2h} + v_{i+1,j+1}^{2h}\right) \\ &\text{for } 0 \le i \le n/2-1 \text{ and } 0 \le j \le m/2-1. \end{aligned} \tag{2.133}$$

This means that for fine grid points which are also coarse grid points the value of the fine grid function is transferred directly from the coarse grid value. For fine grid points on a horizontal coarse grid line but not a vertical one the fine grid value is the average of the values at the 2 coarse grid points either side of it on that line, with the analogous result for fine grid points on a vertical coarse grid line but not a horizontal one. For fine grid points in the middle of four coarse grid points the fine grid value is the average of the coarse grid values at the 4 points. From the first line in (2.133) we have

$$(I_{2h}^h)_{s,(0,0)}^* = (I_{2h}^h \delta_{2h}^s)_{2s} = 1 \tag{2.134}$$

with $s = (i, j)$. From the second line we have

$$(I_{2h}^h)_{s,(-1,0)}^* = (I_{2h}^h \delta_{2h}^s)_{2s+(-1,0)} = 1/2. \tag{2.135}$$

with $s = (i+1, j)$. Similar considerations for other points give the stencil for $(I_{2h}^h)^*$ as

$$(I_{2h}^h)^* = \frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}_{2h}^{h} \tag{2.136}$$

This is sometimes written as

$$I_{2h}^h = \frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}_{2h}^{h} \tag{2.137}$$

A nice way to think about this notation is to imagine that the stencil is centered on a fine grid point. The fine grid value at that point is given by the sum of the stencil entries which lie on a coarse grid point multiplied by the coarse grid value at that point. For example if the fine grid point is also a coarse grid point, then only the middle entry 1 lies on a coarse grid point and therefore as defined above the fine grid value is transferred directly from the coarse grid.

We see from (2.136) that up to a constant the bilinear interpolation operator is the transpose of the full-weighting restriction operator.

Restriction for Cell-Centered Grids

If we use a cell-centered discretization each cell of the coarse grid Ω^{2h} contains within it 4 fine grid cells and each mesh point of Ω^{2h} is surrounded by 4 mesh points of Ω^h. The four cell average restriction operator evaluates the value of a coarse grid function v_{2h} at a coarse grid point by taking the average value of the fine grid function v_h at the four fine grid points surrounding it. This restriction operator can be defined formally by

$$v_{2h} = I_h^{2h} v_h, \tag{2.138}$$

where

$$v_{i,j}^{2h} = \frac{1}{4}\left(v_{2i-1,2j-1}^h + v_{2i-1,2j}^h + v_{2i,2j-1}^h + v_{2i,2j}^h\right). \tag{2.139}$$

Which can be represented by the stencil

$$\frac{1}{4}\begin{bmatrix} 1 & & 1 \\ & \cdot & \\ 1 & & 1 \end{bmatrix}_h^{2h}. \tag{2.140}$$

Remark 2.6.2 *In the case of cell-centered grids the coarse points are not a subset of the fine grid points. The point 2p is the fine grid point northeast of the coarse grid point p, hence the stencil entry $(p,(0,0))$ is positioned northeast of the centre of the stencil.*

Interpolation for Cell-Centered Grids

The simplest cell-centered interpolation operator simply transfers the value at a coarse grid point directly to the four fine grid points contained within that coarse grid cell.

$$v_h = I_{2h}^h v_{2h}, \tag{2.141}$$

where

$$v_{2i,2j}^h = v_{2i,2j-1}^h = v_{2i-1,2j}^h = v_{2i-1,2j-1}^h = v_{i,j}^{2h} \tag{2.142}$$

for $i = 1,..,n/2$ and $j = 1,..,m/2$. The stencil is

$$\begin{bmatrix} 1 & & 1 \\ & \cdot & \\ 1 & & 1 \end{bmatrix}_{2h}^{h}. \tag{2.143}$$

Again note that up to a constant this interpolation operator is the transpose of the cell-centered restriction operator.

The cell-centered bilinear interpolation operator is defined as follows:

$$v_h = I_{2h}^h v_{2h}, \tag{2.144}$$

where

$$v^h_{2i,2j} = \frac{1}{16}\left[9v^{2h}_{i,j} + 3(v^{2h}_{i+1,j} + v^{2h}_{i,j+1}) + v^{2h}_{i+1,j+1}\right]. \tag{2.145}$$

$$v^h_{2i+1,2j} = \frac{1}{16}\left[9v^{2h}_{i+1,j} + 3(v^{2h}_{i,j} + v^{2h}_{i+1,j+1}) + v^{2h}_{i,j+1}\right]$$

$$v^h_{2i,2j+1} = \frac{1}{16}\left[9v^{2h}_{i,j+1} + 3(v^{2h}_{i,j} + v^{2h}_{i,j+1}) + v^{2h}_{i+1,j}\right]$$

$$v^h_{2i+1,2j+1} = \frac{1}{16}\left[9v^{2h}_{i+1,j+1} + 3(v^{2h}_{i+1,j} + v^{2h}_{i,j+1}) + v^{2h}_{i,j}\right]$$

for $i = 1, ..., n/2 - 1$ and $j = 1, ..., m/2 - 1$.
In stencil notation this is

$$\frac{1}{16}\begin{bmatrix} 1 & 3 & 3 & 1 \\ 3 & 9 & 9 & 3 \\ 3 & 9 & 9 & 3 \\ 1 & 3 & 3 & 1 \end{bmatrix}^{h}_{2h} \tag{2.146}$$

Order of Interpolation and Restriction

An Interpolation operator is said to have order $k + 1$ if it can transfer exactly polynomials of order k i.e if the exact values of a polynomial are given at the coarse grid points, the exact value of the polynomial can be found at all fine grid points by interpolating with the given operator. The order of a restriction operator is equal to the order of it's transpose. Bilinear interpolation in both the vertex and cell-centered cases has order 2, which means the full weighting restriction operator also has order 2. The cell-centered interpolation operator defined by (2.141) has order 1 and hence so does the cell-centered restriction operator. The transpose of injection cannot even transfer constants and so its order is zero.

When constructing multigrid methods the order of the restriction operator + the order of the interpolation operator, should be, as a general rule, greater than the order of the PDE being considered.

2.6.4 Smoothing Analysis

I now discuss what precisely is meant by smooth and oscillatory modes of the error with respect to standard coarsening and by way of an example apply some smoothing analysis (a concept invented by A.Brandt 1977) to the weighted Jacobi Method for the model problem of poissons equation with Dirichlet boundary conditions on the unit square, seen previously, before introducing the method of Local Fourier Analysis.

Fourier Expansion of the Error

Consider a vertex centered discretization of the unit square Ω^h with $h = k = 1/n$, represented by the discrete points $(x_i, y_j) = (ih, jk) = (i/n, j/n)$ for $1 < i, j < n-1$. Let the grid function

$e_h(x,y)$ be the error in the approximation to some discrete PDE and assume that we have a Dirichlet boundary condition, then e_h can be written as a discrete fourier sine series as follows:

$$e_h(x,y) = \sum_{l,m=1}^{n-1} \alpha^{l,m} \sin l\pi x \sin m\pi y \tag{2.147}$$

for $(x,y) \in \Omega^h$. The sine series is appropriate here because we have a Dirichlet boundary condition. Clearly if we have a Dirichlet boundary condition the error in any approximation is zero on the boundary (because we are given the boundary values) so we need $e_h(x_i,y_j) = 0$ for $i,j = 0$ or n, substituting $x = 0$ or $x = 1$ into (2.147) the term $\sin l\pi x = 0$ for all l similarly for $y = 0$ or 1. If we have a periodic boundary condition we can write (assuming n even)

$$e_h(x,y) = \sum_{l,m=-n/2}^{n/2-1} \alpha^{l,m} e^{l2\pi i x} e^{m2\pi i y} \tag{2.148}$$

for $(x,y) = (i/n, j/n)$ and $0 \le i,j \le n-1$. It is clear in this case that $e_h(x+1,y) = e_h(x,y)$ and similarly $e_h(x,y+1) = e_h(x,y)$. For a Neumann boundary condition we must have $e_h(-1/n,y) = e_h(1/n,y)$ and similar conditions at the other boundaries therefore we should use a cosine series to expand e_h

$$e_h(x,y) = \sum_{l,m=0}^{n} \alpha^{l,m} \cos l\pi x \cos m\pi y \tag{2.149}$$

for $(x,y) = (i/n, j/n)$ and $0 \le i,j \le n$. If we have a cell-centered discretization, then essentially we have the same expansions, except that the grid points (x_i,y_j) are in different positions and in the case of Dirichlet and Neumann boundary conditions the number of grid points is different and hence the range of l,m will need to change.

High and Low Frequencies

Returning to the case of the vertex-centered grid with Dirichlet boundary conditions let us denote $\sin l\pi x \sin m\pi y$ by $\psi_h^{l,m}$. Now consider a standard coarsened, coarse grid Ω^{2h}. On this grid $x = 2i/n$ $y = 2j/n$ for $1 \le i,j \le n/2-1$. Let us write $\psi_h^{n-l,m}$ as

$$\psi_h^{n-l,m} = \sin(n-l)\pi x \sin m\pi y = (\sin n\pi x \cos l\pi x - \sin l\pi x \cos n\pi x) \sin m\pi y, \tag{2.150}$$

we see that for $(x,y) \in \Omega^{2h}$

$$\sin n\pi x = \sin 2\pi i = 0. \tag{2.151}$$

and

$$\cos n\pi x = \cos 2\pi i = 1, \tag{2.152}$$

so

$$\psi_h^{n-l,m} = -\sin l\pi x \sin m\pi y = -\psi_h^{l,m}. \tag{2.153}$$

Similar considerations show that

$$\psi_h^{l,n-m} = -\psi_h^{l,m} \tag{2.154}$$

$$\psi_h^{n-l,n-m} = \psi_h^{l,m} \tag{2.155}$$

for $(x,y) \in \Omega^{2h}$ i.e $\psi_h^{l,m}$, $\psi_h^{n-l,m}$,$\psi_h^{l,n-m}$ and $\psi^{n-l,n-m}$ cannot be distinguished on Ω^{2h}, this phenomenon is known as aliasing. Clearly if for $l,m < n/2$ the functions $\psi_h^{n-l,m}$,$\psi_h^{l,n-m}$ and $\psi^{n-l,n-m}$ do not represent meaningful grid functions on Ω^{2h} then any coarse grid correction procedure, which utilises Ω^{2h} cannot reduce the part of the error which corresponds to these fourier components, therefore $\psi_h^{l,m}$ is defined to be

$$\text{low frequency if } \max(l,m) < n/2 \tag{2.156}$$

$$\text{high frequency if } n/2 \leq \max(l,m) < n. \tag{2.157}$$

The low frequency components of the error can be eliminated using coarse grid correction, while the high frequency components cannot and must be reduced using a smoother. If it is said that a relaxation scheme smooths the error after a few steps it means that the high frequency or oscillatory components of the error become small after a few iterations.

Remark 2.6.3 *The definition of high and low frequency components is of course related to the choice of the coarse grid. If the fine grid is coarsened only in the y direction, for example, the above definition of high frequency components is replaced by*

$$\psi_h^{l,m} \text{ such that } n/2 \leq m < n. \tag{2.158}$$

Smoothing Analysis for Weighted Jacobi Method

Here I use the above expansion of the error on a vertex-centered grid with Dirichlet boundary conditions and definition of high and low frequencies to illustrate the smoothing effect of the weighted Jacobi method on Poisson's equation (2.30)-(2.31). This example is adapted from [92].

Theorem 2.6.1 *The functions*

$$\psi_h^{l,m} = \sin l\pi x \sin m\pi y \ (l,m = 1,...,N-1) \tag{2.159}$$

are the discrete eigenfunctions of the operator $L_h = -\Delta_h$.

Proof

$$\begin{aligned} L_h\psi_h^{l,m} = \tfrac{1}{h^2}[4\sin l\pi x \sin m\pi y - \sin l\pi(x+h)\sin m\pi y \\ - \sin l\pi(x-h)\sin m\pi y - \sin l\pi x \sin m\pi(y+h) - \sin l\pi x \sin m\pi(y-h)] \end{aligned} . \tag{2.160}$$

Given that $\sin(a \pm b) = \sin(a)\cos(b) \pm \sin(b)\cos(a)$ and therefore $\sin(a+b) + \sin(a-b) = 2\sin(a)\cos(b)$ we have

$$L_h \psi_h^{l,m} = \frac{1}{h^2}\left[4 \sin l\pi x \sin m\pi y - 2\sin l\pi x \cos l\pi h \sin m\pi y - 2 \sin l\pi x \sin m\pi y \cos m\pi h\right] \tag{2.161}$$

or

$$L_h \psi_h^{l,m} = \frac{1}{h^2}\left[4\psi_h^{l,m} - 2\cos(l\pi h)\psi_h^{l,m} - 2\cos(m\pi h)\psi_h^{l,m}\right]. \tag{2.162}$$

So $\psi_h^{l,m}$ are the eigenvectors of L_h and the eigenvalues are

$$\sigma_h^{l,m} = \frac{2}{h^2}(2 - \cos(l\pi h) - \cos(m\pi h)). \tag{2.163}$$

■

From (2.99) we see that the weighted Jacobi method can be written as

$$u_h^k = S_h(\omega) u_h^{(k-1)} + \frac{\omega h^2}{4} f_h, \tag{2.164}$$

where the smoothing operator S_h is represented by the stencil

$$S_h(\omega) = \frac{\omega}{4}\begin{bmatrix} & 1 & \\ 1 & 4(1/\omega - 1) & 1 \\ & 1 & \end{bmatrix} = I_h - \frac{\omega h^2}{4} L_h. \tag{2.165}$$

The eigenfunctions of $S_h(\omega)$ are therefore the same as those of L_h and the eigenvalues are

$$\lambda_h^{l,m} = 1 - \frac{\omega h^2}{4}\sigma_h^{l,m} = 1 - \frac{\omega}{2}(2 - \cos l\pi h - \cos m\pi h) \tag{2.166}$$

$$m, l = 1, ..., n-1.$$

Recall that $h = 1/n$, so provided $n > 2$ the term $\cos l\pi h$ is monotonically decreasing from $\cos \pi h$ at $l = 1$ to $-cos\pi h$ at $l = n-1$. If we assume that n is relatively large we can (using $\sin(s) \approx s$ and $\sqrt{1-s} \approx 1 - s/2$ for s small) approximate $\cos \pi h$ as $1 - \frac{\pi^2 h^2}{2}$. We then have

$$\lambda^{1,1} \approx 1 - \omega\left(\frac{\pi^2 h^2}{2}\right) \tag{2.167}$$

$$\lambda^{n-1,n-1} \approx 1 - 2\omega + \frac{\pi^2 h^2}{2}. \tag{2.168}$$

We see that as the grid spacing $h \to 0$ there is no convergence for $\omega > 1$ since the spectral radius is greater than 1. If $0 < \omega \leq 1$ we have

$$\rho(S_h(\omega)) = |\lambda_h^{1,1}| = 1 - O(\omega h^2). \tag{2.169}$$

In terms of convergence we see that $\omega = 1$ is the best choice of the weighting parameter since this corresponds to the smallest spectral radius, however to achieve reasonable smoothing

with weighted Jacobi we must choose a parameter $\omega \neq 1$. To see this consider approximations before v_h and after $\hat{v}_h$ a relaxation step. Define the error in these approximations as

$$e_h = u_h - v_h \tag{2.170}$$

$$\hat{e}_h = u_h - \hat{v}_h, \tag{2.171}$$

where u_h satisfies $u_h = S_h u_h + \frac{\omega h^2}{4} f_h$. Now expand e_h in terms of the eigenfunctions

$$e_h = \sum_{l,m=1}^{N-1} \alpha^{l,m} \psi_h^{l,m}. \tag{2.172}$$

We know that $\hat{v}_h = S_h v_h + \frac{\omega h^2}{4} f_h$ so

$$\hat{e}_h = S_h e_h = \sum_{l,m=1}^{N-1} \lambda_h^{l,m} \alpha^{l,m} \psi_h^{l,m}. \tag{2.173}$$

From this we see that the component of the error corresponding to the smoothest eigenfunction $\psi_h^{1,1}$ is responsible for the slow convergence of the Jacobi Method because $\lambda_h^{1,1}$ is close to 1. Furthermore the smaller the grid spacing h the larger $\lambda_h^{1,1}$ is and the worse the convergence of smooth error components.

Having established that no choice of ω will effectively damp the low frequency components lets turn our attention to the high frequency components of the error. The smoothing factor for the weighted Jacobi Method $\mu(h;\omega)$ of S_h is defined as the worst factor by which high frequency components are reduced per relaxation step

$$\mu(h;\omega) = \max\{|\lambda_h^{l,m}| : n/2 \leq \max(l,m) \leq n-1\}. \tag{2.174}$$

$\mu^*(\omega)$ is defined by

$$\mu^*(\omega) = \sup_{h \in H} \mu(h;\omega), \tag{2.175}$$

where H is the set of admissible mesh sizes for example for $\Omega = (0,1)^2$ if the coarsest grid on which smoothing is applied corresponds to $h = 1/4$ then $H = \{h = 1/n : n \geq 4\}$. Given the eigenvalues of $S_h(\omega)$ are

$$\lambda_h^{l,m} = 1 - \frac{\omega}{2}(2 - \cos l\pi h - \cos m\pi h) \tag{2.176}$$

we have

$$\mu(h;\omega) = \max\{|1 - \frac{\omega}{2}(2 - \cos l\pi h - \cos m\pi h)| : n/2 \leq \max(l,m) \leq n-1\}. \tag{2.177}$$

The extremes of $\lambda_h^{l,m}$ occur when $l = m = n-1$

$$1 - \frac{\omega}{2}(2 - \cos l\pi h - \cos m\pi h) = 1 - \omega(1 + \cos \pi h) \tag{2.178}$$

and when either m or $l = n/2$ and l or $m = 1$

$$1 - \frac{\omega}{2}(2 - \cos l\pi h - \cos m\pi h) = 1 - \frac{\omega}{2}(2 - \cos \pi h) \tag{2.179}$$

so

$$\mu(h;\omega) = \max\{|1 - \omega(1 + \cos \pi h)|, |1 - \frac{\omega}{2}(2 - \cos \pi h)|\}. \tag{2.180}$$

and

$$\mu^*(\omega) = \max\{|1 - \omega/2|, |1 - 2\omega|\}. \tag{2.181}$$

For $\omega \leq 0$ or $\omega > 1$ we see that $\mu(h;\omega) \geq 1$ provided h is small enough so Jacobi relaxation has no smoothing properties in these cases. For $0 < \omega < 1$ the smoothing factor is less than 1 and bounded away from 1 independently of h. The optimal choice of ω i.e the choice of ω which minimises $\mu^*(\omega)$ occurs when

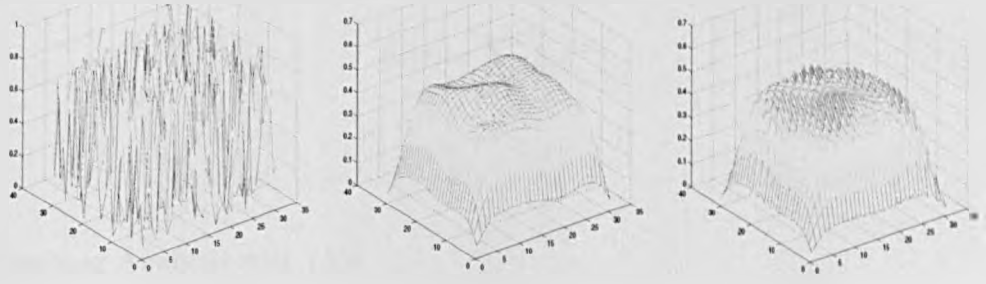
$$-(1 - 2\omega) = 1 - \omega/2 \tag{2.182}$$

$$\omega = \frac{4}{5} \tag{2.183}$$

$$\mu^*(\omega) = \frac{3}{5}. \tag{2.184}$$

One step of weighted Jacobi relaxation with optimal choice of ω will reduce all high frequency components of the error by at least a factor of 3/5 independent of the grid spacing h.

Figure 2.4: Weighted Jacobi for Poisson's equation on a 31×31 grid: original error (left) and error after 20 steps of weighted Jacobi with $\omega = 4/5$ (centre) and $\omega = 1$ (right)



Local Fourier Analysis

The smoothing analysis for the above case was relatively simple because the eigenfunctions of S_h were the fourier sine components. However in general smoothing analysis is more complicated and requires the use of Local Fourier Analysis.

In Local Fourier Analysis boundary conditions are not taken into account, instead analysis is based on an infinite grid $G^h = \{\mathbf{x} = (x, y) = (ih, jk) : (i, j) \in \mathbb{Z}^2\}$ with grid spacing

$\mathbf{h} = (h, k)$. The analysis is based on discrete linear operators with constant coefficients i.e operators L_h whose stencil entries $L_{p,q}$ are not dependant on position p in the grid.

Remark 2.6.4 *Nonlinear operators can be analysed based on a local linearization and linear operators with nonconstant coefficients can be analysed locally.*

The action of such operators on the grid functions

$$\phi_h((\theta_1, \theta_2), \mathbf{x}) = e^{i\theta_1 x/h} e^{i\theta_2 y/h} \tag{2.185}$$

for $(x, y) \in G^h$ is considered. Unlike in (2.148) θ_1, θ_2 are continuous parameters (obviously we need only consider $-\pi \leq \theta_1, \theta_2 < \pi$). With respect to standard coarsening, low frequency components are $\phi_h(\theta, \mathbf{x})$ such that $\theta = (\theta_1, \theta_2) \in [-\pi/2, \pi/2)$ and high frequency components are $\phi_h(\theta, \mathbf{x})$ such that $\theta \in [-\pi, \pi) \backslash [-\pi/2, \pi/2)$. The following theorem forms a basis for most of the results in Local Fourier Analysis

Theorem 2.6.2 *The grid functions $\phi_h(\theta, \mathbf{x})$ are eigenfunctions of any discrete linear operator L_h with constant coefficients.*

Proof

If L_h has constant coefficients then

$$\begin{aligned} L_h \phi_h(\theta, \mathbf{x}) &= \textstyle\sum_{q \in \mathbf{Z}^2} L_q e^{i\theta.(\mathbf{x}+q\mathbf{h})./\mathbf{h}} \\ &= \left(\textstyle\sum_{q \in \mathbf{Z}^2} L_q e^{i\theta.q} \right) e^{i\theta.\mathbf{x}./\mathbf{h}} \end{aligned}$$

where ./ denotes componentwise division. The eigenvalues of L_h are

$$\tilde{L_h}(\theta) = \sum_{q \in \mathbf{Z}^2} L_q e^{i\theta.q}. \tag{2.186}$$

■

Smoothing Analysis with LFA

With the above result it is straightforward to analyse the smoothing effect of any relaxation method on $L_h u_h = f_h$ one step of which can be written as

$$L_h^+ \hat{v}_h + L_h^- v_h = f_h, \tag{2.187}$$

where v_h is the approximation before the relaxation step, $\hat{v}_h$ is the approximation after the step and $L_h = L_h^+ + L_h^-$. Subtracting (2.187) from $L_h u_h = f_h$ we get

$$L_h^+ \hat{e}_h + L_h^- e_h = 0, \tag{2.188}$$

which is equivalent to

$$\hat{e}_h = S_h e_h, \tag{2.189}$$

where S_h is the smoothing operator. From (2.188) and Theorem 2.6.2 we see that $\phi(\theta, \mathbf{x})$ are eigunfunctions of S_h and

$$S_h\phi(\theta,\mathbf{x}) = \tilde{S}_h\phi(\theta,\mathbf{x}) = -\frac{\tilde{L_h^-}(\theta)}{\tilde{L_h^+}(\theta)}\phi(\theta,\mathbf{x}). \tag{2.190}$$

where we are assuming that $\tilde{L_h^+}(\theta) \neq 0$. The smoothing factor μ_{loc} is therefore

$$\mu_{loc}(S_h) = \sup\{|\tilde{S}_h(\theta)| : \theta \in [-\pi,\pi)\backslash[-\pi/2,\pi/2)\}. \tag{2.191}$$

Example

By way of an example let us consider GS-LEX applied to Poisson's equation, from (2.100) we see that

$$L_h^- = \frac{1}{h^2}\begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & -1 \\ 0 & 0 & 0 \end{bmatrix} \quad L_h^+ = \frac{1}{h^2}\begin{bmatrix} 0 & 0 & 0 \\ -1 & 4 & 0 \\ 0 & -1 & 0 \end{bmatrix} \tag{2.192}$$

and hence

$$\begin{aligned} \tilde{L_h}^-(\theta) &= \tfrac{1}{h^2}\left(-e^{i\theta_1} - e^{i\theta_2}\right) \\ \tilde{L_h}^+(\theta) &= \tfrac{1}{h^2}\left(4 - e^{-i\theta_1} - e^{-i\theta_2}\right) \end{aligned} . \tag{2.193}$$

The smoothing factor in this case is

$$\mu_{loc}(S_h) = \sup\left\{\left|\frac{e^{i\theta_1} + e^{i\theta_2}}{4 - e^{-i\theta_1} - e^{-i\theta_2}}\right| : \theta \in [-\pi,\pi)\backslash[-\pi/2,\pi/2)\right\}. \tag{2.194}$$

It is shown in [102] that the supremum is achieved for $(\theta_1, \theta_2) = (\pi/2, \cos^{-1}(4/5))$ and is equal to 0.5. A similar analysis for weighted Jacobi gives the same result as found above with the rigourous fourier analysis. ∎

Remark 2.6.5 *Although Jacobi and GS-LEX (plus their line analogues) can be written in the form (2.187), GS-RB cannot. Local Fourier analysis can still be used to analyse GS-RB type smoothers but the analysis is more involved see [92] for more details.*

2.6.5 Coarse Grid Correction scheme

Having introduced heuristically the idea of coarse grid correction and its combination with iterative methods which smooth the error, I can now define more precisely in Algorithm 2, what one step of a coarse grid correction procedure (CGC) for a linear equation $L_h u_h = f_h$ on a grid Ω^h involves. Here S_h and c_h relate to the choice of the smoother. The parameters ν_1 and ν_2 are the number of relaxation sweeps performed pre and post correction respectively, which in practice will usually be small. The operator L_{2h} is usually the direct analogue of

Algorithm 2 Coarse Grid Correction

$$v_h \leftarrow CGC\,(v_h, f_h, L_h, S_h, c_h, \nu_1, \nu_2)$$

1. For $l = 1$ to ν_1
   $v_h \leftarrow S_h v_h + c_h$
2. Compute residual $r_h = f_h - L_h v_h$.
3. Restrict residual: $r_{2h} = I_h^{2h} r_h$
4. Solve $L_{2h} e_{2h} = r_{2h}$ on Ω^{2h}.
5. Interpolate error: $e_h = I_{2h}^h e_{2h}$
6. Correct fine grid approximation: $v_h \leftarrow v_h + e_h$
7. For $l = 1$ to ν_2
   $v_h \leftarrow S_h v_h + c_h$.

L_h on the grid Ω^{2h} i.e the discrete operator which results from discretizing the continuous problem Ω^{2h}. An alternative is the Galerkin approach which defines L_{2h} as $I_h^{2h} L_h I_{2h}^h$. The Galerkin approach is often combined with more sophisticated, matrix dependent interpolation operators [23, 43, 44, 66] used for more difficult problems in which the coarse grid operator is not well approximated by rediscretization and within the purely black box algebraic multigrid methods (see §2.7 for more details) to automatically define an accurate coarse grid problem.

The Two Grid Operator

From Algorithm 2 we see that we can define coarse grid correction as an iterative procedure for updating v_h

$$v_h \leftarrow M_h^{2h} v_h + q_h, \tag{2.195}$$

where the two-grid operator M_h^{2h} is given by

$$M_h^{2h} = S_h^{\nu_2} [I_h + I_{2h}^h L_{2h}^{-1} I_h^{2h} (-L_h)] S_h^{\nu_1}. \tag{2.196}$$

and

$$q_h = S_h^{\nu_2} I_{2h}^h L_{2h}^{-1} I_h^{2h} \left[f_h - L_h \left(\sum_{j=0}^{\nu_1 - 1} S_h^j c_h \right) \right] + \sum_{j=0}^{\nu_2 - 1} S_h^j c_h. \tag{2.197}$$

The asymptotic convergence factor of this process is therefore given by the spectral radius of the two-grid operator $\rho(M_h^{2h})$.

2.6.6 Multigrid Methods

Each coarse grid correction step requires the residual equation to be solved exactly on the coarse grid Ω^{2h}. Although Ω^{2h} has 4 times fewer grid points than Ω^h if the problem is large solution of the coarse grid problem using a direct solver is still likely to be prohibitively expensive. We could use a uni-grid iterative method, but a better approach might be to use coarse grid correction again i.e solve the residual equation on Ω^{2h} by relaxing on its residual equation on the grid Ω^{4h} (a grid whose grid spacing is twice that of Ω^{2h}). This residual equation can in turn be solved by moving to a grid Ω^{8h} and so on, until we reach some very coarse grid Ω^{ph} on which the residual equation can be solved exactly using a direct method at a very low computational cost. If on each coarse grid μ coarse grid correction steps are used to approximately solve the residual equation we have what is known as a μ-cycle multigrid step. A μ-cycle multigrid step to update the approximation to a linear system $L_h u_h = f_h$ on a grid Ω^h is denoted

$$v_h \leftarrow M\mu_h \left(v_h, f_h, L_h, S_h, c_h, \nu_1, \nu_2\right), \tag{2.198}$$

where $M\mu_{\hat{h}}$ is defined recursively in Algorithm 3. In practice only $\mu = 1$ or 2 is used, these

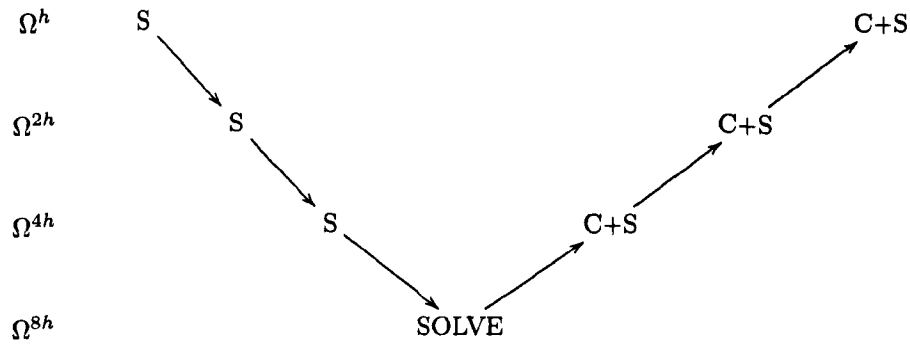
Algorithm 3 $M\mu_h$

$$v_{\hat{h}} \leftarrow M\mu_{\hat{h}} \left(v_{\hat{h}}, f_{\hat{h}}, L_{\hat{h}}, S_{\hat{h}}, c_{\hat{h}}, \nu_1, \nu_2\right)$$

1. If $\Omega^{\hat{h}}$ =coarsest grid, solve $L_{\hat{h}} u_{\hat{h}} = f_{\hat{h}}$ using a direct solver and stop.
   Else For $l = l$ to ν_1
   $v_{\hat{h}} \leftarrow S_{\hat{h}} v_{\hat{h}} + c_{\hat{h}}$
2. $f_{2\hat{h}} \leftarrow I_{\hat{h}}^{2\hat{h}} \left(f_{\hat{h}} - L_{\hat{h}} v_{\hat{h}}\right)$
   $v_{2\hat{h}} \leftarrow 0$
3. For $l = 1$ to μ
   $v_{2\hat{h}} \leftarrow M\mu_{2\hat{h}} \left(v_{2\hat{h}}, f_{2\hat{h}}, S_{2\hat{h}}, c_{2\hat{h}}, \nu_1, \nu_2,\right)$.
4. Correct $v_{\hat{h}} \leftarrow v_{\hat{h}} + I_{2\hat{h}}^{\hat{h}} v_{2\hat{h}}$.
5. For $l = l$ to ν_1
   $v_{\hat{h}} \leftarrow S_{\hat{h}} v_{\hat{h}} + c_{\hat{h}}$.

methods are known as the V-cycle and the W-cycle respectively. The schedule of grids visited for a 4-grid multigrid V-cycle is shown below, S stands for smoothing and C for correction. It is clear from the shape where the V-cycle gets its name from, similarly the W-cycle has a

W shaped grid schedule.



2.6.7 The Multigrid Operator

The multigrid method can be written in the form

$$v_h \leftarrow M_h v_h + q_h, \tag{2.199}$$

where M_h can be defined recursively. To see this first note that, assuming convergence, one step of such a procedure transforms the error as follows

$$e_h \leftarrow M_h e_h. \tag{2.200}$$

Now lets assume we are on the grid $\Omega^{\hat{h}}$ and see what happens to the initial error $e^0_{\hat{h}}$. If we are on the coarsest grid then we solve $L_{\hat{h}} u_{\hat{h}} = f_{\hat{h}}$ exactly and so the error becomes zero, otherwise we start by applying ν_1 smoothing steps resulting in a new error $e^s_{\hat{h}} = S^{\nu_1}_{\hat{h}} e^0_{\hat{h}}$. Next the residual in the approximation is transferred to the grid $\Omega^{2\hat{h}}$, the equation on this grid using $r_{\hat{h}} = L_{\hat{h}} e^s_{\hat{h}}$ is

$$L_{2\hat{h}} u_{2\hat{h}} = I^{\hat{h}}_{2\hat{h}} L_{\hat{h}} e^s_{\hat{h}}. \tag{2.201}$$

At this stage we take an initial guess $v_{2\hat{h}} = 0$ and apply $M_{2\hat{h}}$ μ times. Since the initial guess is zero, the initial error is the exact solution of (2.201) which is

$$L^{-1}_{2\hat{h}} I^{\hat{h}}_{2\hat{h}} L_{\hat{h}} e^s_{\hat{h}}. \tag{2.202}$$

The new approximation after μ applications of $M_{2\hat{h}}$ will be equal to the true solution minus the new error i.e

$$(I_{2\hat{h}} - M^{\mu}_{2\hat{h}}) L^{-1}_{2\hat{h}} I^{\hat{h}}_{2\hat{h}} L_{\hat{h}} e^s_{\hat{h}}. \tag{2.203}$$

This approximation to the solution of the coarse grid problem is transferred back to the fine grid $\Omega^{\hat{h}}$ and used to correct the approximation giving us a new error

$$e^{sc}_{\hat{h}} = \left[I_{\hat{h}} - I^{\hat{h}}_{2\hat{h}} (I_{2\hat{h}} - M^{\mu}_{2\hat{h}}) L^{-1}_{2\hat{h}} I^{\hat{h}}_{2\hat{h}} L_{\hat{h}} \right] S^{\nu_1}_{h} e^0_{\hat{h}}. \tag{2.204}$$

The final application of smoothing gives us the final error

$$e_{\hat{h}}^{scs} = S_{\hat{h}}^{\nu_2}\left[I_{\hat{h}} - I_{2\hat{h}}^{\hat{h}}(I_{2\hat{h}} - M_{2\hat{h}}^{\mu})L_{2\hat{h}}^{-1}I_{2\hat{h}}^{\hat{h}}L_{\hat{h}}\right]S_{\hat{h}}^{\nu_1}e_{\hat{h}}^{0}. \tag{2.205}$$

Overall we have the following expression for $M_{\hat{h}}$

$$M_{\hat{h}} = \begin{cases} 0 & \text{if } \Omega^{\hat{h}} \text{ is the coarsest grid} \\ S_{\hat{h}}^{\nu_2}\left[I_{\hat{h}} - I_{2\hat{h}}^{\hat{h}}(I_{2\hat{h}} - M_{2\hat{h}}^{\mu})L_{2\hat{h}}^{-1}I_{2\hat{h}}^{\hat{h}}L_{\hat{h}}\right]S_{\hat{h}}^{\nu_1} & \text{otherwise} \end{cases}. \tag{2.206}$$

2.6.8 Non-Linear Multigrid

In many cases local Nonlinear relaxation methods such as Gauss-Seidel Newton (§2.5.7) have a similar smoothing effect on the error as their linear analogues and the same principles of recursive application of smoothing and coarse grid correction that are used to construct linear multigrid methods can also be applied to discrete nonlinear problems. In the following I first introduce the nonlinear residual equation, then give a two-grid and a multigrid algorithm for a nonlinear multigrid scheme (developed by Brandt) known as the Full Approximation Scheme (FAS).

The Non-Linear Residual Equation

In the non-linear case the exact residual equation on Ω^h is

$$N_h(u_h) - N_h(v_h) = N_h(v_h + e_h) - N_h(v_h) = r_h, \tag{2.207}$$

where v_h is the approximation to u_h, e_h is the error in v_h and $r_h = f_h - N_h(v_h)$ is the residual. This equation is approximated on Ω^{2h} by

$$N_{2h}(v_{2h} + e_{2h}) - N_{2h}(v_{2h}) = r_{2h}. \tag{2.208}$$

Note that if N_h was linear then the residual equation on Ω^{2h} is

$$N_{2h}v_{2h} + N_{2h}e_{2h} - N_{2h}v_{2h} = r_{2h}, \tag{2.209}$$

which is

$$N_{2h}e_{2h} = r_{2h}. \tag{2.210}$$

So for linear operators FAS is equivalent to linear multigrid.

Two-grid Cycle

I can now define (Algorithm 4) the FAS 2-grid cycle. *SMOOTH* represents one step of some local Nonlinear relaxation method, such as Gauss-Seidel Newton to update v_h. The most important thing to note from the algorithm is that we are solving an equation

$$N_{2h}(u_{2h}) = f_{2h} \tag{2.211}$$

Algorithm 4 FAS Two-Grid

$$v_h \leftarrow FASCGC(v_h, N_h, f_h, \nu_1, \nu_2)$$

1. For $l = 1$ to ν_1
   $v_h \leftarrow SMOOTH(v_h, f_h, N_h)$
2. Compute residual $r_h = f_h - N_h v_h$
3. Restrict residual and approximation
   $v_{2h} \leftarrow \hat{I}_h^{2h} v_h$
   $r_{2h} \leftarrow I_h^{2h} r_h$
4. Solve $N_{2h}(u_{2h}) = r_{2h} + N_{2h}(v_{2h})$
5. Compute error $e_{2h} = u_{2h} - v_{2h}$
6. Interpolate error $e_h \leftarrow I_{2h}^h e_{2h}$
7. Correct fine grid approximation $v_h \leftarrow v_h + e_h$
8. For $l = 1$ to ν_2
   $v_h \leftarrow SMOOTH(v_h, f_h, N_h)$

on the coarse grid with a right hand side

$$f_{2h} = r_{2h} + N_{2h}(v_{2h}). \tag{2.212}$$

This requires the restriction of the approximation v_h obtained after the first lot of smoothing steps in addition to the restriction of the residual. The restriction operator $\hat{I}_h^{2h}$ used to restrict v_h does not necessarily have to be the same as the restriction operator I_h^{2h} used to restrict r_h. Furthermore the solution is

$$w_{2h} = v_{2h} + e_{2h} \tag{2.213}$$

and it is not this full approximation which is interpolated back to the fine grid but the error

$$e_{2h} = u_{2h} - v_{2h}. \tag{2.214}$$

Only the error is guaranteed to be smooth on the fine grid (provided proper smoothing procedures are used).

Obviously to extend the 2-grid method to a multigrid method we employ coarse grid correction repeatedly to solve the nonlinear residual equation until we get to some very coarse grid. Note that on the coarsest grid we will have to solve the residual equation using an iterative method such as Newton's Method. Note also that unlike in the linear case where we use an initial guess 0 for the solution to the residual equation on Ω^{2h} in the nonlinear case because we are working with full approximations we use initial guess v_{2h} the restricted approximation.

μ-cycle algorithm

The FAS μ-cycle operator is defined recursively in Algorithm 5.

2.6.9 Krylov Acceleration of Nonlinear Multigrid

A possible way to accelerate a nonlinear multigrid method is with a Krylov acceleration scheme introduced by Oosterlee and Washio in [100]. If we write our nonlinear system of equations on the finest grid as

$$F(u_h) = N_h(u_h) - f_h = 0 \tag{2.215}$$

then given a current approximation to the solution u_h^C resulting from the most recent multigrid step and l stored intermediate solutions $u_h^1, .., u_h^l$ (obviously if only k previous multigrid cycles have been performed where $k < l$ then we will have only k intermediate solutions) we wish to find a more optimal solution in the space

$$u_h^C + span[u_h^1 - u_h^C, u_h^2 - u_h^C, ..., u_h^l - u_h^C]. \tag{2.216}$$

Algorithm 5 FAS

$$v_h \leftarrow FAS\mu_h(v_h, N_h, f_h, \nu_1, \nu_2)$$

1. If Ω^h = coarsest grid solve $N_h u_h = f_h$ using an iterative solver and stop.
   Else For $l = 1$ to ν_1
   $v_h \leftarrow SMOOTH(v_h, N_h, f_h)$

2. $v_{2h} \leftarrow \hat{I}_h^{2h} v_h$
   $\bar{v}_{2h} \leftarrow v_{2h}$
   $f_{2h} \leftarrow I_h^{2h}(f_h - N_h v_h) + N_{2h}(v_{2h})$

3. For $l = 1$ to μ
   $v_{2h} \leftarrow FAS\mu_{2h}(v_{2h}, N_{2h}, f_{2h}, \nu_1, \nu_2)$

4. Correct $v_h \leftarrow v_h + I_{2h}^h(v_{2h} - \bar{v}_{2h})$

5. For $l = 1$ to ν_1
   $v_h \leftarrow SMOOTH(v_h, N_h, f_h)$

In order to do this we make a linear approximation of the nonlinear operator F around u^C on the space $u^C + span[u_h^1 - u_h^C, u_h^2 - u_h^C, ..., u_h^l - u_h^C]$.

$$F(u_h^C + \sum_{j=1}^{l} \theta_j(u_h^j - u_h^C)) \approx F(u_h^C) + \sum_{j=1}^{l} \theta_j \left(\frac{\partial F}{\partial u}\right)_{u_h^C} (u_h^j - u_h^C) \approx F(u_h^C) + \sum_{j=1}^{l} \theta_j(F(u_h^j) - F(u_h^C)). \tag{2.217}$$

We then define a new solution

$$u_h^A = u_h^C + \sum_{j=1}^{l} \theta_j(u_h^j - u_h^C). \tag{2.218}$$

where the parameters $\theta_1, .., \theta_l$ are chosen so as to minimize

$$\|F(u_h^C) + \sum_{j=1}^{l} \theta_j(F(u_h^j) - F(u_h^C))\|_2. \tag{2.219}$$

If we denote $F(u_h^C)$ by X, $\sum_{j=1}^{l} \theta_j(F(u_h^j) - F(u_h^C))$ by Y and $F(u_h^j) - F(u_h^C)$ by Z_j. Minimizing (2.219) is equivalent to minimizing

$$\Phi = (X + Y, X + Y) = (X, X) + 2(X, Y) + (Y, Y), \tag{2.220}$$

where

$$(X, Y) = \theta_1(X, Z_1) + ... + \theta_l(X, Z_l) \tag{2.221}$$

and

$$(Y,Y) = \sum_{j=1}^{l} \theta_j \left[\sum_{i=1}^{l} \theta_i (Z_j, Z_i) \right]. \tag{2.222}$$

$$\frac{\partial \Phi}{\partial \theta_i} = 2(X, Z_i) + 2 \sum_{j=1}^{l} \theta_j (Z_i, Z_j). \tag{2.223}$$

setting $\frac{\partial \Phi}{\partial \theta_i} = 0$ for all i we get an $l \times l$ linear system

$$A\theta = \sigma. \tag{2.224}$$

to solve in order to find the optimal values for $\theta_1, .., \theta_l$, where

$$\begin{aligned} a_{ij} = (Z_i, Z_j) = (F(u_h^i) - F(u_h^C), F(u_h^j) - F(u_h^C)) = (F(u_h^i), F(u_h^j)) - (F(u_h^C), F(u_h^i)) \\ -(F(u_h^C), F(u_h^i)) + (F(u_h^C), F(u_h^C)) \end{aligned} \tag{2.225}$$

and

$$\sigma_i = -(X, Z_i) = -(F(u_h^C), F(u_h^i) - F(u_h^C)) = (F(u_h^C), F(u_h^C)) - (F(u_h^C), F(u_h^i)). \tag{2.226}$$

If the $F(u_h^C) - F(u_h^j)$ are linearly dependant, then there is no unique solution to (2.224). In order that a direct solver can be used A is replaced by $A + \delta I$, where δ is small compared to the entries of A, to prevent it being singular. It is shown in [100] that the effect of doing this is negligible.

Selection and Restarting Criteria

Since we are using a linear approximation of our nonlinear operator we have to take into account the fact that in some cases this approximation may not be reasonable, and that as the number of intermediate solutions used increases the accuracy of the approximation may decrease. In order to protect against this the following selection and restart criteria are proposed in [100].

Selection Criteria

The following 2 criteria are used to decide whether u_h^A is a suitable solution (if not u_h^C is chosen).

1. $\|F(u_h^A)\|_2 < \gamma_A \min(\|F(u_h^C)\|_2, \|F(u_h^1)\|_2, ..., \|F(u_h^l)\|_2)$
2. $\epsilon \|u_h^A - u_h^C\|_2 < \min(\|u_h^A - u_h^1\|_2, ..., \|u^A - u_h^l\|_2)$
   or $\|F(u_h^A)\|_2 < \delta \min(\|F(u_h^C)\|_2, \|F(u_h^1)\|_2, ..., \|F(u_h^l)\|_2)$

Where γ_A is chosen to be 2, ϵ to be 0.1 and δ to be 0.9 .Condition one says that the residual norm of the new solution should not be considerably larger than that of the intermediate solutions and condition two says that u^A should not be too close to any of the intermediate

solutions unless a significant reduction of the residual norm occurs.

Restart Criteria

The acceleration process is restarted (i.e all stored solutions dropped) if either of the following criteria are found in two consecutive iterations.

1. $\|F(u_h^A)\|_2 \geq \gamma_B \min(\|F(u_h^C)\|_2, \|F(u_h^1)\|_2, ..., \|F(u_h^l)\|_2)$
2. $\epsilon\|u_h^A - u_h^C\|_2 \geq \min(\|u_h^A - u_h^1\|_2, ..., \|u_h^A - u_h^l\|_2)$
   and $\|F(u_h^A)\|_2 \geq \delta \min(\|F(u_h^C)\|_2, \|F(u_h^1)\|_2, ..., \|F(u_h^l)\|_2)$

These conditions are just the opposite of the selection conditions. γ_B is taken as 2 (note γ_A and γ_B can take different values but γ_B must always be greater than 1).

The extra costs associated with Krylov acceleration of a Nonlinear multigrid method arise from the evaluation of several residuals and inner products and the direct solution of a small linear system, the cost of which is negligible, for large problems.

2.6.10 Multilevel Nonlinear Method

When using multigrid to solve nonlinear equations, there are generally, two options: either linearize the nonlinear equation, using for example Newton's Method and use linear multigrid methods as inner solvers on each step, or solve the nonlinear equation directly, using the full approximation scheme. The advantage of the latter approach, is that it is less reliant on a good initial guess, the disadvantage is that the only option for finding N_{2h} in the FAS method is direct rediscretization, there is no equivalent option to the Galerkin method in linear multigrid. If the nonlinear operator can be well approximated by this approach, there is no problem, however if the nonlinear operator has for example highly varying nonsmooth coefficients, the FAS method may not be effective. On the other hand in the linear case, sophisticated problem dependant interpolation operators can potentially be developed and combined with Galerkin coarsening to produce effective multigrid solvers.

A recent development is the multilevel nonlinear method of Yavneh and Dardyk [103], which combines both approaches outlined above to produce a nonlinear multigrid method, which has a large domain of convergence and good convergence properties for difficult problems. A brief outline of the method is given below.

Assume that we have a discrete nonlinear equation

$$N_h(u_h) = f_h \tag{2.227}$$

on some fine grid Ω^h and that one step of a linearization method involves solving the linear equation

$$L_h(\bar{u}_h)\delta u_h = r_h \tag{2.228}$$

where $\bar{u}_h$ is the current approximation. For example in Newton's method $L_h(\bar{u}_h) = N_h'(\bar{u}_h)$. Assume also that this linear operator can be accurately approximated on some coarse grid Ω^H by $\hat{I}_h^H L_h \hat{I}_H^h$ where $\hat{I}_h^H$ and $\hat{I}_H^h$ are some nonstandard interpolation and restriction operators.

Given an approximation v_h to the solution of the nonlinear equation add to the left hand side of the nonlinear residual equation $L_h(v_h)(u_h - v_h) - L_h(v_h)(u_h - v_h)$ to get

$$L_h(v_h)(u_h - v_h) + [N_h(u_h) - N_h(v_h) - L_h(v_h)(u_h - v_h)] = r_h. \tag{2.229}$$

If $L_h(v_h) = N_h'(v_h)$, then we see from Taylor's formula that in the neighbourhood of u_h the second term in the left hand side is $O(|u_h - v_h|^2)$, while the other terms are $O(|u_h - v_h|)$. It is only this small nonlinear term which is approximated directly on the coarse grid. The coarse grid approximation to (2.229) is obtained by directly rediscretizing the second term and using Galerkin coarsening to approximate the first term.

$$\begin{aligned}&[\hat{I}_h^H L_h(v_h)\hat{I}_H^h]u_H + N_H(u_H) - L_H(v_H)u_H \\ &= r_H + [\hat{I}_h^H L_h(v_h)\hat{I}_H^h]v_H + N_H(v_H) - L_H(v_H)v_H.\end{aligned} \tag{2.230}$$

The Two-grid method is given in Algorithm 6. Note that the operators $\hat{I}_h^H$ and $\hat{I}_H^h$ are used respectively to restrict the residual and interpolate the error, while a simple restriction operator I_h^H can be used to restrict the approximation. The recursive application of this approach gives a multigrid method. Similar methods to those employed in the FAS can be used for the smoothing.

2.6.11 Multigrid Convergence

One of the attractive features of traditional multigrid methods is that their convergence properties are independent of the grid spacing of the finest grid used, together with the fact that the cost of a smoothing step with one of the basic iterative methods will usually be $O(N)$ where N is the total number of grid points, this makes them optimal. Below I give an outline of the proof given in [92] which shows that if a particular (linear) two-grid method satisfies

$$\|M_h^{2h}\| \leq \sigma \tag{2.231}$$

for sufficiently small σ independent of h then the corresponding multigrid method with $\mu \geq 2$ has similar convergence properties. The proof is based on the observation that, assuming $\Omega^{\hat{h}}$ is not the coarsest grid

$$M_{\hat{h}} = M_{\hat{h}}^{2\hat{h}} + S_{\hat{h}}^{\nu_2} I_{2\hat{h}}^{\hat{h}} M_{2\hat{h}}^{\mu} L_{2\hat{h}}^{-1} I_{2\hat{h}}^{\hat{h}} L_{\hat{h}} S_{\hat{h}}^{\nu_1}, \tag{2.232}$$

which can be seen easily from (2.206) and (2.196). Furthermore the assumption is made that

$$\|S_{\hat{h}}^{\nu_2} I_{2\hat{h}}^{\hat{h}}\| \|L_{2\hat{h}}^{-1} I_{2\hat{h}}^{\hat{h}} L_{\hat{h}} S_{\hat{h}}^{\nu_1}\| \leq \theta \tag{2.233}$$

Algorithm 6 MNM Two-Grid

$$v_h \leftarrow MNMCGC(v_h, N_h, f_h, \nu_1, \nu_2)$$

1. For $l = 1$ to ν_1
   $v_h \leftarrow SMOOTH(v_h, f_h, N_h)$
2. Construct linearized operator $L_h(v_h)$ for N_h
3. Compute residual $r_h = f_h - N_h v_h$
4. Restrict residual and approximation
   $v_H \leftarrow I_h^H v_h$
   $r_H \leftarrow \hat{I}_h^H r_h$
5. Solve $[\hat{I}_h^H L_h(v_h)\hat{I}_H^h]u_H + N_H(u_H) - L_H(v_H)u_H = r_H + [\hat{I}_h^H L_h(v_h)\hat{I}_H^h]v_H + N_H(v_H) - L_H(v_H)v_H$
6. Compute error $e_H = u_H - v_H$
7. Interpolate error $e_h \leftarrow \hat{I}_H^h e_H$
8. Correct fine grid approximation $v_h \leftarrow v_h + e_h$
9. For $l = 1$ to ν_2
   $v_h \leftarrow SMOOTH(v_h, f_h, N_h)$

for any $\hat{h}$, then, ignoring the coarsest grid on which $M_{\hat{h}} = 0$, we have that $\|M_{\hat{h}}\| \leq \eta_{\hat{h}}$ with

$$\eta_{\hat{h}} = \begin{cases} \sigma & \text{if } \Omega^{2\hat{h}} \text{ is the coarsest grid} \\ \sigma + \theta(\eta_{2\hat{h}})^{\mu} & \text{otherwise} \end{cases} . \tag{2.234}$$

Assuming that σ and θ are small enough and that $\mu = 2$ we can assume that the sequence $\{\eta_k\}$ defined by the recurrence relation $\eta_k = \sigma + \theta(\eta_{k-1})^2$ converges to η satisfying $\eta = \sigma + \theta\eta^2$ and

$$\|M_{\hat{h}}\| \leq \frac{1 - \sqrt{1 - 4\theta\sigma}}{2\theta} < \frac{1 - (1 - 4\theta\sigma)}{2\theta} = 2\sigma, \tag{2.235}$$

where we assume that $1 - 4\theta\sigma \geq 0$.

Establishing (2.231) is more difficult than establishing (2.233) and in general must be done via local fourier analysis of the various multigrid components.

An alternative approach to proving h-independent convergence was developed by Hackbush, it requires that the so called smoothing and approximation properties hold. The smoothing property states that

$$\|L_{\hat{h}} S_{\hat{h}}^{\nu}\| \leq C_S \hat{h}^{-2m} \eta(\nu), \tag{2.236}$$

where $2m$ is the order of the partial differential equation to be solved and $\eta(\nu) \to 0$ as $\nu \to \infty$. The approximation property states that

$$\|L_{\hat{h}}^{-1} - I_{2\hat{h}}^{\hat{h}} L_{2h}^{-1} I_{\hat{h}}^{2\hat{h}}\| \leq C_A \hat{h}^{2m}. \tag{2.237}$$

Again establishing these properties is not straightforward.

A general convergence theory for nonlinear problems is much more difficult. Nonlinear problems are usually analysed using LFA based on a local linearization of the nonlinear operator.

In reality if we want to find the convergence factor of a multigrid method experimentally we have to use the residuals, as that is all that is available. The quantities

$$q^{(k)} = \frac{\|r_h^k\|}{\|r_h^{k-1}\|} \text{ and } \hat{q}^{(k)} = \left(\frac{\|r_h^k\|}{\|r_h^0\|}\right)^{1/k} \tag{2.238}$$

are good estimates for the convergence factor provided k is large. Here $\|r_h^k\|$ is the residual on the finest grid after k multigrid cycles measured in some appropriate norm e.g the Euclidean norm.

2.6.12 Storage and Computational Cost

From the multigrid algorithms we see that at any one time on each level we need to store a discrete approximation and a discrete right hand side, plus the restricted approximation from the grid above in the nonlinear case. For a d-dimensional domain if we assume that the

fine grid is partitioned into n cells in each direction where n is a power of 2, then for the case of a cell centered disrcetization we have n^d grid points on the fine grid 2^{-d} as many points for Ω^{2h} and in general p^{-d} as many grid points for Ω^{ph} (for a vertex centered discretization we will have slightly more or slightly less points on each grid depending on whether boundary points are stored, but for large N these amounts should be negligible) . An upper bound on the storage requirements of a multigrid scheme is then given by the sum to infinity of the geometric series

$$3n^d \sum_{n=0}^{p} (2^{-d})^n, \tag{2.239}$$

which is

$$\frac{3n^d}{1-2^{-d}}. \tag{2.240}$$

For a 2-dimensional grid with $n = 256$ this bound is ≈ 262150.

We can use similar techniques to get an upper bound for the computational cost of a multigrid cycle. First define a work unit (WU) as the cost of performing one relaxation sweep on the finest grid (this will usually be the total number of points multiplied by some relatively small constant). Using the same assumptions as above we require p^{-d} WU to relax on Ω^{ph}. First consider the $V-cycle$ with $\nu_1 = \nu_2 = 1$. We visit each level twice, so if we ignore the cost of intergrid transfers and solving on the coarsest grid, an upper bound on the computational cost is

$$\lim_{p\to\infty} 2\sum_{n=0}^{p} (2^{-d})^n = \frac{2}{1-2^{-d}} WU. \tag{2.241}$$

Now consider the W-cycle with $\nu_1 = \nu_2 = 1$ in this case we relax twice on the finest grid (once at the beginning of the algorithm once at the end) 4 times on the next finest level, 8 times on the level below that and so on, therefore provided $d \geq 2$ an upper bound for the computational cost is

$$\lim_{p\to\infty} 2\sum_{n=0}^{p} [2(2^{-d})]^n = \frac{2}{1-2(2^{-d})} WU. \tag{2.242}$$

In 2 dimensions the bounds are 8/3WU for the V-cycle and 4WU for the W-cycle which is 3/2 times greater than the V-cycle.

2.6.13 Nested Iteration

The multigrid methods discussed so far have been based on the principle of coarse grid correction, another multigrid approach is that of nested iteration.

Nested iteration works on the principle that a good initial guess for an iterative procedure on a fine grid can be obtained by using the transferred solution from a coarser grid problem. Considering a general discrete nonlinear problem $N_h(u_h) = f_h$ on a fine grid Ω^h and assuming that we have a set of coarse grids, which I will assume for now are the standard coarsened grids $\Omega^{2h}, \Omega^{4h}, .., \Omega^{ph}$ where $p = 2^L$ (other coarsening strategies can also be used) the nested

iteration algorithm is given in Algorithm 7. Here N_{lh} is an analogue of N_h on the coarse

Algorithm 7 Nested Iteration

Choose some initial guess $\hat{u}_{ph}$ on the coarsest grid Ω^{ph}.

For k=L:-1:1

$l = 2^k$

Set $\tilde{u}_{lh}$ to be the result of approximately solving

$N_{lh}(u_{lh}) = f_{lh}$ using some iterative procedure with initial guess $\hat{u}_{lh}$.

$\hat{u}_{(l/2)h} = I_{lh}^{(l/2)h}\tilde{u}_h$.

end

Solve $N_h(u_h) = f_h$ on the finest grid using $\hat{u}_h$ as initial guess in some iterative method.

grid Ω^{lh} and f_{lh} is a restriction of the fine grid right hand side to the grid Ω^{lh}. $I_{lh}^{(l/2)h}$ is an interpolation operator for transferring grid functions between Ω^{lh} and $\Omega^{(l/2)h}$. Often more accurate interpolation operators are employed in nested iteration algorithms, than are employed in coarse grid correction schemes. The initial guess $\hat{u}_{ph}$ on the coarsest grid is usually some initial guess from the fine grid restricted down to Ω^{ph}.

If the approximate solution of $N_{lh}(u_{lh}) = f_{lh}$ is achieved by using a multigrid V or W cycle and several multigrid cycles are used to solve the fine grid problem, then the resulting algorithm is known as full multigrid or FMG, see [12, 92, 102] for more details. More generally this procedure is often referred to as cascadic multigrid and can be employed in conjunction with any problem which can be defined on coarser grids and who's solution with iterative methods benefits from a good initial guess, e.g in conjunction with discrete optimization problems.

2.7 Algebraic Multigrid

For many elliptic PDEs discretized on structured grids a geometric multigrid method like the ones described in the previous section, based on a fixed hierarchy of grids, using a simple smoother such as Gauss-Seidel is very efficient as a solver. However for more complex problems such as diffusion equations with highly varying coefficients more complex smoothers and transfer operators need to be designed in order to maintain efficiency and robustness, this can be particularly complicated in three dimensional problems. For a review of alternative smoothers, transfer operators and coarsening strategies used for elliptic linear problems with nonsmooth coefficients see [23] ,see also the work of De Zeeuw [44] and Khalil and Wesseling [66] on matrix dependant transfer operators. In addition geometric multigrid is also difficult to apply to problems defined on unstructured grids. In order to try and overcome these problems the Algebraic multigrid method was developed.

An algebraic multigrid method (or AMG) is a black box solver for solving a sparse linear

system $A\mathbf{u} = \mathbf{f}$ using only the information in the matrix A. Unlike in geometric multigrid the smoother is fixed as point Gauss-Seidel and the coarse points and prolongation operator are defined automatically based on the entries of A. Algebraic multigrid methods were originally developed by Brandt et al [9] and Ruge and Stuben [80], work in the direction of a black box multigrid solver had also been done by Dendy [43] but the defining of the coarse grids was still geometrically based.

An AMG should not be considered as an alternative to an efficient geometric multigrid method, rather it should be used in cases where geometric multigrid fails or is too difficult to apply such as diffusion problems with highly varying coefficients or problems based on unstructured grids, it can also be used in problems with no geometric interpretation at all. For problems in which A is spd Algebraic multigrid methods are robust and can be made to converge quickly. In the following I briefly review some known results and techniques from the algebraic multigrid literature. For a more comprehensive treatment of algebraic multigrid see for example [92] appendix A, or [99].

2.7.1 Neighbours and strong connections

Given a linear system $A\mathbf{u} = \mathbf{f}$ where A is of size $n \times n$ and $\mathbf{u} = (u_i)_{i=1,..,n}$, in some loose sense we can consider the index set $\{1,..,n\}$ as a set of points on a fine grid, we say that $j \neq i$ is a neighbour of i if $a_{ij} \neq 0$ where $1 \leq i, j \leq n$. The set of points which are neighbours of i is defined as

$$N_i = \{j \mid a_{ij} \neq 0\}. \tag{2.243}$$

Of these neighbouring points, the set of points strongly connected to i is defined as

$$S_i = \{j \mid -a_{ij} \geq \theta \max_{k \neq i}(-a_{ik})\}. \tag{2.244}$$

Here $\theta \in (0,1)$ and typically is taken to be 0.25. Note that j is considered to be strongly connected to i only if $a_{ij} < 0$. A point which is a neighbour of i but not strongly connected to i is said to be weakly connected to i. The set of points to which i is strongly connected is defined as

$$S_i^T = \{j \mid i \in S_j\}. \tag{2.245}$$

2.7.2 The AMG setup phase

Any Algebraic multigrid method is made up of 2 phases, the second phase is the usual application of multigrid cycles, the first phase known as the setup phase is where the multigrid components needed for the second phase are defined. Given the matrix $A^{(1)} = A$ and the index set $\{1,..,n\}$ which we will refer to as $\Omega^{(1)}$ we first split $\{1,..,n\}$ into two disjoint sets C and F (exactly how this is done will be discussed §2.7.5) where C is the set of coarse points which we will call $\Omega^{(2)}$ and F is the set of fine points. Having obtained this coarse/fine

splitting we then define an interpolation operator $I_{(2)}^{(1)}$ for transferring between $\Omega^{(2)}$ and $\Omega^{(1)}$. Various ways to define the interpolation operator will be discussed in §2.7.6 but it will always have the general form

$$(I_{(2)}^{(1)}\mathbf{e}^{(2)})_i = e_i^{(1)} = \begin{cases} e_i^{(2)} & \text{if } i \in C \\ \sum_{k \in P_i} w_{ik} e_k^{(2)} & \text{if } i \in F \end{cases}. \tag{2.246}$$

Here $P_i \subset C$ is called the interpolatory set and is often taken to be the set of coarse points strongly connected to i which is denoted C_i. However P_i is chosen it should be reasonably small in order to produce an efficient AMG method. The w_{ik} are the interpolation weights. The definition of the interpolation weights for several interpolation operators will be given in §2.7.6. Once the interpolation operator has been defined, the restriction operator for transferring between $\Omega^{(1)}$ and $\Omega^{(2)}$ is defined to be the transpose of the interpolation operator.

$$I_{(1)}^{(2)} = (I_{(2)}^{(1)})^T. \tag{2.247}$$

The Galerkin principle is then used to define the coarse grid matrix $A^{(2)}$.

$$A^{(2)} = I_{(1)}^{(2)}(A^{(1)})I_{(2)}^{(1)}. \tag{2.248}$$

In exactly the same way an even smaller matrix $A^{(3)}$ along with interpolation and restriction operators is defined from the entries of $A^{(2)}$. This process is repeated until we have a sequence of matrices $A^{(1)}, ..., A^{(L)}$ with corresponding transfer operators, where $A^{(L)}$ is small enough to be solved efficiently using a direct solver. Once these matrices and transfer operators have been defined and stored it is clearly straightforward to apply a multigrid cycle with point Gauss-Seidel smoother to the original linear problem. Given an initial guess $\mathbf{v}^{(1)}$ and the setup data, one μ-cycle is

$$\mathbf{v}^{(1)} \leftarrow AMG\mu^{(1)}(\mathbf{v}^{(1)}, \mathbf{f}^{(1)}, \nu_1, \nu_2). \tag{2.249}$$

where $AMG\mu^{(k)}$ is defined recursively in Algorithm 8:

2.7.3 The Variational Principle

In the following we see that the convergence of Galerkin-based V-cycles is guaranteed when A is symmetric positive definite and restriction is taken as the transpose of interpolation (note that this implies $A^{(2)}, ..., A^{(L)}$ are also spd), provided that the smoother being used converges. First a theorem on orthogonal projectors is needed.

Theorem 2.7.1 *Let $(.,.)$ be any inner product with the norm $\|.\|$. Let Q be symmetric with respect to this inner product and let $Q^2 = Q$, then Q is an orthogonal projector i.e the following statements hold*

(1) $R(Q) \perp R(I - Q)$

Algorithm 8 $AMG\mu^{(k)}$

$$\mathbf{v}^{(k)} \leftarrow AMG\mu^{(k)}(\mathbf{v}^{(k)}, \mathbf{f}^{(k)}, \nu_1, \nu_2)$$

(1) If $A^{(k)}$ is the smallest matrix (on the coarsest level), set $\mathbf{v}^{(k)} = (A^{(k)})^{-1}\mathbf{f}^{(k)}$ and return else do ν_1 steps of smoothing on level k:
For $l = 1 : \nu_1$, $\mathbf{v}^{(k)} \leftarrow S^{(k)}\mathbf{v}^{(k)} + (I - S^{(k)})(A^{(k)})^{-1}\mathbf{f}^{(k)}$.

(2) Restrict the residual to the coarser level: $\mathbf{f}^{(k+1)} = I_{(k)}^{(k+1)}(\mathbf{f}^{(k)} - A^{(k)}\mathbf{v}^{(k)})$
and set the correction $\mathbf{v}^{(k+1)} = 0$.

(3) Repeat μ steps of, $\mathbf{v}^{(k+1)} \leftarrow AMG\mu^{(k+1)}(\mathbf{v}^{(k+1)}, \mathbf{f}^{(k+1)}, \nu_1, \nu_2)$

(4) Add the coarse level correction: $\mathbf{v}^{(k)} \leftarrow \mathbf{v}^{(k)} + I_{(k+1)}^{(k)}\mathbf{v}^{(k+1)}$

(5) Do ν_2 steps of smoothing on level k:
For $l = 1 : \nu_2$, $\mathbf{v}^{(k)} \leftarrow S^{(k)}\mathbf{v}^{(k)} + (I - S^{(k)})(A^{(k)})^{-1}\mathbf{f}^{(k)}$.

Here $S^{(k)}$ is the Gauss-Seidel smoothing operator for $A^{(k)}$ i.e

$$S^{(k)} = I - (Q^{(k)})^{-1}A^{(k)}. \tag{2.250}$$

where $Q^{(k)}$ is the lower triangular part of $A^{(k)}$ including the diagonal.

(2) For $u \in R(Q)$ and $v \in R(I - Q)$ we have $\|u + v\| = \|u\| + \|v\|$.

(3) $\|Q\| = 1$

(4) For all u: $\|Qu\| = \min_{v \in R(I-Q)} \|u - v\|$

where $R(Q)$ denotes the range of Q.

Now we consider the action of coarse grid correction on two grids on the fine grid error $\mathbf{e}^{(1)}$. Clearly if $\bar{\mathbf{e}}^{(1)}$ represents the error after coarse grid correction

$$\bar{\mathbf{e}}^{(1)} = \mathbf{e}^{(1)} - I_{(2)}^{(1)}\mathbf{e}^{(2)} = K_{1,2}\mathbf{e}^{(1)}, \tag{2.251}$$

where

$$K_{1,2} = I - I_{(2)}^{(1)}(A^{(2)})^{-1}I_{(1)}^{(2)}A^{(1)}. \tag{2.252}$$

Now note that given that A is symmetric $AK_{1,2} = A^{(1)} - A^{(1)}I_{(2)}^{(1)}(A^{(2)})^{-1}I_{(1)}^{(2)}A^{(1)}$ is symmetric which means $K_{1,2}$ is symmetric with respect to the energy inner product $(.,.)_A$ where

$$(\mathbf{u}, \mathbf{v})_A = (A\mathbf{u}, \mathbf{v})_2 \tag{2.253}$$

and $(.,.)_2$ denotes the Euclidean inner product. Now consider $K_{1,2}^2$

$$K_{1,2}^2 = I - I_{(2)}^{(1)}(A^{(2)})^{-1}I_{(1)}^{(2)}A^{(1)}[I - I_{(2)}^{(1)}(A^{(2)})^{-1}I_{(1)}^{(2)}A^{(1)}]$$

$$
\begin{aligned}
&= K_{1,2} - I_{(2)}^{(1)}(A^{(2)})^{-1}I_{(1)}^{(2)}A^{(1)} + [I_{(2)}^{(1)}(A^{(2)})^{-1}I_{(1)}^{(2)}A^{(1)}][I_{(2)}^{(1)}(A^{(2)})^{-1}I_{(1)}^{(2)}A^{(1)}] \\
&= K_{1,2} - I_{(2)}^{(1)}(A^{(2)})^{-1}I_{(1)}^{(2)}A^{(1)} + [I_{(2)}^{(1)}(A^{(2)})^{-1}A^{(2)}(A^{(2)})^{-1}I_{(1)}^{(2)}A^{(1)}] \\
&= K_{1,2}. \qquad (2.254)
\end{aligned}
$$

Finally observe that

$$R(I - K_{1,2}) = R(I_{(2)}^{(1)}). \qquad (2.255)$$

We therefore have that the 2-grid operator $K_{1,2}$ is an orthogonal projector with respect to the energy norm and from statement (4) in Theorem 2.7.1 we have for any fine grid error $\mathbf{e}^{(1)}$

$$\|K_{1,2}\mathbf{e}^{(1)}\|_A = \min_{\mathbf{e}^{(2)}} \|\mathbf{e}^{(1)} - I_{(2)}^{(1)}\mathbf{e}^{(2)}\|_A. \qquad (2.256)$$

In other words the Galerkin-based coarse grid corrections minimize the energy norm of the error with respect to all variations in the range of the interpolation operator. As a consequence of this fact if the smoother converges a two grid method will converge. To extend this result to complete V-cycles assume that the exact coarse level correction $\mathbf{e}^{(2)}$ is replaced by an approximation $\tilde{\mathbf{e}}^{(2)}$ (in this case obtained by correction on coarser levels) where $\|\mathbf{e}^{(2)} - \tilde{\mathbf{e}}^{(2)}\|_{A^{(2)}} \leq \|\mathbf{e}^{(2)}\|_{A^{(2)}}$ and consider the action of the resulting approximate two-grid operator $\tilde{K}_{1,2}$. We have

$$\tilde{K}_{1,2}\mathbf{e}^{(1)} = \mathbf{e}^{(1)} - I_{(2)}^{(1)}\tilde{\mathbf{e}}^{(2)} = K_{1,2}\mathbf{e}^{(1)} + I_{(2)}^{(1)}\mathbf{e}^{(2)} - I_{(2)}^{(1)}\tilde{\mathbf{e}}^{(2)} = K_{1,2}\mathbf{e}^{(1)} + I_{(2)}^{(1)}(\mathbf{e}^{(2)} - \tilde{\mathbf{e}}^{(2)}). \qquad (2.257)$$

The first term here belongs to $R(K_{1,2})$ and the second term belongs to $R(I_{(2)}^{(1)})$ and so by statement (2) in Theorem 2.7.1 and the fact that $K_{1,2}$ is an orthogonal projector we have

$$\|\tilde{K}_{1,2}\mathbf{e}^{(1)}\|_A^2 = \|K_{1,2}\mathbf{e}^{(1)}\|_A^2 + \|I_{(2)}^{(1)}(\mathbf{e}^{(2)} - \tilde{\mathbf{e}}^{(2)})\|_A^2. \qquad (2.258)$$

Now for any $\mathbf{v}^{(2)}$

$$
\begin{aligned}
\|I_{(2)}^{(1)}\mathbf{v}^{(2)}\|_{A^{(1)}}^2 &= \left(A^{(1)}I_{(2)}^{(1)}\mathbf{v}^{(2)}, I_{(2)}^{(1)}\mathbf{v}^{(2)}\right)_2 = (\mathbf{v}^{(2)})^T I_{(1)}^{(2)}A^{(1)}I_{(2)}^{(1)}\mathbf{v}^{(2)} \\
&= (\mathbf{v}^{(2)})^T A^{(2)}\mathbf{v}^{(2)} = \|\mathbf{v}^{(2)}\|_{A^{(2)}}^2. \qquad (2.259)
\end{aligned}
$$

So

$$\|I_{(2)}^{(1)}(\mathbf{e}^{(2)} - \tilde{\mathbf{e}}^{(2)})\|_{A^{(1)}}^2 = \|\mathbf{e}^{(2)} - \tilde{\mathbf{e}}^{(2)}\|_{A^{(2)}}^2 \leq \|\mathbf{e}^{(2)}\|_{A^{(2)}}^2 = \|I_{(2)}^{(1)}\mathbf{e}^{(2)}\|_{A^{(1)}}^2. \qquad (2.260)$$

Using this result in (2.258) we have

$$\|\tilde{K}_{1,2}\mathbf{e}^{(1)}\|_{A^{(1)}}^2 \leq \|K_{1,2}\mathbf{e}^{(1)}\|_{A^{(1)}}^2 + \|I_{(2)}^{(1)}\mathbf{e}^{(2)}\|_{A^{(1)}}^2 = \|K_{1,2}\mathbf{e}^{(1)} + I_{(2)}^{(1)}\mathbf{e}^{(2)}\|_{A^{(1)}}^2 = \|\mathbf{e}^{(1)}\|_{A^{(1)}}^2. \qquad (2.261)$$

We therefore see that the approximate two-grid method and hence the V-cycle method converges. Of course this result says nothing about the speed of convergence of a Galerkin-based V-cycle method, which is determined by the choice of interpolation operator.

2.7.4 Algebraically smooth error

In the following section I discuss how a smooth error is defined in a purely algebraic setting for this discussion the energy inner product $(.,.)_A$ as defined previously and two other inner products defined below will be needed

$$(\mathbf{u},\mathbf{v})_D = (D\mathbf{u},\mathbf{v})_2 \ , \ (\mathbf{u},\mathbf{v})_{DA} = (D^{-1}A\mathbf{u},A\mathbf{v})_2, \tag{2.262}$$

where D is a diagonal matrix with the same diagonal as A.

Definition 2.7.1 (Algebraically Smooth Error)
An algebraically smooth error is an error which is slow to converge with respect to some smoothing operator S i.e

$$\|S\mathbf{e}\|_A \approx \|\mathbf{e}\|_A$$

For basic relaxation methods such as Gauss-Seidel the following inequality holds with $\sigma > 0$

$$\|S\mathbf{e}\|_A^2 \leq \|\mathbf{e}\|_A^2 - \sigma\|\mathbf{e}\|_{DA}^2. \tag{2.263}$$

This implies that an error is algebraically smooth when

$$\|\mathbf{e}\|_{DA} \ll \|\mathbf{e}\|_A. \tag{2.264}$$

This is equivalent to

$$(D^{-1}A\mathbf{e},A\mathbf{e})_2 \ll (A\mathbf{e},\mathbf{e})_2 \tag{2.265}$$

recalling that $A\mathbf{e} = \mathbf{r}$ we have

$$(D^{-1}\mathbf{r},\mathbf{r})_2 \ll (\mathbf{r},\mathbf{e})_2 \tag{2.266}$$

or

$$\sum_i r_i\,(r_i/a_{ii}) \ll \sum_i r_i e_i. \tag{2.267}$$

Therefore on the average for each i the scaled residual is much smaller than the error

$$\frac{|r_i|}{a_{ii}} \ll |e_i|. \tag{2.268}$$

Given that $r_i = (A\mathbf{e})_i$ this implies that

$$e_i \approx -\left(\sum_{j\in N_i} a_{ij}e_j\right)/a_{ii}.. \tag{2.269}$$

A very accurate interpolation operator can therefore be defined if on each level P_i is chosen to be N_i and the interpolation weights are chosen to be $w_{ik} = -a_{ik}/a_{ii}$. However this would require a coarse/fine-splitting in which each fine point had all its neighbours contained in the coarse set. Such an approach would be very expensive in terms of memory and computational time and so is not a practical option.

If we rewrite $(Ae, e)_2$ as $(D^{-1/2}Ae, D^{1/2}e)_2$ and apply the Cauchy-Schwartz inequality we get

$$\|e\|_A^2 \leq \|D^{-1/2}Ae\|_2 \|D^{1/2}e\|_2 = (D^{-1/2}Ae, D^{-1/2})_2^{1/2} (D^{1/2}e, D^{1/2})_2^{1/2} = \|e\|_{DA}\|e\|_D. \tag{2.270}$$

Since an algebraically smooth error satisfies $\|e\|_A \gg \|e\|_{DA}$ it must also satisfy

$$\|e\|_A \ll \|e\|_D. \tag{2.271}$$

In the following matrices with positive diagonal entries and negative off-diagonal entries (M-matrices) are considered. We can rewrite $(Ae, e)_2$ as follows

$$(Ae, e)_2 = \sum_{i,j} a_{ij}e_ie_j = \sum_{i,j} -a_{i,j}(-e_ie_j) = \sum_{i,j} -a_{i,j}(1/2(e_i - e_j)^2 - 1/2e_i^2 - 1/2e_j^2). \tag{2.272}$$

Substituting into (2.271) we have

$$1/2 \sum_{i,j} -a_{ij}(e_i - e_j)^2 + \sum_i \left(\sum_j a_{ij} \right) e_i^2 \ll \sum_i a_{ii}e_i^2. \tag{2.273}$$

In the case where $\sum_{j \neq i} |a_{ij}| \approx a_{ii}$ i.e the row sum is approximately zero we have on the average for each i

$$1/2 \sum_{j \neq i} -a_{i,j}(e_i - e_j)^2 \ll a_{ii}e_i^2$$

$$\sum_{j \neq i} \frac{|a_{i,j}|}{a_{ii}} \frac{(e_i - e_j)^2}{e_i^2} \ll 2. \tag{2.274}$$

This implies that if $|a_{i,j}|/a_{ii}$ is relatively large then the error varies slowly from e_i to e_j in other words the error varies slowly from e_i to e_j if j is strongly connected to i. The error at a point is therefore well approximated by a weighted average of the error at the points strongly connected to it.

If the matrix contains some small positive off-diagonal entries then the error can again be shown to vary slowly in the direction of large (negative) entries. If large positive and negative off-diagonal entries exist then the error varies slowly in the direction of large negative entries but oscillates in the direction of large positive entries.

2.7.5 The coarse and fine level splitting

As stated earlier at each level k, we must split $\Omega^{(k)} = \{1, ..., n_k\}$ into two disjoint sets C and F where C is the set of coarse points which make up the next level and F is the set of fine points. When making this C/F-splitting we should look to achieve the following two conditions.

(1) For each point $i \in F$, every point in S_i is either in C_i or strongly connected to a point in C_i.

(2) C should be a maximal subset of all points with the property that no two C-points are strongly connected to each other.

In practice it is not usually possible to strictly satisfy both conditions. The algorithm for creating a C/F-splitting given below (Algorithm 9) attempts to enforce the second condition. At each step of the algorithm a coarse point i is chosen which has maximal λ_i where

$$\lambda_i = |S_i^T \cap U| + 2|S_i^T \cap F|, \tag{2.275}$$

where U here denotes the set of points which have yet to be defined as either C or F points. Once a C point has been chosen all the points strongly connected to that point which have yet to be defined (i.e all points in $S_i^T \cap U$) are defined to be F points. The condition that the new coarse point must have maximal λ_i ensures that a reasonably uniform distribution of C and F points is obtained.

Remark 2.7.1 *If a point $j \in S_i^T$ moves from U to C then λ_i decreases by 1, whereas if it moves from U to F λ_i increases by 1.*

Algorithm 9 C/F-Splitting Algorithm

Set $U = \Omega^{(k)}$, $C = \varnothing$, $F = \varnothing$, $\lambda_i = |S_i^T|$ for all i.
While $U \neq \varnothing$
    Select $i \in U$ with maximal λ_i.
    $C = C \cup \{i\}$, $U = U - \{i\}$
    For $j \in S_i^T \cap U$
        $F = F \cup \{j\}$
        $U = U - \{j\}$
        For $l \in S_j \cap U$
            $\lambda_l = \lambda_l + 1$
        end
    end
    For $j \in S_i \cap U$
        $\lambda_j = \lambda_j - 1$
    end
end

Algorithm 9 is based on a definition of strong connections as large negative connections. In some cases we may have a small number of large positive connections (small positive connections are not significant) which must be taken into account in the C/F-splitting. Provided

there are not too many positive connections, coarse points corresponding to positive connections can be added a posteriori based on the positive connections between F points only. Algorithm 10 tests each F point i to check whether it has large positive connections to other F points, if it has these points are added to S_i and the point corresponding to the largest connection is added to C.

Algorithm 10 Post C/F-Splitting Algorithm

Assume C and F have been obtained from Algorithm 9. Set $C_1 = \varnothing$, $F_0 = F$, $FT = F$

While $FT \neq \varnothing$

Set i =smallest entry in FT.

Set $M = 0$ $M_1 = \varnothing$

For $j \in N_i^+ \cap F$

If $a_{ij} \geq 0.5 \max_{k \neq i} |a_{ik}|$

$S_i = S_i \cup \{j\}$

If $a_{ij} > M$

$M = a_{ij}$

$M_1 = j$

end

end

end

If $M_1 \neq \varnothing$

$C_1 = C_1 \cup M_1$

$F = F \backslash M_1$ $FT = FT \backslash M_1$

end

$FT = FT \backslash \{i\}$

end

Set $C = C \cup C_1$, $F = F_0 \backslash C_1$.

2.7.6 Interpolation Operators

In the following I review several different methods for defining the interpolation operator to be used in an AMG method. To aid the description I introduce the following notation: let $D_i^W = N_i \backslash \{S_i\}$ denote the set of all points weakly connected to i and $D_i^S = S_i \backslash \{C_i\}$ denote the set of all fine points strongly connected to i. Recall also that any interpolation operator should be of the form

$$(I_{(l+1)}^{(l)} e^{(l+1)})_i = e_i^{(l)} = \begin{cases} e_i^{(l+1)} & \text{if } i \in C \\ \sum_{k \in P_i} w_{ik} e_k^{(l+1)} & \text{if } i \in F \end{cases}, \tag{2.276}$$

where $P_i \subset C$.

Ruge and Stuben's Interpolation Operator

In the method of Ruge and Stuben [80] P_i is taken to be C_i and an assumption is made that the matrix is an M-matrix. In order to define the interpolation weights w_{ik} recall that an algebraically smooth error satisfies

$$a_{ii}e_i \approx -\sum_{j\in N_i} a_{ij}e_j = -\sum_{j\in C_i} a_{ij}e_j - \sum_{j\in D_i^S} a_{ij}e_j - \sum_{j\in D_i^W} a_{ij}e_j. \tag{2.277}$$

Now at each $i \in F$ carry out the following steps:

1. For all $j \in D_i^W$ replace e_j by e_i.
2. For all $j \in D_i^S$ replace e_j by a weighted average of the error at the points in C_i

$$e_j = \frac{\sum_{k\in C_i} a_{jk}e_k}{\sum_{k\in C_i} a_{jk}}. \tag{2.278}$$

Substituting into (2.277) and rearranging gives

$$e_i \approx \sum_{k\in C_i} w_{ik}e_k, \tag{2.279}$$

where

$$w_{ik} = -\frac{1}{a_{ii} + \sum_{j\in D_i^W} a_{ij}} \left(a_{ik} + \sum_{j\in D_i^S} \frac{a_{ij}a_{jk}}{\sum_{m\in C_i} a_{jm}} \right). \tag{2.280}$$

Given that points in D_i^W are only weakly connected to i step 1 is reasonable. Step 2 uses the fact that the error varies slowly in the direction of strong connections. We see that the weighting is biased towards e_k for $k \in S_j$, this requires that each point in D_i^S be strongly connected to at least one point in C_i, this is the first condition that the C/F splitting tries to enforce, however it is not guaranteed to be satisfied for every $j \in D_i^S$. To ensure that this condition is satisfied the following process is carried out as the interpolation weights w_{ik} for $i \in F$ are being defined:

1. If a $\hat{k} \in D_i^S$ is found such that $S_{\hat{k}} \cap C_i = \varnothing$ $\hat{k}$ is provisionally added to the coarse set and the process of defining the interpolation weights for i restarted.
2. If we now have $S_k \cap C_i \neq \varnothing$ for all $k \in D_i^S$ the interpolation weights are defined and $\hat{k}$ is permanently moved into the coarse set, however if this is still not the case i itself is moved into the coarse set and $\hat{k}$ is moved back into the fine set.

Chang, Wong and Fu's Interpolation Operator

In [35] Chang, Wong and Fu present an interpolation operator, which is an improvement on the method of Ruge and Stuben in the sense that it can deal with matrices with positive

and negative off-diagonal entries and that it takes into account more geometric information. Their AMG method uses a different definition of Strong connections as entries of A with relatively large absolute value

$$S_i = \{j : |a_{ij}| \geq \theta \max_{k \neq i} |a_{ik}|\}. \tag{2.281}$$

along with the usual C/F splitting algorithm (Algorithm 9). The interpolation weights are defined based on the assumption that the error between i and j is geometrically smooth provided a_{ij} is not a large positive entry and on the assumption that the larger $|a_{ij}|$ is the closer j is to i. Based on these assumptions two variables are introduced for each $i \in F$

$$\zeta_{ij} = \frac{-\sum_{k \in C_i} a_{jk}}{\sum_{k \in C_i} |a_{jk}|} \tag{2.282}$$

and

$$\eta_{ij} = \frac{|a_{ij}||C_i \cap N_j|}{\sum_{k \in C_i} |a_{jk}|}. \tag{2.283}$$

The weights $g_{jk} = \frac{|a_{jk}|}{\sum_{k \in C_i} |a_{jk}|}$ for each $j \in N_i \backslash C$ and $k \in C_i$ are also introduced. The variable ζ_{ij} gives an indication of how many large negative entries there are among the a_{jk}. If $\zeta_{ij} \geq .5$ and $a_{ij} < 0$ it is assumed that the error between i and j is geometrically smooth. The variable η_{ij} is approximately the inverse ratio of the distance between i and j to the average distance between j and the points $k \in C_i \cap N_j$. If $\eta_{ij} < 3/4$ it is assumed that the average location of points in $C_i \cap N_j$ is closer to j than i and hence lies somewhere between i and j, the error at j is therefore approximated by an extrapolation between e_i and a weighted average of the e_k for $k \in C_i \cap N_j$. If $\eta_{ij} > 2$ j is assumed to be between i and the average location of points in $C_i \cap N_j$ and an interpolation between e_i and a weighted average of the e_k is used to approximate e_j. In other cases the average location of the k points is assumed to be very close to j and e_j is approximated using a weighted average of the e_k. Overall the following approximations are made

1. If $j \in D_i^W$

$$e_j = \begin{cases} e_i & \text{if } C_i \cap S_j = \varnothing \text{ and } a_{ij} < 0 \\ -e_i & \text{if } C_i \cap S_j = \varnothing \text{ and } a_{ij} > 0 \\ 2\sum_{k \in C_i} g_{jk} e_k - e_i & \text{if } C_i \cap S_j \neq \varnothing \text{ and } a_{ij} < 0 \text{ and } \zeta_{ij} \geq 0.5 \\ \sum_{k \in C_i} g_{jk} e_k & \text{otherwise} \end{cases}. \tag{2.284}$$

2. If $j \in D_i^S$

$$e_j = \begin{cases} 1/2\left(\sum_{k \in C_i} g_{jk} e_k + e_i\right) & \text{if } \eta_{ij} > 2 \text{ and } a_{ij} < 0 \text{ and } \zeta_{ij} > 0.5 \\ 2\sum_{k \in C_i} g_{jk} e_k - e_i & \text{if } \eta_{ij} < 3/4 \text{ and } a_{ij} < 0 \text{ and } \zeta_{ij} \geq 0.5 \\ \sum_{k \in C_i} g_{jk} e_k & \text{otherwise} \end{cases}. \tag{2.285}$$

Substituting these approximations into (2.277) gives the interpolation weights for each $i \in F$.

Direct interpolation

Direct Interpolation, along with standard interpolation are introduced by Stuben in [92] Appendix A.

In the case of M-matrices i.e spd matrices with negative off-diagonal entries we have that the error varies slowly from i to j if j is strongly connected to i (a_{ij} is a relatively large negative entry of A) and the error at i can therefore be determined by a weighted average of the error at its strongly connected neighbours. This means for $i \in F$ we can assume

$$\frac{\sum_{k\in P_i} a_{ik}e_k}{\sum_{k\in P_i} a_{ik}} \approx \frac{\sum_{j\in N_i} a_{ij}e_j}{\sum_{j\in N_i} a_{ij}} \tag{2.286}$$

and the more strong connections of i there are contained in P_i the better (2.286) is satisfied. An interpolation operator can therefore be constructed by making the approximation

$$\sum_{j\in N_i} a_{ij}e_j = \alpha_i \sum_{k\in P_i} a_{ik}e_k. \tag{2.287}$$

where

$$\alpha_i = \frac{\sum_{j\in N_i} a_{ij}}{\sum_{k\in P_i} a_{ik}}. \tag{2.288}$$

Substituting this into

$$a_{ii}e_i + \sum_{j\in N_i} a_{ij}e_j \approx 0 \tag{2.289}$$

we get

$$e_i = \sum_{k\in P_i} w_{ik}e_k \tag{2.290}$$

with

$$w_{ik} = -\alpha_i a_{ik}/a_{ii}. \tag{2.291}$$

In the case where there exist some relatively small positive off diagonal entries of A it is enough just to move these on to the diagonal. Using superscripts + and − to denote positive and negative entries of A we have

$$\alpha_i = \frac{\sum_{j\in N_i} a_{ij}^-}{\sum_{k\in P_i} a_{ik}^-} \tag{2.292}$$

and

$$w_{ik} = \frac{-\alpha_i a_{ik}^-}{a_{ii} + \sum_{j\in N_i} a_{ij}^+}. \tag{2.293}$$

If there exist a few large positive entries of A and these are represented in the C/F-splitting then we make separate approximations for the positive and negative connections.

$$\sum_{j\in N_i} a_{ij}^- e_j = \alpha_i \sum_{k\in P_i} a_{ik}^- e_k \qquad \sum_{j\in N_i} a_{ij}^+ e_j = \beta_i \sum_{k\in P_i} a_{ik}^+ e_k, \tag{2.294}$$

where

$$\alpha_i = \frac{\sum_{j \in N_i} a_{ij}^-}{\sum_{k \in P_i} a_{ik}^-} \qquad \beta_i = \frac{\sum_{j \in N_i} a_{ij}^+}{\sum_{k \in P_i} a_{ik}^+}. \tag{2.295}$$

The approximation for positive connections can be justified because although the error between i and j is likely to be oscillatory if a_{ij} is positive and relatively large, the error can be expected to vary slowly among points k for which $a_{ik} > 0$. Denoting by P_i^+ and P_i^- the points in P_i which correspond to positive and negative connections respectively, we have

$$e_i = \sum_{k \in P_i} w_{ik} e_k, \tag{2.296}$$

where

$$w_{ik} = \begin{cases} -\alpha_i a_{ik}/a_{ii} & k \in P_i^- \\ -\beta_i a_{ik}/a_{ii} & k \in P_i^+ \end{cases}. \tag{2.297}$$

Given that it is assumed that there are only a small number of large positive connections, at some i we may have $P_i^+ = 0$, if this is the case we simply revert to moving any positive connections on to the diagonal.

Standard Interpolation

Standard interpolation looks to improve upon direct interpolation by indirectly including in the interpolation for $i \in F$ strong F-connections via the points C_j for $j \in D_i^S$. For all $j \in D_i^S$, e_j is eliminated in (2.289) using the jth equation

$$e_j = -\sum_{k \in N_j} a_{jk} e_k / a_{jj}. \tag{2.298}$$

This results in a new equation

$$\hat{a}_{ii} e_i + \sum_{j \in \hat{N}_i} \hat{a}_{ij} e_j \approx 0 \qquad \hat{N}_i = \{j \neq i : \hat{a}_{ij} \neq 0\}, \tag{2.299}$$

where for each $j \notin D_i^S$

$$\hat{a}_{ij} = a_{ij} - \sum_{k \in D_i^S} a_{kj} a_{ik} / a_{kk}. \tag{2.300}$$

Direct interpolation is now applied as above with the a_{ij} replaced by $\hat{a}_{ij}$, N_i replaced by $\hat{N}_i$ and P_i taken to be the union of C_i and all C_j for $j \in D_i^S$.

Chapter 3

Total-Variation Based Image Restoration

Useful Section References: §2.1, §2.2, §2.3, §2.4, §A.1, §A.2, §A.3, §A.4, §A.5, §A.6.3
Main Reference Material: [1, 2] [5]-[7],[16],[17]-[21],[25]-[29], [39, 46, 48, 52, 58, 61, 68, 71],[74]-[86],[89]-[91],[93]-[97], [98, 104].

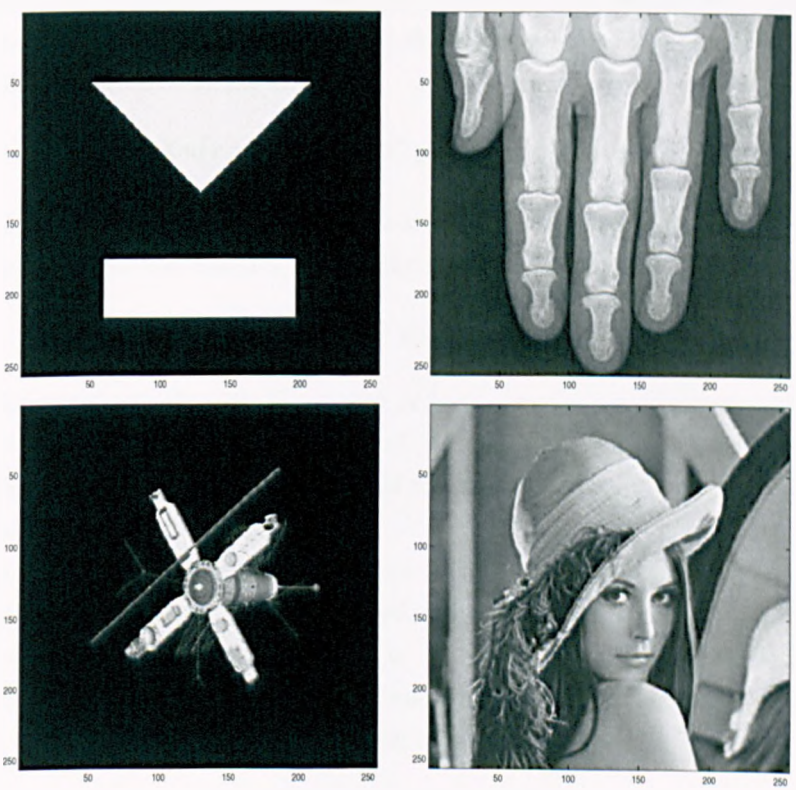
This chapter is an introduction to the ideas surrounding the recovery of possibly blurred, noisy images using regularization techniques. I start with an introduction to the image deblurring/denoising problem and then go on to introduce the regularization techniques used to approximately solve such problems focusing on the use of the total-variation regularization method, on which my work presented in the next 4 chapters has mainly been based. I give a brief review in §3.3 of some of the theory associated with this method with references to the original work where more detail can be found. The rest of the chapter focuses on the pure denoising problem (deblurring is returned to in Chapter 7). In §3.4 I review some of the properties of total variation regularization, demonstrating its advantages (edge recovery) and disadvantages (staircasing) again references are given to the original papers where more detail can be found. In §3.5 I introduce the discretization scheme that I use throughout my work as well as giving a review of other discretization methods used in the literature. §3.6 is a review of various iterative methods used to solve the total-variation denoising problem, §3.7 introduces some measures of image quality, while the final section gives a flavour of some of the improvements/extensions to the basic TV denoising method which have been developed in recent years.

3.1 What is an Image ?

An image can be interpreted as a real (scalar) valued function $u(x, y)$ on a bounded and open domain Ω (usually a rectangle) of $\mathbb{R}^2$. The type of images I am referring to here are grey-scale images, which take values in the range $[0, 255]$, if an image is a colour image it will have several components (red, green and blue) and will be represented by a vector valued function, for more details see [5] and references therein. In practice the images we deal with are digital and will therefore be discrete quantities represented by an array of pixel values. Each pixel value in the array represents the average light intensity over a small rectangular portion of the analogue image (a pixel), the more pixels used in sampling the image (the higher the resolution) the more detail can be seen. We typically deal with digital images with 256×256 to 1024×1024 pixels.

An image is generally piecewise smooth and is made up of flat regions, smoothly varying regions and edges (boundaries where a jump in intensity occurs). In Figure 3.1 examples of 4 images which will be used in the next few chapters are given.

Figure 3.1: Four example images: clockwise from top left, the blocky triangle image, the X-ray type fingers image, the realistic Lenna image and a simulated image of a satellite.



A one-dimensional version of an image is known as a signal.

3.2 Image Reconstruction

During the recording of an image, it is often the case that some blurring is introduced, this can be due, for example, to light from an object in space having to pass through the earths atmosphere, or to instrumental restrictions in medical imaging. In addition to this blurring, random noise can also be introduced during the recording and transmission of the image. We model a blurred noisy image by the following equation:

$$z(x,y) = \mathcal{K}u(x,y) + n(x,y), \qquad (x,y) \in \Omega. \tag{3.1}$$

Here z is the noisy blurred observed image which is known, u is the true image which we wish to recover and n is an additive random noise term, which in our work we assume to be gaussian white noise with mean 0 and standard deviation σ. I remark here that we refer in our work only to additive noise i.e at (x_i, y_j) the observed image z takes the value $\mathcal{K}u(x_i,y_j) + n(x_i,y_j)$ where $n(x_i,y_j)$ is some random number. Another type of noise is impulse noise in which $z(x_i,y_j)$ takes the value $\mathcal{K}u(x_i,y_j)$ with some probability p and the value $r(x_i,y_j)$ with probability $1-p$ where $r(x_i,y_j)$ is some random number, for more detail see, for example, [21]. $\mathcal{K}: L^2(\Omega) \to L^2(\Omega)$ is the blurring operator which is known and is a Fredhlom integral operator of the first kind

$$\mathcal{K}u(x,y) = \int_\Omega k(x,x',y,y')u(x',y')dx'dy'. \tag{3.2}$$

The kernel k, which describes the blurring, is known as the point spread function (PSF). In my work I assume that the blurring is spatially invariant i.e the PSF is of the form

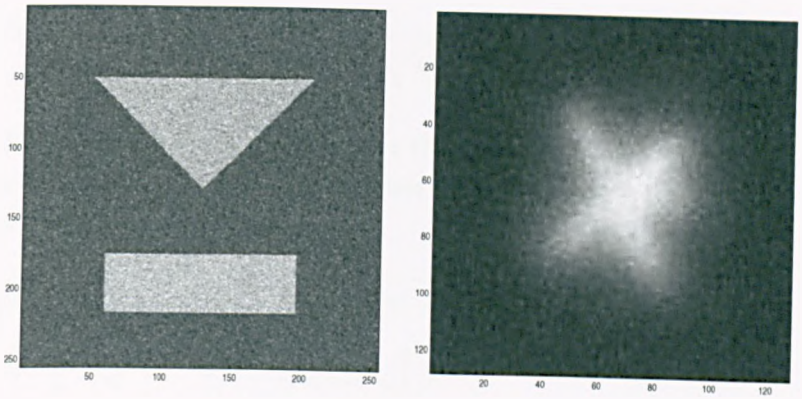
$$k(x,x',y,y') = k(x-x',y-y'). \tag{3.3}$$

If no blurring is present then we have what is known as a pure denoising problem and $\mathcal{K}$ is replaced by the identity operator. Most of my work has focused on the pure denoising problem. Figure 3.2 shows a noisy version of the triangle image seen above and a blurred noisy version of the satellite image.

The problem of recovering the true image u from the observed image z is an inverse problem and in the deblurring case is ill-posed, therefore, some sort of regularization procedure must be used in order to approximate u. There are two main approaches used in the literature, the first which we shall mainly focus on is the Tikhonov regularization (§2.3.4) approach in which the following unconstrained minimization problem is solved

$$\min_u \alpha R(u) + 1/2\|\mathcal{K}u - z\|^2_{L^2(\Omega)}. \tag{3.4}$$

Figure 3.2: Examples of noisy and blurred and noisy images



Here $R(u)$ is a regularization functional which penalizes against certain artifacts in the solution, the second term is a fit to data term which ensures goodness of fit to the observed data and α is a regularization parameter (usually chosen by experiment) which balances between the two. The second approach is the constrained regularization approach. In this approach the following constrained minimization problem is solved

$$\min_u R(u) \tag{3.5}$$

$$\text{subject to } \|\mathcal{K}u - z\|^2_{L^2(\Omega)} = \sigma^2 \tag{3.6}$$

where σ is the standard deviation of the noise, which is assumed to be known. The two approaches are similar in that the unconstrained problem can be viewed as a lagrange multiplier approach to solving the constrained problem with lagrange multiplier $\lambda = \frac{1}{2\alpha}$.

There are many possible regularization functionals which penalize non-smooth images e.g $\|u\|_{L^2}$ (the classic Tikhonov regularization) or $\int_\Omega |\nabla u|^2 dxdy$. The latter of these is known as the H^1 norm and Tikhonov regularization with this regularization functional leads to the following minimization problem

$$\min_u \int_\Omega \alpha/2|\nabla u|^2 dxdy + \frac{1}{2}(\mathcal{K}u - z, \mathcal{K}u - z)_{L^2(\Omega)}. \tag{3.7}$$

To find the condition for a minimum of this convex functional let us consider the more general functional (other functionals of this form are encountered later)

$$J(u) = \int_\Omega \alpha\Phi(\sqrt{u_x^2 + u_y^2 + \beta}) + 1/2(\mathcal{K}u - z)^2 dxdy. \tag{3.8}$$

in this case $\beta = 0$ and $\Phi(x) = 1/2x^2$. Assuming that Φ' exits, we have

$$\begin{aligned} \frac{d}{dt}J(u+tv) = \int_\Omega \alpha\Phi'(\sqrt{(u_x+tv_x)^2 + (u_y+tv_y)^2 + \beta})\frac{u_x+tv_x}{\sqrt{(u_x+tv_x)^2+(u_y+tv_y)^2+\beta}}v_x \\ +\alpha\Phi'(\sqrt{(u_x+tv_x)^2 + (u_y+tv_y)^2 + \beta})\frac{u_y+tv_y}{\sqrt{(u_x+tv_x)^2+(u_y+tv_y)^2+\beta}}v_y dxdy \\ +(\mathcal{K}u - z, \mathcal{K}v)_{L^2(\Omega)} + t(\mathcal{K}v, \mathcal{K}v)_{L^2(\Omega)} \end{aligned} \tag{3.9}$$

therefore the first variation (see §A.3.2) is

$$\frac{d}{dt}J(u+tv)|_{t=0} = \int_\Omega \left[\alpha\Phi'(\sqrt{u_x^2+u_y^2+\beta})\frac{\nabla u}{\sqrt{u_x^2+u_y^2+\beta}}\right].\nabla v dxdy + (\mathcal{K}u-z,\mathcal{K}v)_{L^2(\Omega)}. \tag{3.10}$$

Assuming that u is smooth enough we can integrate by parts (i.e use $\int_\Omega v\nabla.\vec{w}dxdy = -\int_\Omega \nabla v.\vec{w}dxdy + \int_\Gamma v\vec{w}.\vec{n}dS$) to rewrite the first term and using the adjoint (Theorem 2.2.5) we can rewrite the last term to get

$$\begin{aligned}\tfrac{d}{dt}J(u+tv)|_{t=0} &= -\int_\Omega \nabla.\left[\alpha\Phi'(\sqrt{u_x^2+u_y^2+\beta})\tfrac{\nabla u}{\sqrt{u_x^2+u_y^2+\beta}}\right]vdxdy \\ &+ \int_\Gamma v\left[\Phi'(\sqrt{u_x^2+u_y^2+\beta})\tfrac{\nabla u}{\sqrt{u_x^2+u_y^2+\beta}}\right].\vec{n}dS + (\mathcal{K}^*(\mathcal{K}u-z),v)_{L^2(\Omega)}\end{aligned}. \tag{3.11}$$

Imposing the so-called natural Neumann boundary condition $\nabla u.\vec{n} = 0$ on Γ the second term disappears. We have $\frac{d}{dt}J(u+tv)|_{t=0} = D_GJ(u)v$ where $D_GJ(u)$ is the Gateaux derivative of J at u and the first order condition for a minimum (or Euler-Lagrange equation) is $D_GJ(u) = 0$ (see §A.3.3) i.e

$$-\alpha\nabla.\left[\Phi'(\sqrt{u_x^2+u_y^2+\beta})\frac{\nabla u}{\sqrt{u_x^2+u_y^2+\beta}}\right] + \mathcal{K}^*(\mathcal{K}u-z) = 0. \tag{3.12}$$

In the case of (3.7) the Euler-Lagrange equation is

$$-\alpha\Delta u + \mathcal{K}^*\mathcal{K}u = \mathcal{K}^*z. \tag{3.13}$$

This equation is linear and can be solved fairly efficiently (after discretization) see for example [20, 79]. However because the regularization functional penalizes non-smooth images, the effect of this regularization will be noise removal but also a smoothing of the edges in the image. To overcome this disadvantage with classical noise removal techniques Rudin, Osher and Fatemi (ROF) in their seminal 1992 paper [78] introduced the Total Variation (TV) regularization functional

$$TV(u) = \int_\Omega |\nabla u|dxdy. \tag{3.14}$$

The TV regularization functional does not distinguish between smooth and piecewise smooth solutions with the same total variation, and thus Tikhonov regularization with the TV regularization functional can remove noise while still preserving the edges in an image, see §3.4 for more details.

Tikhonov regularization with the TV regularization functional involves the solution of the minimization problem

$$\min_u \int_\Omega \alpha|\nabla u| + \frac{1}{2}(\mathcal{K}u-z)^2dxdy. \tag{3.15}$$

Refereing to (3.12) the resulting Euler-Lagrange equation is

$$-\alpha\nabla.\left(\frac{\nabla u}{|\nabla u|}\right)+\mathcal{K}^*\mathcal{K}u=\mathcal{K}^*z. \tag{3.16}$$

This equation is degenerate when $|\nabla u|=0$ so in practice (3.14) is usually replaced by

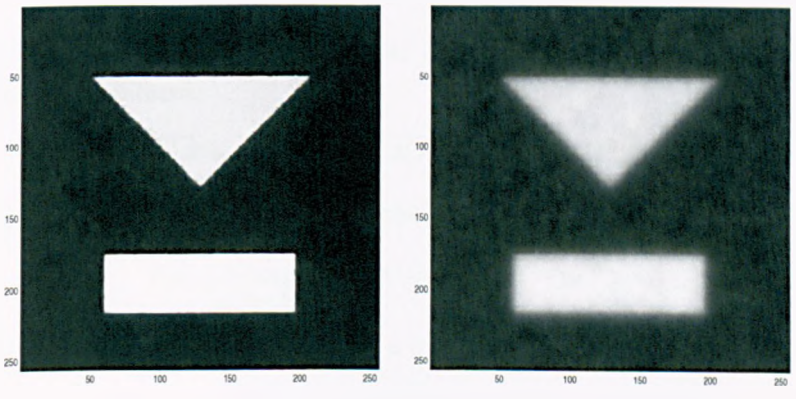
$$TV_\beta(u)=\int_\Omega\sqrt{u_x^2+u_y^2+\beta}dxdy, \tag{3.17}$$

where β is some small perturbing parameter. Replacing $TV(u)$ by $TV_\beta(u)$ in (3.15) the new Euler-Lagrange equation is

$$-\alpha\nabla.\left(\frac{\nabla u}{\sqrt{|\nabla u|^2+\beta}}\right)+\mathcal{K}^*\mathcal{K}u=\mathcal{K}^*z. \tag{3.18}$$

Unlike (3.13) this equation is highly nonlinear and the efficient solution of the discrete version of this equation using iterative methods in both the deblurring and the pure denoising cases has been an active area of research over the last decade or so and is the main focus of the work in this thesis.

Figure 3.3: Noise removal with TV (left) and H^1 (right) regularization functionals, the observed image is the one seen in Figure 3.2. Note the sharper edges in the TV case.



3.3 Theory of Total Variation Regularization

In this section I give a brief outline of the mathematical analysis of Total-Variation Regularization. For a more comprehensive treatment see the original work of Acar and Vogel [1] and also [17, 29, 74, 97, 94]

3.3.1 The space of functions of Bounded Variation

In for example [52] the space of functions of bounded variation in Ω, an open set in $\mathbb{R}^d$, is defined as

$$BV(\Omega) = \{u \in L^1(\Omega) : \int_\Omega |Du| < \infty\}, \tag{3.19}$$

where

$$\int_\Omega |Du| = \sup_{w \in \mathcal{W}} \int_\Omega -u \nabla . w dx \quad \mathcal{W} = \{w \in C_0^1(\Omega : \mathbb{R}^d) : |w(\mathbf{x})| \leq 1 \text{ for all } \mathbf{x} \in \Omega\}. \tag{3.20}$$

The space $BV(\Omega)$ is a Banach space with respect to the BV norm

$$\|u\|_{BV} = \|u\|_{L^1(\Omega)} + \int_\Omega |Du|. \tag{3.21}$$

furthermore $\int_\Omega |Du|$ is a seminorm with respect to $BV(\Omega)$.

3.3.2 The Dual Formulation

The seminorm $\int_\Omega |Du|$ is an extension of $TV(u)$ for non-smooth u and is often called the dual formulation of $TV(u)$. An argument using integration by parts shows that for u belonging to the Sobolev space $W^{1,1}(\Omega)$, which is a proper subset of $BV(\Omega)$ see [52], (3.20) is equivalent to (3.14).

$$\int_\Omega |Du| = \sup_{w \in \mathcal{W}} \int_\Omega \nabla u . w dx = \int_\Omega |\nabla u| = TV(u). \tag{3.22}$$

In [1] Acar and Vogel use (3.20) and the fact that the convex function $\sqrt{|\mathbf{q}|^2 + \beta}$ has the following dual representation

$$\sqrt{|\mathbf{q}|^2 + \beta} = \sup\{\mathbf{q}.\mathbf{v} + \sqrt{\beta(1 - |\mathbf{v}|^2)} : \mathbf{v} \in \mathbb{R}^d, |\mathbf{v}| \leq 1\} \tag{3.23}$$

derived using the techniques in §A.4.1, to introduce

$$F_\beta(u) = \sup_{w \in \mathcal{W}} \int_\Omega -u \nabla . w + \sqrt{\beta(1 - |w|^2)} dx, \tag{3.24}$$

which they show is equivalent to $TV_\beta(u)$ for $u \in W^{1,1}$, with the supremum attained for

$$w = \frac{\nabla u}{\sqrt{|\nabla u|^2 + \beta}}. \tag{3.25}$$

We see that $F_0(u) = \int_\Omega |Du|$. In [1] it is shown that $F_0(u) < \infty$ if and only if $F_\beta(u) < \infty$ for any $\beta > 0$ and $u \in L^1(\Omega)$, also for any $u \in BV(\Omega)$

$$\lim_{\beta \to 0} F_\beta(u) = F_0(u). \tag{3.26}$$

Using $F_\beta(u)$ the total variation regularization problem can be defined as

$$\min_{u \in BV(\Omega) \cap L^2(\Omega)} F_\beta(u) + 1/2 \|\mathcal{K}u - z\|^2_{L^2(\Omega)}, \tag{3.27}$$

which it can be shown under certain conditions on $\mathcal{K}$ is a well-posed problem.

Existence and Uniqueness

Theorem 3.3.1

The functional $F_\beta(u)$ defined above is weakly lower semicontinuous in $L^2(\Omega)$ and convex.

Proof

To prove weak lower semicontinuity let u_n converge weakly to u in $L^2(\Omega)$, then for any $v \in L^2(\Omega)$ the sequence $(u_n, v)_{L^2(\Omega)}$ converges and hence has infinum limit equal to its limit $(u, v)_{L^2(\Omega)}$. Taking $v = \nabla.w$ for $w \in \mathcal{W}$ we have

$$-(u, \nabla.w)_{L^2(\Omega)} + \int_\Omega \sqrt{\beta(1 - |w|^2)}dx = \lim_{n\to\infty} \inf -(u_n, \nabla.w)_{L^2(\Omega)} + \int_\Omega \sqrt{\beta(1 - |w|^2)}dx$$

$$\leq \lim_{n\to\infty} \inf F_\beta(u_n). \tag{3.28}$$

Taking the supremum over $\mathcal{W}$ gives the desired result.

To prove the convexity let $u, v \in L^2(\Omega)$, $\lambda \in [0, 1]$, and $w \in W$ then

$$F_\beta(\lambda u + (1 - \lambda)v) = \int_\Omega -(\lambda u + (1 - \lambda v)\nabla.w + \sqrt{\beta(1 - |w|^2)}dx$$

$$= \lambda \int_\Omega -u\nabla.w + \sqrt{\beta(1 - |w|^2)}dx + (1 - \lambda) \int_\Omega -v\nabla.w + \sqrt{\beta(1 - |w|^2)}dx$$

$$\leq \lambda F_\beta(u) + (1 - \lambda)F_\beta(v). \tag{3.29}$$

Taking the supremum over $\mathcal{W}$ gives the desired result. ∎

In the case $\mathcal{K} = I$, using the above result and the weak lower semicontinuity and coercivity of the $L^2(\Omega)$ norm, theorem A.2.1 implies that a global minimizer of (3.27) exists. The convexity of $F_\beta(u)$ together with the strict convexity of the $L^2(\Omega)$ norm ensures that the solution is unique (see §A.4). For the more general case see [1] for a proof of existence and uniqueness given certain conditions on $\mathcal{K}$. See [1] also for a proof of stability with respect to pertubations in z, α, β and $\mathcal{K}$.

If we rewrite (3.27) as

$$\min_u \sup_{w\in\mathcal{W}} \Phi(u, w). \tag{3.30}$$

with

$$\Phi(u, w) = \int_\Omega -\alpha u\nabla.w + \sqrt{\beta(1 - |w|)} + 1/2(\mathcal{K}u - z)^2, \tag{3.31}$$

then given that Φ is convex in u and concave in w we can interchange the min and sup to get

$$\sup_{w\in\mathcal{W}} \min_u \Phi(u, w). \tag{3.32}$$

The minimum over u is given by the solution of the equation

$$-\alpha\nabla.w + \mathcal{K}^*\mathcal{K}u - \mathcal{K}^*z = 0. \tag{3.33}$$

In the denoising case where $\mathcal{K}$ is the identity we can use this equation to eliminate u in Φ and taking $\beta = 0$ we get what is known as the dual formulation of the total variation regularization problem.

$$\sup_{w \in \mathcal{W}} \int_{\Omega} -\alpha z \nabla . w - 1/2\alpha^2 (\nabla . w)^2. \tag{3.34}$$

Note that this is equivalent to

$$\sup_{w \in \mathcal{W}} \int_{\Omega} \alpha z \nabla . w - 1/2\alpha^2 (\nabla . w)^2, \tag{3.35}$$

where if w^* is the value of w that maximises (3.34) then $-w^*$ maximises (3.35). I use (3.35) because it is the more commonly used formulation. The advantage of working with the dual formulation is that it is differentiable without the need for a perturbing parameter β.

The discussion in the rest of this chapter will focus on the denoising problem only, which is the focus of the next three chapters. Some specific issues relating to the deblurring problem will be addressed in Chapter 7 in which iterative methods for this problem are presented.

3.4 Properties of Total-Variation Regularization

In [90, 91] Strong and Chan prove some interesting properties of total variation regularization for the case of piecewise constant functions in the 1-dimensional case and in higher dimensions for radially symmetric functions. In $\mathbb{R}^1$ a simple piecewise constant function z is defined as follows

$$z(x) = \begin{cases} 1 & x \in \Omega_1 \\ 0 & x \in \Omega_2 \end{cases}. \tag{3.36}$$

Strong and Chan prove that the result of applying total variation regularization to this functional is to preserve exactly the discontinuity (or edge) while reducing the contrast of the function so that

$$u(x) = \begin{cases} 1 - \delta_1 & x \in \Omega_1 \\ \delta_2 & x \in \Omega_2 \end{cases} \tag{3.37}$$

with the change in intensity δ_i being proportional to the regularization functional α and inversely proportional to the length of the domain Ω_i.

$$\delta_i = \frac{\alpha}{|\Omega_i|}. \tag{3.38}$$

Furthermore this result is extended to noisy images z with

$$\frac{\int_{\Omega_1} z dx}{|\Omega_1|} = 1 \text{ and } \frac{\int_{\Omega_2} z dx}{|\Omega_2|} = 0. \tag{3.39}$$

With this result the effect of total variation regularization on any piecewise constant function can be analysed. If we split up a piecewise constant function z into n regions Ω_i on which z is constant then total variation regularization preserves the discontinuities and the change in intensity on each region can be classified as follows.

1. At the boundaries Ω_1 and Ω_n the change in intensity is $\frac{\alpha}{|\Omega_i|}$, this change is positive if z is larger on the neighbouring region and vice versa.

2. At an extremum, which is a region Ω_i such that z on Ω_i is greater (less) than z on both neighbouring regions Ω_{i+1} and Ω_{i-1}, there is a decrease (increase) in intensity of $\frac{2\alpha}{|\Omega_i|}$.

3. At a step, which is a region Ω_i such that z on Ω_{i-1} is less than z on Ω_i and z on Ω_{i+1} is greater than z on Ω_i (or vice versa), there is no change in the intensity.

This last result follows from the fact that the total variation of a nondecreasing (nonincreasing) piecewise constant function is just the total jump over the whole function, changing the value at the steps will not change the total variation but it will increase the value of the fitting term. If noise is added so that z has piecewise constant mean, results (1) and (2) hold, while result (3) holds approximately.

Note that in the discrete setting a single pixel of noise can be considered an extremum with width h where h is the grid spacing. Given that loss of intensity is inversely proportional to the width of the image feature we see how total variation regularization can remove noise while preserving other image features.

In two dimensions things are more complicated but for radially symmetric piecewise constant functions, the following results have been proved in [90, 91]

1. As in the 1-dimensional case the position of edges is preserved exactly.

2. In boundary regions $\delta_i = \frac{|\partial\Omega_{i,i-1}|}{|\Omega_i|}\alpha$.

3. In extremum regions $\delta_i = \frac{|\partial\Omega_{i,i+1}|+|\partial\Omega_{i,i-1}|}{|\Omega_i|}$.

4. In step regions $\delta_i = \frac{|\partial\Omega_{i,i+1}|-|\partial\Omega_{i,i-1}|}{|\Omega_i|}$.

Here $|\Omega_i|$ is the area of the region Ω_i and $|\partial\Omega_{i,j}|$ is the length of the boundary between Ω_i and Ω_j.

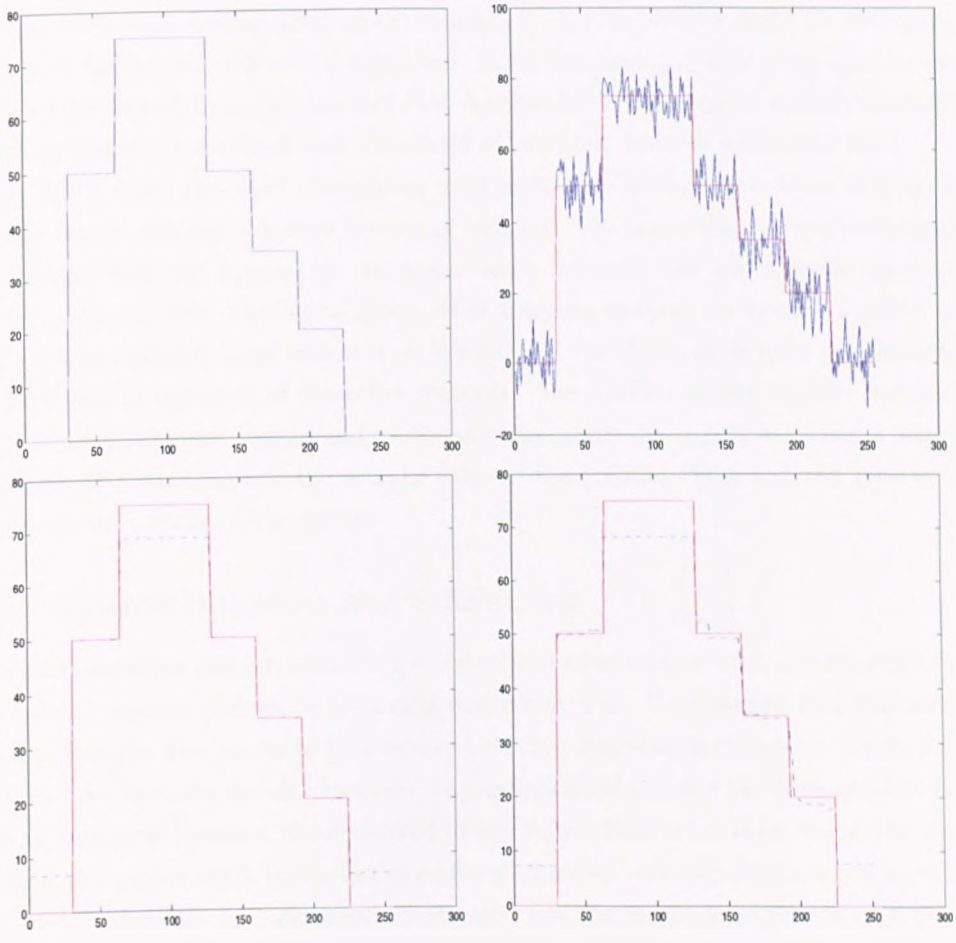
3.4.1 Scale and α

We see that for extremum regions the change in intensity is given above by

$$\delta = \frac{|\partial\Omega|}{|\Omega|}\alpha. \tag{3.40}$$

or in other words α times the length of the boundary of the feature divided by the area of the feature. Defining the scale of the feature as $\frac{|\Omega|}{|\partial\Omega|}$, the change in intensity is given as α divided by the scale. In fact in [91] it is shown that (away from the boundary where there may be for example some rounding of sharp corners) this formula is a very good estimate for the effect of total variation regularization on any constant image feature e.g a rectangle.

Figure 3.4: Clockwise from top left, a signal with boundary extremum and stepped regions, a noisy version of the signal, the result (dashed line) of applying total-variation regularization to the noisy signal and the result of applying total variation regularization to the true signal with the same α



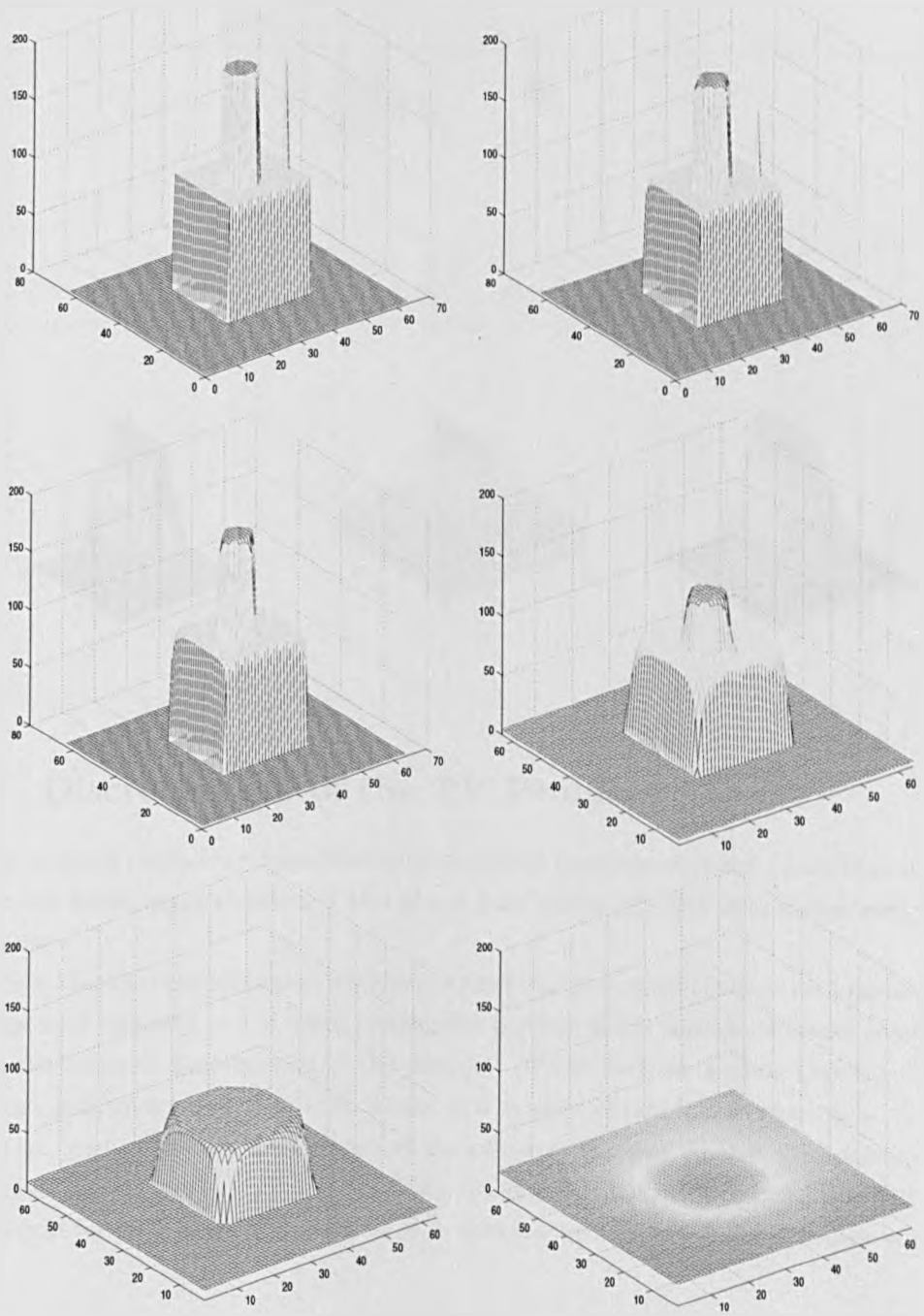
Using the above formula (3.40) relating the scale of an image feature with the loss of intensity due to total variation regularization and the fact that in a discrete setting every image is piecewise constant (even if only at the one pixel level) Strong, Aujol and Chan [89] investigate the effect of α on the scale in images. As the value of α increases effectively smaller scaled features are eliminated and merge to form larger scaled features until eventually if α is larger than some threshold value $\hat{\alpha}$ the resulting image is just the mean of the original image z (a flat image has zero total variation). I note here that for the results above Strong and Chan assume that α is large enough to remove the noise, while small enough to maintain all features originally present in the image (this cannot always be achieved). In [89] an algorithm is presented to find the smallest value $\tilde{\alpha}$ of α needed to remove all features in an image with scale smaller than some threshold. A suitable threshold for denoising, for example, would be the scale of a single pixel. Some interesting results from tests on various images are presented, including the fact that $\tilde{\alpha}$ seems to increase approximately linearly with scale threshold and for a fixed scale threshold of one pixel linearly with noise level.

Figure 3.5 shows the effect of applying total variation regularization to an original image (top left) which contains a square feature of intensity 100, a circular feature (with scale 2.5 times smaller than the square) on the square with intensity 200 and a single pixel feature also with intensity 200. The figure shows what happens as total variation is applied to this image with increasingly large values of α. We see that the single pixel spike is removed, with very little loss of intensity of the other features. The smaller scaled circular feature loses intensity faster than the square and eventually disappears, as α gets even larger the image heads towards a flat image at the average value of the original. Note also the greater loss of intensity at the corners of the square.

3.4.2 Smooth functions and Staircasing

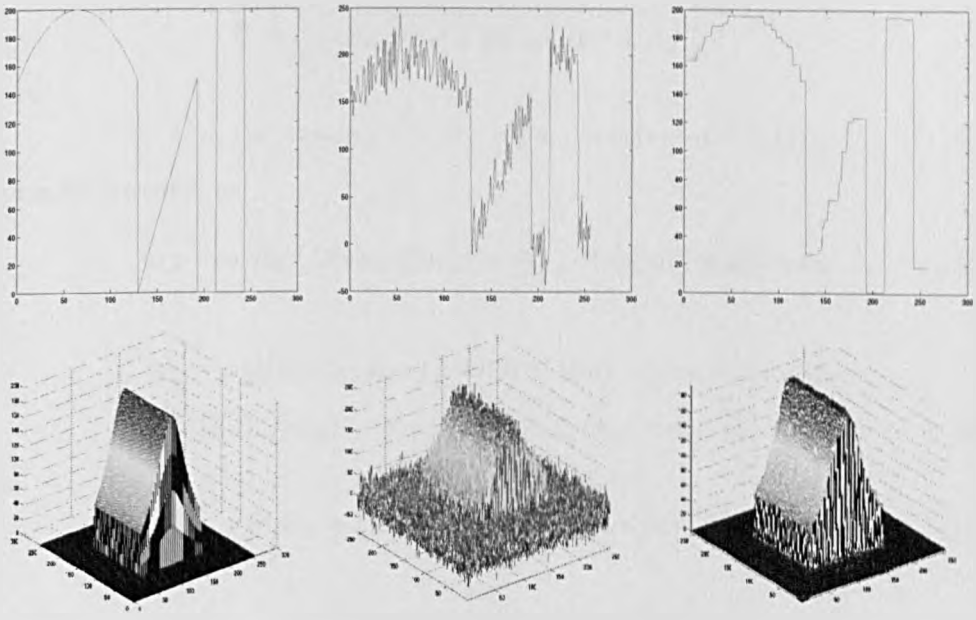
When total variation regularization is applied to noise free images with smooth regions, then these smooth regions tend to be preserved reasonably well. For example in 1-dimension the slope of a straight line, seems to be preserved, (in two dimensions things are less straightforward) see [90] for more details. However we are interested in noisy images and when noise is added to a smooth function, the recovered image suffers from what is known as the staircasing effect, the region which (prior to the addition of noise) was originally smooth is recovered as piecewise constant. In 1-dimension a straight line (or ramp) is recovered as a piecewise constant stepped function (a staircase). To provide insight as to why this occurs Chambolle and Lions [19] provide the following example: they consider the function $u_0(x) = x$ between 0 and m, they then assume that the addition of noise transforms this function into a non-decreasing piecewise constant function, using the results above we see that the only effect of total variation regularization (assuming α isn't large enough to merge image features) is to change the intensity of the two boundary regions, leaving the steps unchanged, this may be

Figure 3.5: From Left to right and down, the original image in mesh plot form and the effect of total variation with increasing values of α



an extreme example but it illustrates the point well.

Figure 3.6: The effect of staircasing in one and two dimensions. True signal/image (left), noisy observed signal/image (middle) and recovered signal/image using total-variation regularization (right)



3.5 Discretization of the TV Problem

In this section I outline the discretization of the Euler-Lagrange equation (3.18) that we shall use in the forthcoming chapters, I also give a brief survey of other approaches used in the literature .

Given that the observed image z is given in the form of $n \times m$ pixel values each representing average light intensity over a small rectangular portion of the domain, it seems sensible to use a cell-centered discretization of the domain. In our work we assume that the domain Ω is the unit square and then split it into $n \times m$ cells of size $h \times k$ where $h = 1/n$ and $k = 1/m$, grid points are then placed at the centre of the cells so that grid point (i, j) is located at $(x_i, y_j) = ((i - 1/2)h, (j - 1/2)k)$, the discrete domain is denoted Ω^h. The value of the grid function z_h at grid point (i, j) is denoted by z_{ij}. The Euler-Lagrange equation

(3.18) is discretized using a finite difference scheme. The equation at grid point (i,j) is

$$u_{i,j} - \alpha \left[\frac{\delta_x^-}{h} \left(\frac{\delta_x^+ u_{i,j}/h}{\sqrt{(\delta_x^+ u_{i,j}/h)^2 + (\delta_y^+ u_{i,j}/k)^2 + \beta}} \right) \right. \tag{3.41}$$

$$\left. + \frac{\delta_y^-}{k} \left(\frac{\delta_y^+ u_{i,j}/k}{\sqrt{(\delta_x^+ u_{i,j}/h)^2 + (\delta_y^+ u_{i,j}/k)^2 + \beta}} \right) \right] = z_{i,j}$$

, where

$$\delta_x^{\pm} u_{i,j} = \pm \left(u_{i\pm 1,j} - u_{i,j} \right) \qquad \delta_y^{\pm} u_{i,j} = \pm \left(u_{i,j\pm 1} - u_{i,j} \right). \tag{3.42}$$

This can be rewritten as

$$u_{i,j} - \alpha_h \left[\delta_x^- \left(D(u)_{i,j} \delta_x^+ u_{i,j} \right) + \gamma \delta_y^- \left(D(u)_{i,j} \gamma \delta_y^+ u_{i,j} \right) \right] = z_{i,j} \tag{3.43}$$

or

$$u_{i,j} - \alpha_h ((D(u)_{ij}(u_{i+1,j} - ui, j) - D(u)_{i-1,j}(u_{i,j} - u_{i-1,j})$$

$$+ \gamma^2 \left[D(u)_{i,j}(u_{i,j+1} - u_{i,j}) - D(u)_{i-1,j}(u_{i,j} - u_{i,j-1}) \right]) = z_{i,j}, \tag{3.44}$$

where

$$D(u)_{ij} = \left((\delta_x^+ u_{ij})^2 + (\gamma \delta_y^+ u_{ij})^2 + \beta_h \right)^{-1/2} \tag{3.45}$$

and

$$\alpha_h = \alpha/h, \ \beta_h = h^2 \beta \text{ and } \gamma = h/k, \tag{3.46}$$

with Neumann boundary condition

$$u_{i,0} = u_{i,1}, u_{i,m+1} = u_{i,m}, u_{0,j} = u_{1,j}, u_{n+1,j} = u_{n,j}. \tag{3.47}$$

I denote the discrete Euler-Lagrange equation defined by (3.43)-(3.47) by

$$N_h^{TV}(u_h) = z_h. \tag{3.48}$$

We discretize the Euler-Lagrange equation in this way for two reasons, firstly because this discretization scheme is the same as the one used by Chan et al in [29] and their primal-dual Newton method (see later) is one of the main methods to which we compare our work and secondly because (3.43) is equivalent to $\left(\nabla J_h^{TV} \right)_{i,j} = 0$ where

$$\min_{u_h} J_h^{TV}(u_h) \text{ with } J_h^{TV}(u_h) = \sum_{i,j} \alpha_h \sqrt{(\delta_x^+ u_{i,j})^2 + (\gamma \delta_y^+ u_{i,j})^2 + \beta_h} + \frac{1}{2}(u_{i,j} - z_{i,j})^2 \tag{3.49}$$

is a discrete version of the total-variation minimization problem i.e the discretization of the Euler-Lagrange equation for the continuous problem is equivalent to the condition for a minimum of the discrete problem.

3.5.1 Alternative Notation

I may at some points find it useful to use the following notation. If u is a scalar quantity defined on Ω^h then

$$(\text{Grad } u)_{i,j} = ((\text{Grad } u)^1_{ij}, (\text{Grad } u)^2_{ij})^T, \tag{3.50}$$

where

$$(\text{Grad } u)^1_{i,j} = \begin{cases} \delta^+_x u_{i,j} & i < n \\ 0 & i = n \end{cases} \tag{3.51}$$

$$(\text{Grad } u)^2_{i,j} = \begin{cases} \gamma\delta^+_y u_{i,j} & j < m \\ 0 & j = m \end{cases} \tag{3.52}$$

Also if w is a vector quantity defined on Ω^h so that $w_{ij} = (w^1_{ij}, w^2_{ij})^T$ then

$$(\text{Div } w)_{ij} = (\text{Div}^1 w)_{ij} + \gamma(\text{Div}^2 w)_{ij}, \tag{3.53}$$

where

$$(\text{Div}^1 w)_{i,j} = \begin{cases} \delta^-_x w^1_{i,j} & 1 < i < n \\ w^1_{i,j} & i = 1 \\ -w^1_{i-1,j} & i = n \end{cases} \tag{3.54}$$

$$(\text{Div }^2 w)_{i,j} = \begin{cases} \delta^-_y w^2_{i,j} & 1 < j < m \\ w^2_{i,j} & j = 1 \\ -w^2_{i,j-1} & j = m \end{cases} . \tag{3.55}$$

With this notation the discrete Euler-Lagrange equation is

$$-\alpha_h \text{Div}\left(\frac{\text{Grad } u_h}{\sqrt{|\text{Grad } u_h|^2 + \beta_h}}\right) + u_h = z_h. \tag{3.56}$$

I may also wish to use a matrix notation. Denote by

$$\mathbf{u}_h = (u_{1,1}, u_{2,1}, ..., u_{n,1}, u_{1,2}, ...u_{n,m})^T \tag{3.57}$$

the $N \times 1$ vector which results from stacking the grid function u_h along rows of pixels. Now define B^T_l for $l = 1, .., N$ as the $2 \times N$ matrix for which $B^T_l \mathbf{u}_h = (u_{l+1} - u_l, \gamma(u_{l+n} - u_l))^T$ with appropriate modifications if l corresponds to a boundary point so that if l corresponds

to the grid point (i,j) then $B_l^T\mathbf{u}$ is essentially (Grad $u)_{i,j}$.

$$B_l = \begin{bmatrix} 0 & 0 \\ . & . \\ 0 & 0 \\ -1 & -\gamma \\ 1 & 0 \\ 0 & 0 \\ . & . \\ 0 & \gamma \\ 0 & 0 \\ . & . \\ 0 & 0 \end{bmatrix} \begin{matrix} \text{row } 1 \\ . \\ . \\ \text{row } l \\ \text{row } l+1 \\ . \\ . \\ \text{row } l+n \\ . \\ . \\ \text{row } N \end{matrix}$$

Now if we define

$$B = [B_1, .., B_N] \tag{3.58}$$

we see that $B^T\mathbf{u}$ is a $2N \times 1$ vector equivalent to $((\text{Grad u})_{1,1}^T, ..., (\text{Grad } u)_{n,m}^T)^T$. Also if we have a $2N \times 1$ vector $\mathbf{w} = (\mathbf{w}_1^T, .., \mathbf{w}_N^T)^T = (w_{1,1}^T, ..., w_{n,m}^T)^T$ which corresponds to stacking the discrete vector quantity w_h into a vector, then we see that (for non boundary points)

$$(B\mathbf{w})_l = -(\mathbf{w}_l)_1 - \gamma(\mathbf{w}_l)_2 + (\mathbf{w}_{l-1})_1 + \gamma(\mathbf{w}_{l-n})_2. \tag{3.59}$$

The first two terms coming from $B_l\mathbf{w}_l$ the third term form $B_{l-1}\mathbf{w}_{l-1}$ and the last term from $B_{l-n}\mathbf{w}_{n-1}$. In other words if l corresponds to the grid point (i,j) then $(B\mathbf{w})_l$ is equivalent to $-(\text{Div } w)_{i,j}$. In this notation the Euler-Lagrange equation is

$$\left(I + \alpha_h BE(\mathbf{u}_h)^{-1}B^T\right)\mathbf{u}_h = \mathbf{z}_h, \tag{3.60}$$

where E is a $N \times N$ block diagonal matrix with diagonal blocks and the block ll is $\left(\sqrt{|B_l^T\mathbf{u_h}| + \beta_h}\right) I_2$ and I_2 is the identity matrix of size 2.

3.5.2 Alternative Discretizations

The approach outlined above is not the only, method of discretizing the Euler-Lagrange equation used in the literature. In [78] the nonlinear term is approximated by

$$\frac{1}{h}\left[\delta_x^-\left(D^x(u)_{i,j}\delta_x^+ u_{i,j}\right) + \gamma\delta_y^-\left(D^y(u)_{i,j}\delta_y^+ u_{i,j}\right)\right], \tag{3.61}$$

where

$$D^x(u)_{i,j} = \left((\delta_x^+ u_{ij})^2 + (\gamma(m(\delta_y^+ u_{ij}, \delta_y^- u_{i,j})))^2 + \beta_h\right)^{-1/2} \tag{3.62}$$

and

$$D^y(u)_{i,j} = \left((m(\delta_x^+ u_{ij}, \delta_x^- u_{i,j}))^2 + (\gamma\delta_y^+ u_{ij})^2 + \beta_h\right)^{-1/2}. \tag{3.63}$$

Here m is the minmod function defined by

$$m(a,b) = \left(\frac{\text{sgn } a + \text{ sgn } b}{2}\right) \min(|a|,|b|). \tag{3.64}$$

Some evidence that this scheme better preserves the corners of image features is presented in [90].

In [93] Vogel uses the following (central differencing) scheme (equivalent to a finite volume discretization) to discretize (3.18)

$$u_{i,j} - \alpha\left[\frac{\delta_x^c}{h^2}\left(D^c(u)_{i,j}\delta_x^c u_{i,j}\right) + \frac{\delta_y^c}{k^2}\left(D^c(u)_{i,j}\delta_y^c u_{i,j}\right)\right] = z_{i,j} \tag{3.65}$$

or

$$u_{i,j} - \alpha\left[\frac{1}{h^2}\left((D^c(u)_{i+1/2,j}(u_{i+1,j} - u_{i,j}) - D^c(u)_{i-1/2,j}(u_{i,j} - u_{i-1,j}\right)\right.$$

$$\left. + \frac{1}{k^2}\left(D^c(u)_{i,j+1/2}(u_{i,j+1/2} - u_{i,j}) - D^c(u)_{i,j-1/2}(u_{i,j} - u_{i,j-1})\right)\right] = z_{i,j}, \tag{3.66}$$

where $\delta_x^c u_{i,j} = (u_{i+1/2,j} - u_{i-1/2,j})$ and $\delta_y^c u_{i,j} = (u_{i,j+1/2} - u_{i,j-1/2})$. In this case the diffusion terms D^c must be evaluated at the x-edge midpoints $(x_{i\pm1/2}, y_j)$ and y-edge midpoints $(x_i, y_{j\pm1/2})$ of the cells. For example

$$D^c(u)_{i+1/2,j} = \left[\left(\frac{\delta_x^c u_{i+1/2,j}}{h}\right)^2 + \left(\frac{\hat{\delta}_y u_{i+1/2,j}}{k}\right)^2 + \beta\right]^{-1/2} \tag{3.67}$$

with

$$\hat{\delta}_y u_{i+1/2,j} = \frac{1}{4}\left(\delta_y^c u_{i,j-1/2} + \delta_y^c u_{i,j+1/2} + \delta_y^c u_{i+1,j-1/2} + \delta_y^c u_{i+1,j+1/2}\right). \tag{3.68}$$

Another approach used in [71, 41] is to expand out the nonlinear term as follows

$$\nabla.\left(\frac{\nabla u}{\sqrt{u_x^2 + u_y^2 + \beta}}\right) = \frac{(u_x^2 + \beta)u_{yy} + (u_y^2 + \beta)u_{xx} - 2u_x u_y u_{xy}}{(u_x^2 + u_y^2 + \beta)^{3/2}} \tag{3.69}$$

and then use central differencing approximations of u_x, u_y, u_{xx}, u_{yy} and u_{xy}.

As well as differences in the finite difference schemes used, there are also different approaches to the choice and discretization of the image domain. Some authors e.g. [48] use a vertex rather than a cell-centered discretization of the domain. Also the choice of the image domain Ω is somewhat arbitrary, there are two main approaches, the first is the approach outlined above to take Ω to be the unit square whatever the size of the image, the other is to take the domain to be such that the grid spacing in each direction is 1, e.g if the image is of size 256×256 then $\Omega = (0,256) \times (0,256)$, this approach is used in, for example, [41, 71]. If I were to replace α_h and β_h by simply α and β in (3.43) this would be equivalent to using this type of discretization. The value of α_h, β_h for $\Omega = (0,1) \times (0,1)$ should be the same as α, β for $\Omega = (0,256) \times (0,256)$.

Remark 3.5.1 *Assuming for the moment that $h = k$, the scale of a single pixel of noise is $h/4$. Assume that we have a particular image that was sampled at say 128×128 and 256×256 pixels and a similar level of noise was present in each case. If Ω is the unit square then the scale of a single pixel in the 256×256 case will be half what it was in the 128×128 case and from (3.40) we see that the value of α needed to remove the noise will be twice what it was in the 128×128 case, however the value of $\alpha_h = \alpha/h$ (which is what we actually choose in the algorithm) will remain unchanged. On the other hand if Ω is chosen so that $h = 1$ always, then the value of α needed to remove the noise in the two cases will be the same.*

3.6 Solving the TV Problem

There have been various different approaches proposed in the literature for solving the TV regularization problem, here I give a brief review.

3.6.1 Explicit Time Marching

In [78] Rudin et al solve the Euler-Lagrange equation using an artificial time marching method. The equation is solved by using an explicit time marching (forward Euler) method to find the steady state of the following parabolic equation

$$u_t = \alpha \nabla . \left(\frac{\nabla u}{\sqrt{|\nabla u|^2 + \beta}} \right) - (u - z) \tag{3.70}$$

So on step $k+1$, u_h is updated by

$$u_h^{k+1} = u_h^k - \Delta t N_h^{TV}(u_h^k), \tag{3.71}$$

where Δt is the time step. I remark here that a steepest descent method (§A.6.4) will also involve updating u_h on each step via equation (3.71), but with Δt replaced by a step length parameter, determined via a line search on J_h^{TV}.

I note here that in [78] Rudin et al actually solve the constrained problem, the lagrange multiplier λ is found by multiplying both sides of the Euler-Lagrange equation by $(u - z)$ and integrating over Ω.

$$\lambda \int_\Omega (u - z)^2 dxdy = \int_\Omega (u - z) \nabla . \left(\frac{\nabla u}{\sqrt{|\nabla u|^2 + \beta}} \right) dxdy. \tag{3.72}$$

Making use of the constraint the left hand side of (3.72)is $2\lambda\sigma^2$. Using

$$\int_\Omega v \nabla . \vec{w} dxdy = - \int_\Omega \nabla v . \vec{w} dxdy + \int_\Gamma v \vec{w} . \vec{n} dS \tag{3.73}$$

with $v = (u - z)$ and $\vec{w} = \frac{\nabla u}{\sqrt{|\nabla u|^2 + \beta}}$ and the fact that $\frac{\partial u}{\partial n} = 0$ on the boundary, the right hand side of (3.72) becomes

$$- \int_\Omega \nabla (u - z) . \frac{\nabla u}{\sqrt{|\nabla u|^2 + \beta}} \tag{3.74}$$

and so

$$\lambda = -\frac{1}{2\sigma^2}\int_\Omega \nabla(u-z).\frac{\nabla u}{\sqrt{|\nabla u|^2+\beta}} \tag{3.75}$$

A discretization of this equation is used to update λ on each step of the time marching method.

The main disadvantage of the explicit time marching approach is that to ensure stability a restriction must be imposed on the time step Δt, this results in very slow convergence of the method. In [71] Marquina and Osher reduce the restriction on the time step by multiplying the right hand side of (3.70) by $|\nabla u|$. The new equation can be written as

$$\begin{aligned} u_t &= |\nabla u|(z-u) + \alpha\sqrt{|\nabla u|^2+\beta}\nabla.\left(\frac{\nabla u}{\sqrt{|\nabla u|^2+\beta}}\right) \\ &= |\nabla u|(z-u) + \alpha\frac{(u_x^2+\beta)u_{yy} + (u_y^2+\beta)u_{xx} - 2u_x u_y u_{xy}}{u_x^2+u_y^2+\beta}. \end{aligned} \tag{3.76}$$

Importantly note that β is included in the $|\nabla u|$ term multiplying the second term on the right hand side but not the first term, central differencing is used to discretize the second term while an upwind differencing scheme is used for the first term, the numerical steady state obtained is therefore different to the straight TV case and is less staircased.

3.6.2 The Fixed Point Method

A method that can be viewed as a semi-implicit time marching method with infinite time step is the 'lagged diffusivity' fixed point method of Vogel and Oman [93]. In this method the Euler-Lagrange equation is linearized by freezing the term $\frac{1}{\sqrt{|\nabla u|^2+\beta}}$ at the value of the previous iterate. Therefore on step $k+1$ of the method we have a linear equation of the form

$$u^{k+1} - \alpha\nabla.\left(\frac{\nabla u^{k+1}}{\sqrt{|\nabla u^k|^2+\beta}}\right) = z \tag{3.77}$$

to solve in order to update the approximation. In [46] Dobson and Vogel analyse this method and show that it is globally convergent.

The linear system which results from the discretization of (3.77) is

$$\left(I + \alpha_h B E\left((\mathbf{u}_h)^k\right)^{-1} B^T\right)(\mathbf{u}_h)^{k+1} = \mathbf{z}^h. \tag{3.78}$$

From the fact that $\mathbf{v}^T B E^{-1} B^T \mathbf{v} = \left(B^T\mathbf{v}\right)^T E^{-1} B^T \mathbf{v}$ and the fact that all the entries of E are positive we see that the system is symmetric positive definite. Several different methods have been used in the literature to solve (3.78), these include preconditioned conjugate gradient (PCG) with incomplete Cholesky preconditioner [29], geometric multigrid [96] (either on its own or as a preconditioner for preconditioned conjugate gradient) and Algebraic multigrid [34], which is more robust with respect to small values of β than geometric multigrid. In

practice accurate solution of the linear equation is not necessary and the most efficient method usually results from reducing the linear residual by some small amount e.g. a factor of 10 before updating u.

In Chapter 5 the fixed point method with AMG linear solver is studied and a technique to reduce the overall cost proposed.

3.6.3 Newton's Method

Writing the Euler-Lagrange equation as

$$g(u) = -\alpha\nabla.\left(\frac{\nabla u}{\sqrt{|\nabla u|^2+\beta}}\right) + u - z = 0 \tag{3.79}$$

we can use Newton's method to solve it. Starting from some initial guess, on each step of the method the update δu is found by solving

$$g'(u)\delta u = -g(u) \tag{3.80}$$

where $g'(u)$ is the Frechet derivative of the operator g (see §A.3.1). Using $(|\nabla u|^2)' = 2\nabla u^T\nabla$ we have

$$g'(u) = -\alpha\nabla.\left[\frac{\nabla}{\sqrt{|\nabla u|^2+\beta}} + \nabla u\left(\frac{\nabla u^T\nabla}{(\sqrt{|\nabla u|^2+\beta})^3}\right)\right] + I \tag{3.81}$$

which can alternatively be written as

$$-\alpha\nabla.\left[\frac{1}{\sqrt{|\nabla u|^2+\beta}}\left(I - \frac{\nabla u\nabla u^T}{|\nabla u|^2+\beta}\right)\nabla\right] + I \tag{3.82}$$

Newton's method is quadratically convergent provided the initial guess is within the domain of convergence (close enough to the solution), however Chan et al in [32] show that the domain of convergence for Newton's Method in this case is very small for small values of β. To overcome this problem they propose a continuation procedure on both β and α. Starting with some small value of α and some large value of β and the noisy image as initial guess they solve the Euler-Lagrange equation using Newton's Method. With β fixed the value of α is increased and the new problem solved with the solution to the previous problem as initial guess. This procedure is repeated until the desired value of α has been reached. With α fixed a continuation procedure on β is then applied.

The sparse linear system which results from discretizing (3.80) with our usual finite difference operators is symmetric positive definite and is also equivalent to the Hessian of the discrete minimization problem (3.49). An alternative to the above continuation procedure is therefore to apply a line search to the Newton direction (see §A.6.4), this was done in [97]. The discrete linear system is solved using the conjugate gradient method.

3.6.4 The Primal-Dual Newton Method

To overcome the problems associated with Newton's method for the TV problem in a more fundamental way Chan et al [29] introduce the dual variable

$$w = \frac{\nabla u}{\sqrt{|\nabla u|^2 + \beta}} \tag{3.83}$$

The original system can then be replaced by the equivalent (u, w) system.

$$\begin{aligned} -\alpha\nabla.w + u - z = 0 = f_1(w,u) \\ w\sqrt{|\nabla u|^2 + \beta} - \nabla u = 0 = f_2(w,u) \\ |w(x,y)| \leq 1 \text{ for all } (x,y) \end{aligned} . \tag{3.84}$$

The first equation here is just (3.33), the second equation is from (3.25) and the constraint is the usual constraint on the dual variable.

The Primal-dual system (3.84) is much better behaved with respect to Newton's method than the Euler-Lagrange equation of the original primal problem (because it is 'more' linear), in fact the method presented in [29] seems to be globally convergent with quadratic convergence, without the need for a continuation procedure. The linearization of (3.84) results in the following linear system which must be solved on each step:

$$\begin{bmatrix} I & -\alpha\nabla. \\ \frac{w\nabla u^T\nabla}{\sqrt{|\nabla u|^2+\beta}} - \nabla & \sqrt{|\nabla u|^2 + \beta} \end{bmatrix} \begin{bmatrix} \delta u \\ \delta w \end{bmatrix} = \begin{bmatrix} -f_1(u,w) \\ -f_2(u,w) \end{bmatrix} \tag{3.85}$$

Eliminating δw using the second equation gives

$$\delta w = \frac{1}{\sqrt{|\nabla u|^2 + \beta}} \left[-f_2(w,u) + \left(I - \frac{w\nabla u^T}{\sqrt{|\nabla u|^2 + \beta}} \right) \nabla \delta u \right]. \tag{3.86}$$

Then δu is obtained by solving

$$\left[-\alpha\nabla. \left(\frac{1}{\sqrt{|\nabla u|^2 + \beta}} \left(I - \frac{w\nabla u^T}{\sqrt{|\nabla u|^2 + \beta}} \right) \nabla \right) + I \right] \delta u = -f_1(w,u) + \alpha\nabla. \left(\frac{-f_2(u,w)}{\sqrt{|\nabla u|^2 + \beta}} \right). \tag{3.87}$$

Given that

$$\frac{-f_2(w,u)}{\sqrt{|\nabla u|^2 + \beta}} = -w + \frac{\nabla u}{\sqrt{|\nabla u|^2 + \beta}} \tag{3.88}$$

the rhs in (3.87) is

$$\alpha\nabla.w - (u-z) + \alpha\nabla. \left(-w + \frac{\nabla u}{\sqrt{|\nabla u|^2 + \beta}} \right) = \alpha\nabla. \left(\frac{\nabla u}{\sqrt{|\nabla u|^2 + \beta}} \right) - (u-z) = -g(u), \tag{3.89}$$

where $g(u) = 0$ is the Euler-Lagrange equation for the primal problem, and

$$\delta w = -w + \frac{\nabla u}{\sqrt{|\nabla u|^2 + \beta}} + \frac{1}{\sqrt{|\nabla u|^2 + \beta}} \left(I - \frac{w\nabla u^T}{\sqrt{|\nabla u|^2 + \beta}} \right) \nabla \delta u. \tag{3.90}$$

We also have the condition that $|w(x,y)| \leq 1 \; \forall (x,y)$. To maintain this condition Chan et al apply a line search to the discrete version of δw, with the step length t chosen such that

$$t = 0.9 \sup\{\kappa : |w_{ij} + \kappa \delta w_{ij}| \leq 1 \forall i,j\}. \tag{3.91}$$

We see that the main cost associated with the primal-dual Newton method is the solution of the linear system $Hdu = -g(u)$ which results from discretization of (3.87). In [29] Chan et al state that H is positive definite provided α is positive and $|w_{i,j}| \leq 1$ at all grid points, however H is not symmetric. In order that the preconditioned conjugate gradient method can be used, Chan et al use an approximate Newton's method in which H is replaced by its symmetrization $1/2(H + H^T)$, which is equivalent to a discretization of

$$\left[-\alpha \nabla . \left(\frac{1}{\sqrt{|\nabla u|^2 + \beta}} \left(I - \frac{1}{2} \frac{w \nabla u^T + \nabla u w^T}{\sqrt{|\nabla u|^2 + \beta}} \right) \nabla \right) + I \right]. \tag{3.92}$$

Preconditied conjugate gradient with incomplete Cholesky preconditioner (§A.6.3) is then used to solve the linear system on each step. To prevent oversolving the following stopping criteria is used for the linear solver in [29]; on step $k+1$ of Newton's method the inner PCG iterations are stopped when the relative linear residual is less than

$$\min(0.1, 0.9 \|g(u^k)\|_2^2 / \|g(u^{k-1}\|_2^2). \tag{3.93}$$

3.6.5 Dual Approaches

In the primal-dual method Chan et al solve a system which contains both the primal and the dual variables, other authors have worked directly with the dual problem (3.35). In [17] Carter presents several relaxation methods to solve the discretization of (3.35). Another more recent approach is that used by Chambolle in [18]. Writing the dual problem as

$$\sup_{w \in W} \int_\Omega -1/2(\alpha \nabla . w - z)^2 + 1/2 z^2. \tag{3.94}$$

We can write the discrete dual problem as

$$\min_w \sum_{i,j} (\alpha_h ((\text{Div } w)_{i,j}) - z_{i,j})^2 \tag{3.95}$$

$$\text{subject to } [(w_{ij}^1)^2 + (w_{ij}^2)^2] - 1 \leq 0 \; \forall (i,j) \tag{3.96}$$

The Karush-Kuhn-Tucker (KKT) (see §A.5) conditions for this problem are

$$2\alpha_h \begin{bmatrix} (\alpha_h(\text{Div } w)_{i,j} - z_{ij}) - (\alpha_h(\text{Div } w)_{i+1,j} - z_{i+1,j}) \\ (\alpha_h(\text{Div } w)_{i,j} - z_{ij}) - (\alpha_h(\text{Div } w)_{i,j+1} - z_{i,j+1}) \end{bmatrix} + \hat{\theta}_{i,j} \begin{bmatrix} 2w_{i,j}^1 \\ 2w_{i,j}^2 \end{bmatrix} = 0 \tag{3.97}$$

for all (i,j) with either $\hat{\theta}_{i,j} > 0$ and $|w_{i,j}| = 1$ or $\hat{\theta}_{i,j} = 0$ and $|w_{i,j}| < 1$. Since the function and the constraints are convex, the KKT conditions are necessary and sufficient. Defining $\theta_{i,j} = \frac{\hat{\theta}_{i,j}}{\alpha_h}$ the KKT conditions can be written as

$$-(\text{Grad } (\alpha_h \text{Div } w - z))_{i,j} + \theta_{i,j} w_{i,j} = 0 \; \forall (i,j) \tag{3.98}$$

with

$$\theta_{i,j} > 0 \text{ and } |w_{i,j}| = 1 \tag{3.99}$$

$$\text{or } \theta_{i,j} = 0 \text{ and } |w_{i,j}| < 1 \tag{3.100}$$

In [18] Chambolle makes the following observation; if condition (3.99) is satisfied then from (3.98)

$$\theta_{i,j} = |\left(\text{Grad}\ (\alpha_h \text{Div}\ w - z)\right)_{i,j}|. \tag{3.101}$$

However if the second condition (3.100) is satisfied we see from (3.98) that (3.101) also holds on account of $|\left(\text{Grad}\ (\alpha_h \text{Div}\ w - z)\right)_{i,j}|$ also being equal to zero. The KKT conditions can therefore be written simply as

$$-\left(\text{Grad}\ (\alpha_h \text{Div}\ w - z)\right)_{i,j} + |\left(\text{Grad}\ (\alpha_h \text{Div}\ w - z)\right)_{i,j}| w_{i,j} = 0\ \forall (i,j). \tag{3.102}$$

Chambolle uses a semi-implicit time marching method to solve this discrete system so that on step $n+1$, w is updated as follows

$$w_{i,j}^{n+1} = \frac{w_{ij}^n + \Delta t \left(\text{Grad}\ (\alpha_h \text{Div}\ w^n - z)\right)_{i,j}}{1 + \Delta t |\left(\text{Grad}\ (\alpha_h \text{Div}\ w^n - z)\right)_{i,j}|}. \tag{3.103}$$

Where $w^0 = 0$.

3.6.6 Multilevel Approaches

In Chapter 4 I present a nonlinear multigrid method for solving the discrete Euler-Lagrange equation, nonlinear multigrid is also used by Frohn-Schauf et al in [48]. Chan and Chen [25] and Chen and Tai [39] have multilevel methods working directly with the minimization problem (not the Euler-Lagrange equation) which can solve the problem with very small values of β_h. For a brief introduction to this kind of approach see §A.7.2. Cascadic Multigrid (§2.6.13) is employed by Oehsen in [74] to improve the performance of fixed point and time marching methods, using both linear and nonlinear WENO interpolation operators.

3.6.7 Active Set Methods

Finally I mention that active set methods have been used by Karkkainen and Majava [61] and Ito and Kunisch [58].

3.7 Measuring Image Quality: SNR and PSNR

The signal to noise ratio or SNR of a noisy image is a measure of how much noise is present in the image (the smaller the SNR the more noise there is), it is given by the following formula

$$SNR = \frac{\sum_{(i,j)} (u_{i,j} - \bar{u})^2}{\sum_{(i,j)} (n_{i,j})^2}. \tag{3.104}$$

here $u_{i,j}$ is the value of the true image at the grid point (i,j), $\bar{u}$ is the average pixel value of u and $n_{i,j}$ is the value of the noise at grid point (i,j). In real applications of course neither the true image or the noise will be known, but in the simulations carried out later we do have the true image and the noise, which is added artificially, available. The Peak Signal to Noise Ratio or PSNR is used to measure how close two images (of the same size) are to each other, it is given by the following formula

$$PSNR(u,v) = 20\log_{10}(255/RMSE) \qquad RMSE = \sqrt{\frac{\sum_{(i,j)}(u_{i,j}-v_{i,j})^2}{N}}, \tag{3.105}$$

where N is the total number of pixels. The PSNR is not an absolute measure, just a relative measure, it can be used for example to compare how close the results of two different denoising processes (say TV and H^1) are to the true image.

Often people will also refer to the 'eyeball norm' when comparing images, this just refers to how the image appears to the human eye, for example two reconstructed images may have similar PSNR values when compared to the original, but one may look better than the other because it is smooth and not staircased in regions where it is expected to be smooth.

3.8 Beyond The ROF Model

Finally I mention that there have been many attempts in the literature to improve upon and extend the standard ROF model. One of the major areas of research has been to develop denoising methods which preserve edges as well as the ROF model while also recovering better the smooth regions present in the original image (reducing the staircasing effect), I go into these methods in more detail in Chapter 6 and so will not mention them any further here.

Other ways to improve the ROF model include the iterative regularization method of Osher et al [75], in which the $(k+1)$th iterate u^{k+1} results from minimizing (3.15) with z replaced by $z+v^k$ where v^k is the noise from the previous step and the related inverse scale space method of Burger et al [16] in which one starts from the zero image and gradually adds back information arriving eventually (if the method is not stopped) at the original noisy image. Another active area of research is on the use of the TV regularization term with alternative fidelity terms. In [26] Chan and Esedoglu consider the L^1 norm as fidelity term. It is well known that for any nonzero α the standard ROF model will reduce the contrast of image features (at a rate inversely proportional to their scale) but with this new model the contrast of image features tends to be preserved until they disappear at some threshold value of α. Yin et al [104] also use TV-L^1 for decomposing an image into cartoon and texture parts. Several approximations of Meyers G norm [76, 98] have also been used as fidelity term for texture extraction and denoising. By combining these alternative fidelity terms with

some of the staircase reducing regularization functionals Chan et al [28] and Levine [68] have combined staircase reduction (in the cartoon image) with texture extraction.

Diffusion filtering is an alternative denoising approach in which a nonlinear parabolic equation, usually of the general form $u_t = \nabla.(g(|\nabla u|)\nabla u)$ is marched forward in time. The noisy image is used used as initial guess and when the process is stopped determines the quality of the image (at convergence the image will be flat). Specific choices of g include the original choice of Perona and Malik [77], $g(x) = 1/(1+x^2)$ and the TV filterring choice $g(x) = 1/\sqrt{x^2+\beta}$. The connection between diffusion filtering and regularization is investigated in [86].

Chapter 4

A Nonlinear Multigrid Method For Total Variation Denoising

Useful Section References: §2.5, §2.6, §3.2, §3.5, §3.6, §A.7
Main Reference Material: [12, 29, 48, 57, 73, 78, 83, 92, 96, 97, 100, 102]

In this chapter I present my attempts to develop a nonlinear multigrid method based on the FAS (§2.6.8), for solving the discrete nonlinear equation $N_h^{TV}(u_h) = z_h$ (as defined in §3.5) which results from Tikhonov regularization of the denoising problem with the total variation regularization functional. The material in this chapter is based on work published by myself and K. Chen in [83].

Recall from §2.6 that a (non)linear multigrid method is defined by the choice of the coarse grid, transfer operators and the smoother, in our work we use standard coarsening and the standard cell-centered transfer operators defined in §2.6.3, the main focus of the work is on the choice of a suitable smoother for use in the nonlinear multigrid method. I first outline my experience using the standard approach of Gauss-Seidel Newton and then consider alternatives to this approach, comparisons of the nonlinear multigrid method with various smoothers against each other and against the fixed point and primal-dual Newton methods are made.

4.1 Choice of Smoother

We tried several different iterative methods as smoothers for our nonlinear multigrid method here we give the details of each of them, to aid the description recall that we can write the

discrete nonlinear equation at grid point (i,j) as

$$u_{i,j} - \alpha_h((D(u)_{ij}(u_{i+1,j} - u_{i,j}) - D(u)_{i-1,j}(u_{i,j} - u_{i-1,j})$$

$$+\gamma^2 [D(u)_{i,j}(u_{i,j+1} - u_{i,j}) - D(u)_{i,j-1}(u_{i,j} - u_{i,j-1})]) = z_{i,j}, \tag{4.1}$$

where

$$D(u)_{ij} = \left((\delta_x^+ u_{ij})^2 + (\gamma \delta_y^+ u_{ij})^2 + \beta_h\right)^{-1/2} \tag{4.2}$$

with appropriate modifications at the boundary.

4.1.1 Gauss-Seidel Newton

If at (i,j) we freeze all non (i,j) terms in (4.1) at the value of the most recent approximation $\bar{u}_{ij}$ we have a nonlinear equation in one variable to solve in order to update u_{ij}.

$$u_{ij} - \alpha_h g(u_{ij}) = z_{ij}. \tag{4.3}$$

This can be approximated using Newton's method as

$$u_{ij} - \alpha_h \left[g(\bar{u}_{ij}) + c(\bar{u}_{ij})(u_{ij} - \bar{u}_{ij})\right] = z_{ij}, \tag{4.4}$$

where $c(u_{ij}) = \frac{d}{du_{ij}} g(u_{ij})$ and is given by

$$c(u_{ij}) = \frac{-c_1(u_{ij}) - (\bar{u}_{i+1,j} - u_{ij})\left[\frac{u_{ij} - \bar{u}_{i+1,j} + \gamma^2(u_{ij} - \bar{u}_{i,j+1})}{c_1(u_{ij})}\right]}{c_1^2(u_{ij})}$$

$$- \frac{c_2(u_{ij}) - (u_{ij} - \bar{u}_{i-1,j})\left[\frac{u_{ij} - \bar{u}_{i-1,j}}{c_2(u_{ij})}\right]}{c_2^2(u_{ij})}$$

$$+\gamma^2 \left[\frac{-c_1(u_{ij}) - \gamma^2(\bar{u}_{i,j+1} - u_{ij})\left[\frac{u_{ij} - \bar{u}_{i+1,j} + \gamma^2(u_{ij} - \bar{u}_{i,j+1})}{c_1(u_{ij})}\right]}{c_1^2(u_{ij})} \right. \tag{4.5}$$

$$\left. - \frac{+c_3(u_{ij}) - \gamma^2(u_{ij} - \bar{u}_{i,j-1})\left[\frac{\gamma^2(u_{ij} - \bar{u}_{i,j-1})}{c_3(u_{ij})}\right]}{c_3^2(u_{ij})} \right]$$

with

$$\begin{aligned} c_1(u_{ij}) &= \sqrt{(\bar{u}_{i+1,j} - u_{ij})^2 + \gamma^2(\bar{u}_{i,j+1} - u_{ij})^2 + \beta_h} \\ c_2(u_{ij}) &= \sqrt{(u_{ij} - \bar{u}_{i-1,j})^2 + \gamma^2(\bar{u}_{i-1,j+1} - u_{i-1j})^2 + \beta_h} \\ c_1(u_{ij}) &= \sqrt{(\bar{u}_{i+1,j-1} - u_{ij-1})^2 + \gamma^2(u_{ij} - \bar{u}_{i,j-1})^2 + \beta_h} \end{aligned} \tag{4.6}$$

with appropriate modifications at points adjacent to the boundary. The algorithm for updating an approximation v_h to the solution of $N_h^{TV}(u_h) = z_h$ using Gauss-Seidel Newton with lexicographical ordering of the grid points is given in Algorithm 11.

Algorithm 11 Gauss-Seidel Newton

$$v_h \leftarrow GSNW(v_h, z_h, maxit, tol)$$

for $j = 1 : m$
  for $i = 1 : n$
    for $iter = 1 : maxit$
      $\bar{v}_h \leftarrow v_h$
      update $v_{i,j}$

$$v_{i,j} \leftarrow \frac{z_{ij} - \alpha_h \left(g(\bar{v}_{ij}) - c(\bar{v}_{ij})\bar{v}_{ij}\right)}{1 - \alpha_h c(\bar{v}_{ij})}$$

      if $|v_{i,j} - \bar{v}_{i,j}| < tol$ stop
    end
  end
end

Note that although only one step of Newton's method is usually used at each grid point we include the option of performing up to *maxit* steps, stopping if the value of $|v_{i,j} - \bar{v}_{i,j}|$ falls below some specified tolerance.

We found that this method on its own only converged if a small weighting parameter was applied to the Newton step i.e after the update of v_{ij} we set $v_{ij} = \omega v_{ij} + (1 - \omega)\bar{v}_{ij}$ where ω is typically $0.2 - 0.5$ and did not perform well as a smoother in a nonlinear multigrid method, performing more than one inner Newton step offered no real advantage. The poor performance of this method as a smoother forced us to look at other alternatives.

4.1.2 Local Linear Smoother

Note that $D(u)_{ij}$, $D(u)_{i-1,j}$ and $D(u)_{i,j-1}$ in (4.1) all contain u_{ij} terms. In our second smoother, for each grid point (i, j) as well as substituting current values of the approximation at non (i, j) points into (4.1) we also substitute current values of the approximation at (i, j) into the D terms; this gives us a linear equation in one variable to solve in order to update the approximation at (i, j). The algorithm for updating v_h using this method is given in Algorithm 12.

Note that because the linear equation used to update $v_{i,j}$ involves $\bar{v}_{i,j}$, at each grid point we perform up to *maxit* inner iterations. Typically we take $maxit = 2$ and $tol = 10^{-6}$. We found that this method was slowly convergent and could be speeded up by the use of nonlinear multigrid, however we found that using this smoother in a nonlinear multigrid method was less efficient than using another smoother based on a global linearization of $N_h^{TV}(u_h)$ which

Algorithm 12 Local Linear Smoother

$$v_h \leftarrow GSLL(v_h, z_h, maxit, tol)$$

for $j = 1 : m$

for $i = 1 : n$

for $iter = 1 : maxit$

$\bar{v}_h \leftarrow v_h$

update $v_{i,j}$ by solving the linear equation

$$v_{i,j} - \alpha_h((D(\bar{v})_{i,j}(\bar{v}_{i+1,j} - v_{i,j}) - D(\bar{v})_{i-1,j}(v_{i,j} - \bar{v}_{i-1,j})) +$$

$$\gamma^2 (D(\bar{v})_{i,j}(\bar{v}_{i,j+1} - v_{ij}) - D(\bar{v})_{i,j-1}(v_{i,j} - \bar{v}_{i,j-1}))) = z_{i,j}$$

if $|v_{i,j} - \bar{v}_{i,j}| < tol$ stop

end

end

end

I now outline.

4.1.3 Global Linear Smoother

Our third smoother is similar to the lagged diffusivity fixed point method of Vogel and Oman. In this method the system of nonlinear equations is linearized globally at each step by evaluating $D_{i,j}$ for all (i, j) using the current approximation, several steps of Gauss-Seidel relaxation are then applied to the resulting linear system. We found that while exactly solving the linear system, (or solving to some specified accuracy using conjugate gradient or linear multigrid) at each step seems to give a method which is not speeded up at all by nonlinear multigrid, applying just a few steps of Gauss-Seidel to the linear system results in a method that while obviously slower to converge than the fixed point method, can be used to good effect as a smoother in a nonlinear multigrid method. The algorithm is given in Algorithm 13.

We typically take $it = 3$ i.e we perform 3 inner Gauss-Seidel steps on each smoothing step.

4.1.4 Further Experiments

The two smoothers detailed above were investigated in [83], here I give details of some modifications considered more recently.

Algorithm 13 FPGS Smoother

$$v_h \leftarrow FPGS(v_h, z_h, it)$$

Evaluate $D(v_h)_{ij} = ((\delta_x^+ v_{i,j})^2 + (\gamma \delta_y^+ v_{i,j})^2 + \beta_h)^{-1/2}$ for all (i,j)
Perform Gauss-Seidel steps on linear system
$w_h = v_h$
for $iter = 1 : it$
    for $j = 1 : m$
        for $i = 1 : n$
            $\bar{w}_h \leftarrow w_h$

$$w_{i,j} \leftarrow \frac{z_{i,j} + \alpha_h(D(v_h)_{i,j}(\bar{w}_{i+1,j} + \gamma^2 \bar{w}_{i,j+1}) + D(v_h)_{i-1,j}\bar{w}_{i-1,j} + \gamma^2 D(v_h)_{i,j-1}\bar{w}_{i,j-1})}{1 + \alpha_h((1+\gamma^2)D(v_h)_{i,j} + D(v_h)_{i-1,j} + \gamma^2 D(v_h)_{i,j-1})}$$

        end
    end
end
$v_h \leftarrow w_h$

Red-Black Ordering

The first thing to note is that in the above algorithms a lexicographical ordering of the grid points is used I have also considered a red-black ordering of the grid points for both the local linear smoother and for the linear Gauss-Seidel steps within the global linear smoother, in some cases there is a slight advantage in using a red-black ordering, see results section for more details.

Jacobi Variants

For completeness I also give results for Jacobi variants of the two smoothers, although I have found no advantage in using these. Some numerical local fourier analysis (LFA) suggests that a weighting parameter of around 0.7 should give the best results in terms of smoothing, this seems to be confirmed by experiment.

Line Smoothers

Recently I have become aware of work by Frohn-Schauf, Henn and Witsch [48] on non-linear multigrid methods for the total-variation denoising problem. I do not make a direct comparison with their method here as they use a vertex-centered based discretization, which is different to the discretization scheme I am using, but I do take account of some of the techniques used. In [48] a pointwise smoother similar to the local linear smoother detailed

above is used, also considered is a line variant of this approach.

In the results section I give results for a Gauss-Seidel alternating line variant of the local linear smoother, this involves updating each line of the grid simultaneously by freezing all values of the approximation not on the line and also all values in the D terms before solving the resulting linear system. I also considered the use of a Gauss-Seidel alternating line relaxation method for solving the linear system on each step of my global linear smoother.

Over Relaxation

Also proposed in [48] is the use of over-relaxation of the local linear pointwise and line smoothers. I consider the use of over-relaxation for all the smoothers I have investigated, with the exception of the Jacobi version of the local linear smoother, where I am using an under-relaxation parameter.

Notation

In the following I will denote my fixed point type smoother by FP followed by a code denoting the type of relaxation used on the linear system GS for pointwise Gauss-Seidel with lexicographical ordering GSRB for pointwise Gauss-Seidel with red black ordering GSL for Gauss-Seidel alternating line relaxation and JA for pointwise Jacobi relaxation, shown in brackets is the number of inner relaxation steps used on each step (this corresponds to *it* in the above algorithm). I shall denote the local linear smoother by LL preceded by a code denoting the type of nonlinear relaxation used, in brackets will be the number of inner steps used for each grid point (this corresponds to *maxit* in the algorithm).

4.2 The Multigrid Method

For clarity I give the algorithm for the nonlinear multigrid method that we are using (Algorithm 14).

Note that we include the option to use the Krylov acceleration procedure outlined in §2.6.9, the effect of this is discussed in the results section. In most cases we take the noisy image z_h as the initial guess and set $tol = 10^{-4}\|z_h - N_h(z_h)\|_2$. We use a V-cycle method as we have found that we only gain a very small advantage in terms of convergence by using the more expensive W-cycle method. $FAS1^{TV}$ is defined recursively in Algorithm 15. Here N_{2h}^{TV} is the coarse grid analogue of N_h^{TV} i.e the operator resulting from discretization of the Euler-Lagrange equation on Ω^{2h}. The equation at a grid point of Ω^{2h} is just (4.1) with α_h replaced by $\alpha_{2h} = \alpha_h/2$ and β_h replaced by $\beta_{2h} = 4\beta_h$. The restriction and interpolation

Algorithm 14 Nonlinear Multigrid for TV Problem

Select Smoother, ν_1, ν_2 and initial guess v_h.

Set $k = 0$

While $\|z_h - N_h^{TV}(v_h)\|_2 < tol$

$k \leftarrow k + 1$

$v_h \leftarrow FAS1_h^{TV}(v_h, N_h^{TV}, z_h, \nu_1, \nu_2)$

If Krylov requested

Apply Krylov acceleration to find more optimal solution in space

$v_h + span[v_h^1 - v_h, ..., v_h^l - v_h]$

If $k < l$ set $v_h^k = v_h$ else set $v_h^{k \bmod l} = v_h$.

end

end

Algorithm 15 $FAS1^{TV}$

$$v^h \leftarrow FAS1_h^{TV}(v_h, N_h^{TV}, z_h, \nu_1, \nu_2)$$

1. If Ω^h = coarsest grid solve $N_h^{TV} u_h = z_h$ using the primal-dual Newton method and stop.
   Else For $l = 1$ to ν_1
   $v_h \leftarrow SMOOTH(v_h, N_h^{TV}, z_h)$
2. $v_{2h} \leftarrow I_h^{2h} v_h$
   $\bar{v}_{2h} \leftarrow v_{2h}$
   $z_{2h} \leftarrow I_h^{2h}(z_h - N_h^{TV} v_h) + N_{2h} v_{2h}$
3. $v_{2h} \leftarrow FAS1_{2h}^{TV}(v_{2h}, N_{2h}^{TV}, z_{2h}, \nu_1, \nu_2)$
4. Correct $v_h \leftarrow v_h + I_{2h}^h(v_{2h} - \bar{v}_{2h})$
5. For $l = 1$ to ν_1
   $v_h \leftarrow SMOOTH(v_h, N_h^{TV}, z_h)$

operators used are the cell-centered operators given by the stencils:

$$I_h^{2h} = \frac{1}{4}\begin{bmatrix} 1 & & 1 \\ & . & \\ 1 & & 1 \end{bmatrix}_h^{2h} \qquad I_{2h}^{h} = \frac{1}{16}\begin{bmatrix} 1 & 3 & 3 & 1 \\ 3 & 9 & 9 & 3 \\ 3 & 9 & 9 & 3 \\ 1 & 3 & 3 & 1 \end{bmatrix}_{2h}^{h}. \tag{4.7}$$

4.2.1 Complexity

The main cost of the multigrid method is the cost of the smoothing steps. Using the fixed-point type smoother, with 3 inner steps of Gauss-Seidel, costs approximately $90N$ flops per smoothing step, as the cost of evaluating $D_{i,j}$ (performed once per smoothing step) is 9 flops and the cost of one Gauss-Seidel step is 26 flops per grid point. An upper bound on the cost of one V-cycle is:

$$\lim_{l\to\infty}(\nu_1+\nu_2)90N\sum_{n=0}^{l}(1/4)^n = 90(\nu_1+\nu_2)N\left(\frac{4}{3}\right) = 120(\nu_1+\nu_2)N. \tag{4.8}$$

The extra costs associated with Krylov acceleration of a multigrid step are the evaluation of several residuals and inner products, the cost of which is approximately $100N$ and the direct solution of a small linear system the cost of which is negligible. Using for example $\nu_1 = \nu_2 = 5$, Krylov acceleration adds around 8% to the cost of a multigrid step.

4.3 Numerical Results

4.3.1 Comparison of Various Smoothers

In our first test we compare the performance of the FAS with the various smoothers outlined above for total variation denoising applied to 3 different 256×256 noisy images (Figure 4.1). The results for the first image, which is the blocky triangle image with $SNR = 3.4$ are given in Table 4.1, shown is the smoother used, the optimal number of pre and post correction smoothing steps (this is usually the smallest number of steps for which the multigrid method converges, although this is not always the case) the weighting factor ω used on the smoother, the number of steps required by the multigrid method to reduce the relative residual by a factor of 10^{-4} and the corresponding cpu time in seconds. Also shown is the number of steps and cpu time needed by each of the smoothers to converge on their own (without multigrid). Similar results are given for the Lenna image with $SNR = 3.2$ in Table 4.2 and for the fingers image with $SNR = 3.6$ in 4.3. In all cases we take the noisy image as the initial guess and take $\beta_h = 10^{-2}$, for the triangle and Lenna images we use a value of 30 for α_h, for the fingers image we take $\alpha_h = 35$.

Table 4.1: Comparison of Nonlinear Multigrid with various smoothers for Triangle Image

		FAS			Smoother Alone	
Smoother	ω	ν_1/ν_2	Steps	cpu(s)	Steps	cpu(s)
FPGS(3)	1	6/6	8	62.8	654	323.5
FPGS(3)	1.5	5/5	7	46.8	420	206.8
FPGSRB(3)	1	5/5	9	60.5	699	351.3
FPGSRB(3)	1.5	5/5	6	40.2	451	227.0
FPGSL(1)	1	5/5	9	148.9	489	509.9
FPGSL(1)	1.5	5/5	6	98.9	314	327.6
FPJA(3)	1	10/10	9	140.1	1895	1099.1
FPJA(3)	1.5	9/9	6	82.3	1247	726.4
GSLL(2)	1	10/10	8	171.7	1805	1486.1
GSLL(2)	1.5	10/10	8	171.5	1128	932.3
GSRBLL(2)	1	10/10	9	194.8	1913	1583.4
GSRBLL(2)	1.5	7/7	7	106.7	1252	1043.6
GSLLL(1)	1	4/4	7	111.2	563	701.4
GSLLL(1)	1.5	4/4	7	111.0	335	422.3
JALL(2)	0.7	30/30	7	275.0	4993	2583.9

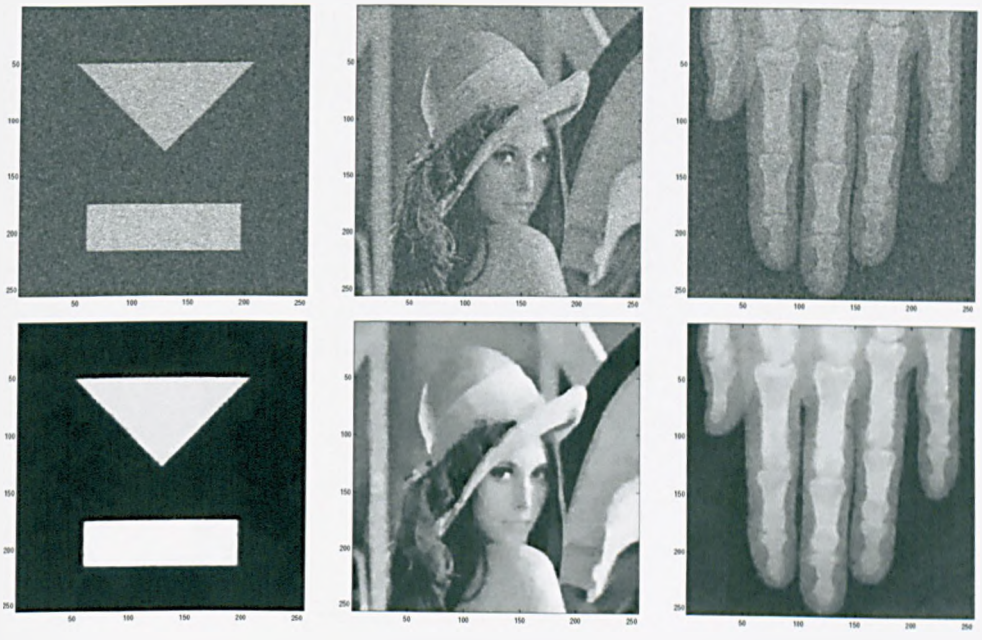
Table 4.2: Comparison of Nonlinear Multigrid with various smoothers for Lenna Image

		FAS			Smoother Alone	
Smoother	ω	ν_1/ν_2	Steps	cpu(s)	Steps	cpu(s)
FPGS(3)	1	5/5	14	91.6	596	301.4
FPGS(3)	1.5	4/4	12	63.2	382	192.6
FPGSRB(3)	1	5/5	14	92.3	632	319.1
FPGSRB(3)	1.5	5/5	9	59.3	403	205.2
FPGSL(1)	1	5/5	13	214.0	449	473.8
FPGSL(1)	1.5	4/4	11	145.0	283	298.1
FPJA(3)	1	15/15	12	268.2	1727	1021.6
FPJA(3)	1.5	12/12	9	167.5	1141	657.6
GSLL(2)	1	17/17	11	399.4	1633	1365.2
GSLL(2)	1.5	10/10	12	258.9	1009	843.3
GSRBLL(2)	1	17/17	11	402.6	1715	1438.3
GSRBLL(2)	1.5	11/11	11	263.3	1114	936.8
GSLLL(1)	1	5/5	11	218.3	529	662.6
GSLLL(1)	1.5	4/4	10	159.3	311	388.4
JALL(2)	0.7	*	*	*	4436	2282.2

Table 4.3: Comparison of Nonlinear Multigrid with various smoothers for Fingers Image

		FAS			Smoother Alone	
Smoother	ω	ν_1/ν_2	Steps	cpu(s)	Steps	cpu(s)
FPGS(3)	1	8/8	11	111.97	628	314.9
FPGS(3)	1.5	5/5	11	71.0	405	203.4
FPGSRB(3)	1	9/9	10	115.8	664	335.8
FPGSRB(3)	1.5	6/6	9	70.0	427	215.0
FPGSL(1)	1	7/7	11	250.0	475	502.9
FPGSL(1)	1.5	6/6	7	157.2	303	319.8
FPJA(3)	1	26/26	9	345.8	1805	1122.5
FPJA(3)	1.5	18/18	8	222.1	1193	741.4
GSLL(2)	1	24/24	10	512.1	1714	1436.1
GSLL(2)	1.5	18/18	8	306.5	1099	919.2
GSRBLL(2)	1	27/27	9	517.9	1803	1510.0
GSRBLL(2)	1.5	20/20	7	310.2	1164	975.4
GSLLL(1)	1	8/8	9	283.8	556	695.8
GSLLL(1)	1.5	6/6	8	189.3	347	432.5
JALL(2)	0.7	*	*	*	4631	2444.2

Figure 4.1: Noisy (top) and recovered (bottom) triangle, Lenna and fingers images



We see that overall the best performing smoothers are the GSLLL smoother and our FP type smoothers (with the exception of FPJA), the GSLLL smoother has slightly better convergence properties, however the FPGS and FPGSRB methods, which have almost as good convergence properties and are cheaper to implement, perform the best in terms of cpu, being over twice as fast as the GSLLL smoother. Overall their is a slight advantage in using a red-black ordering for the inner Gauss-Seidel steps. The pointwise local linear smoothers tend to need more smoothing steps within the FAS and are less robust, the Jacobi variant is particularly bad, convergence of the nonlinear multigrid method could not be achieved in two of the three cases tested. In all cases (except JALL) we gave results for the case where no over-relaxation was used and an over-relaxation parameter of 1.5 was used (1.5 was found to be the optimal value, larger values led to a break down in convergence). In all cases except the triangle image with GSLLL smoother, there was an advantage in using over-relaxation, typically giving methods which were one and a half times faster than the $\omega = 1$ case. In some cases less smoothing steps were needed per multigrid step when over-relaxation was used. The use of over-relaxation also speeded up the smoothers when used on their own. Finally we note that the multigrid method typically reduced the cost of the smoother alone by around $70 - 80\%$.

4.3.2 Krylov Acceleration

In the next experiment we investigate the effect of using the Krylov acceleration procedure of Washio and Oosterlee (see §2.6.9) to accelerate the FAS. We apply Krylov acceleration with 5 stored solutions, as this seems to give the best results. For each of the 3 images we run the FAS with Krylov acceleration with the FPGSRB smoother and for comparison the GSLLL smoother as well. Shown in Table 4.4 is the optimal number of pre and post correction steps (this is not always the same as when no acceleration is used) the number of steps required to reduce the residual by a factor of 10^{-4} the corresponding cpu and the reduction in cost as compared to the case where no Krylov acceleration is used. In all experiments we use z as the initial guess and α_h and β_h are as above.

Table 4.4: Krylov Accelerated Results

Image	Smoother	ω	ν_1/ν_2	Steps	cpu(s)	Cost Reduction
Triangle	FPGSRB(3)	1	4/4	9	50.3	17%
Triangle	FPGSRB(3)	1.5	4/4	6	33.4	17%
Triangle	GSLLL(1)	1	2/2	11	94.3	15%
Triangle	GSLLL(1)	1.5	4/4	6	96.4	14%
Lenna	FPGSRB(3)	1	4/4	12	67.3	27%
Lenna	FPGSRB(3)	1.5	4/4	9	50.2	15%
Lenna	GSLLL(1)	1	4/4	10	159.9	27%
Lenna	GSLLL(1)	1.5	4/4	8	128.3	19%
Fingers	FPGSRB(3)	1	3/3	21	90.5	22%
Fingers	FPGSRB(3)	1.5	3/3	13	56.0	20%
Fingers	GSLLL(1)	1	6/6	10	239.9	15%
Fingers	GSLLL(1)	1.5	4/4	10	160.9	15%

We see that in all cases the application of Krylov acceleration leads to a speed up in the multigrid method. In all cases their is still an advantage in using over-relaxation except for the triangle image with GSLLL smoother (this was also the case without acceleration), even though in some cases the slower $\omega = 1$ method is speeded up more by the application of Krylov acceleration. In all cases the FPGSRB smoother is faster than the GSLLL smoother, in two of the three cases the advantage has grown.

4.3.3 Comparison with Other Methods

In the next test I compare the nonlinear multigrid method, with the primal dual Newton method and the fixed point method (§3.6). Tests are carried out for various sizes of the

triangle and fingers images. Shown in Table 4.5 are the results of applying, the FAS method with FPGSRB smoother, on its own and with Krylov acceleration, the primal-dual Newton method and the fixed point method, on its own and with Krylov acceleration. In all cases we take $\beta_h = 10^{-2}$, for all sizes of the triangle we take $\alpha_h = 30$ and for all fingers images $\alpha_h = 35$. The noisy image is used as initial guess and the methods are stopped when the nonlinear residual has been reduced by a factor of 10^{-4}. The number of pre and post correction smoothing steps used in the nonlinear multigrid method are shown in the table, an over-relaxation parameter of 1.5 is used in all cases. In the primal dual Newton method an incomplete Cholesky preconditioned conjugate gradient method is used as the inner linear solver, with the stopping criteria as in (3.93), the Cholesky preconditioner is generated using MATLAB's cholinc program, with drop tolerance 0.1, as this gave the best results over the range of sizes. In the fixed point method we use a linear multigrid method with 2 pre and 2 post correction smoothing steps (see §5.2.2), the inner stopping criteria is a halving of the inner linear residual. If fixed point is applied without Krylov acceleration we use over-relaxation parameter 1.5 as this also speeds up the fixed point method, if however Krylov acceleration is used we use no over-relaxation, as we have found that this makes very little difference to the convergence and can in some cases increase the number of steps required, slightly.

We see that in all cases the nonlinear multigrid method outperforms the primal-dual Newton and Fixed point methods. If no Krylov acceleration is used, the nonlinear multigrid method is around 2.3 times faster than the fixed point method. The advantage of the nonlinear multigrid method over the primal-dual Newton method is larger for the larger images, in the case of the fingers image, for example, nonlinear multigrid method is approximately 1.5 times faster in the 256 case, 1.6 times faster in the 512 case and 2.3 times faster in the 1024 case. If Krylov acceleration is applied to both, the nonlinear multigrid method and the fixed point method, then the advantage of the nonlinear multigrid method is reduced, but it is still on average around 1.5 times faster.

In most cases, the primal-dual Newton method is faster than the fixed point method, but when Krylov acceleration is applied to the fixed point method it performs better. Note that if we used the same Cholesky preconditioned conjugate gradient method for the fixed point method as we did for the primal-dual Newton we would expect that the primal-dual Newton method would perform best.

4.3.4 Noisier Images

In the next test, I look at the performance of the nonlinear multigrid method for images with heavier noise levels, I compare the performance of the Krylov accelerated nonlinear multigrid method, with that of the primal-dual Newton and fixed point methods for the triangle image and the Lenna image (size 256×256) both with $SNR \approx 0.8$ (see Figure 4.2).

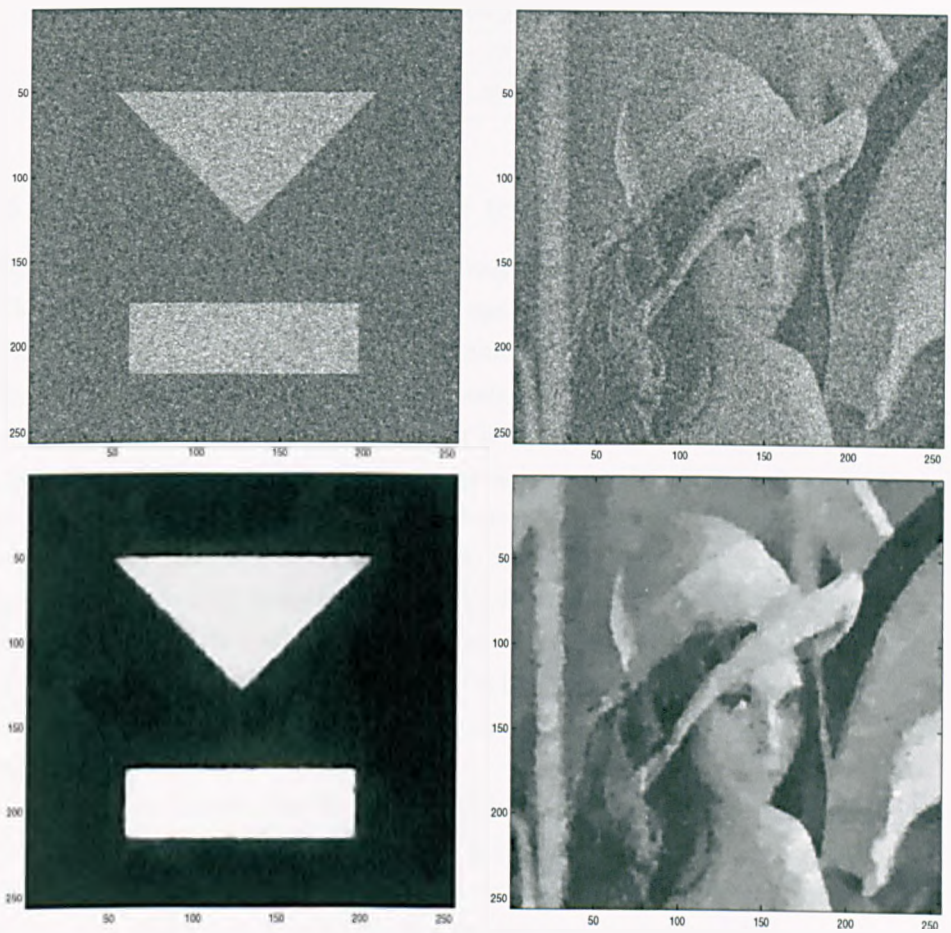
Table 4.5: Comparisons with Primal-Dual Newton and Fixed Point

Image	Size	FAS			K-FAS		
		ν_1/ν_2	Steps	cpu(s)	ν_1/ν_2	Steps	cpu(s)
fingers	256	6/6	9	70.0	3/3	13	56.0
fingers	512	6/6	10	332.9	3/3	13	233.4
fingers	1024	6/6	10	1401.4	3/3	13	994.4
triangle	256	5/5	6	40.2	4/4	6	33.4
triangle	512	5/5	6	160.7	5/5	5	140.9
triangle	1024	5/5	6	694.1	4/4	6	590.3

Image	Size	P-D NEW		FP		K-FP	
		Steps	cpu(s)	Steps	cpu(s)	Steps	cpu(s)
fingers	256	14	104.6	91	164.7	49	91.4
fingers	512	16	544.4	94	740.7	40	306.9
fingers	1024	18	3153.7	92	3424.9	41	1542.2
triangle	256	11	76.9	60	98.9	32	47.2
triangle	512	14	496.6	55	396.6	31	219.0
triangle	1024	14	1998.8	51	1602.9	32	951.1

For the nonlinear multigrid method I use 4 pre and 4 post correction smoothing steps, the

Figure 4.2: Noisy (top) and recovered (bottom) triangle and Lenna images with high noise levels



inner solvers in the primal-dual Newton and fixed point methods are as above. Since we have higher noise levels, we must use a larger value of α_h to remove the noise. For the triangle image we take $\alpha_h = 75$ and for the Lenna image $\alpha_h = 45$, in all cases $\beta_h = 10^{-2}$ and the stopping criteria and initial guess are as before.

For this more difficult problem, the cost of the nonlinear multigrid method increases, going up from 6 to 9 steps in the case of the triangle and from 9 to 12 in the case of the Lenna image, when compared to the less noisy cases seen earlier, however the nonlinear multigrid method still maintains its advantage over the fixed point and primal-dual Newton methods.

Table 4.6: Comparison of Nonlinear multigrid versus primal-dual Newton and Fixed Point for two noisier images

Image	K-FAS		P-D NEW		K-FP	
	steps	cpu(s)	steps	cpu(s)	steps	cpu(s)
Triangle	9	53.2	12	113.6	47	83.1
Lenna	12	70.1	15	139.5	54	131.7

4.3.5 Performance with Respect to β_h

Finally I comment on the performance of our multigrid method with respect to the parameter β_h. The larger β_h is the better the multigrid method performs. If β_h is reduced from 10^{-2} to 10^{-3} then we typically have to increase the number of smoothing steps to 20/20 to achieve convergence and the method costs approximately two and a half times as much as in the 10^{-2} case. The multigrid method reduces the cost of the smoother alone by approximately 70%. If β_h is reduced further to 10^{-4}, then we have to increase the number of smoothing steps to 50/50 to achieve convergence of the multigrid method, again there is around a 70% reduction in the cost of the smoother. For these smaller values of β_h I have found that the nonlinear multigrid method is still competitive with the fixed point method but is outperformed by the primal-dual Newton method (the optimal choice of preconditioner/linear solver in the fixed point and primal-dual Newton methods is likely to be different than in the 10^{-2} case). For even smaller values of β_h, I have been unable to achieve convergence of the nonlinear multigrid method.

We can achieve convergence for any β_h, with a fixed number of smoothing steps if a line search is used on the coarse grid correction as in §A.7 (in this case we replace the prolongation operator with $4(I_h^{2h})^T$ as this is required to guarantee a descent direction), however the actual reduction in the number of smoothing steps as compared to the smoother alone becomes very small as β_h decreases.

When β_h is very small the diffusion coefficients (the $D(u)_{ij}$ terms) can become very large in flat regions where $(\delta_x^+ u_{i,j})^2 + (\delta_y^+ u_{ij})^2 = 0$. The highly varying nature of these diffusion terms means that the nonlinear operator N_h^{TV} cannot be well approximated by the operator N_{2h}^{TV} resulting from a rediscretization on Ω^{2h} and this leads to the poor performance. Linear geometric multigrid methods used within the fixed point method suffer from a similar deterioration in performance. In the next chapter the fixed point method with Algebraic multigrid linear solver (proposed by Chang and Chern [34]) which is robust for small β_h is considered and a technique to improve its efficiency proposed.

Remark 4.3.1 *The choice $\beta_h = 10^{-2}$ used in the majority of the experiments in this section, is a commonly used choice which produces reasonable quality reconstructions and has been*

used in various papers e.g. [29] to test solvers for the TV problem.

4.4 Conclusion

I presented a nonlinear multigrid method, with a new smoother based on the fixed point method, but with just a few steps of Gauss-Seidel applied to the linear system on each step. The smoother was compared to various other smoothers, including one similar to that used in [48] and found to perform best. Provided β_h is not too small experiments suggest that the nonlinear multigrid method can outperform the primal-dual Newton and Fixed Point methods and that there is a particular advantage over the primal-dual Newton method for larger images. In all cases tested the nonlinear multigrid method could be speeded up by the application of a Krylov acceleration procedure. For smaller values of β_h an increase in the number of smoothing steps is required to achieve convergence and the nonlinear multigrid method loses its advantage over other methods (particularly the primal-dual method), if β_h is too small, the method will not converge at all without the use of a line search on the coarse grid correction.

Chapter 5

An Efficient Implementation of the Fixed Point Method with AMG Linear Solver

---

Useful Section References: §2.6, §2.7, §3.2, §3.5, §3.6, §A.6.3, §B.1, §B.2, §B.3
Main Reference Material: [9, 29, 34, 35, 78, 80, 85, 92, 99, 100], [93]-[97]

In this chapter I present a method which attempts to speed up the fixed point method, when algebraic multigrid is used as the inner linear solver, by recycling the AMG setup data. I start with a brief review of the fixed point method (§5.1) and the linear solvers used within it (§5.2). In §5.3 I introduce the recycling idea and present the results of some preliminary tests which motivate the final algorithm presented at the end of the section. Finally some numerical results showing the effectiveness of the new method in speeding up the fixed point with AMG method are presented in §5.4 and conclusions drawn in §5.5.

5.1 The Fixed Point Method

Recall from Chapter 3 that step $k+1$ of the fixed point (FP) method requires the solution of the linear system

$$A(\mathbf{u}_h^k)\mathbf{u}_h^{k+1} = \mathbf{z}_h. \tag{5.1}$$

where

$$A(\mathbf{u}_h^k) = I + \alpha_h BE(\mathbf{u}_h^k)^{-1}B^T. \tag{5.2}$$

and B and E are as defined in §3.5.1. The matrix A is symmetric positive definite and is sparse. It has an $m \times m$ block tridiagonal structure with $n \times n$ tridiagonal diagonal blocks and $n \times n$ diagonal off-diagonal blocks. The number of operations required by a direct solver is $O(N^2)$, where N is the number of pixels and so as the size of the problem increases, iterative solvers must be used. The general fixed point method is given in Algorithm 16.

Algorithm 16 Fixed Point

Input Initial Guess $\mathbf{u}_h^0$, Set $k = 0$
While $\|\mathbf{z}_h - A(\mathbf{u}_h^k)\mathbf{u}_h^k\|_2 > tol_1$
    Set $\mathbf{w} = \mathbf{u}_h^k$
    While $\|\mathbf{z}_h - A(\mathbf{u}_h^k)\mathbf{w}\|_2 > tol_2\|\mathbf{z}_h - A(\mathbf{u}_h^k)\mathbf{u}_h^k\|_2$
        Update $\mathbf{w}$ with one step of some iterative method for solving the linear equation $A(\mathbf{u}_h^k)\mathbf{w}^* = \mathbf{z}_h$.
    end
    Set $\mathbf{u}_h^{k+1} = \mathbf{w}$
    $k \leftarrow k + 1$
end

In most cases we take $\mathbf{u}_h^0 = \mathbf{z}_h$ and $tol_1 = 10^{-4}\|\mathbf{z}_h - A(\mathbf{z}_h^k)\mathbf{z}_h^k\|_2$. The value of tol_2 determines by how much we require the linear residual to be reduced by before moving to the next outer fixed point step. For clarity I focus here on the case that $tol_2 = 0.1$ as suggested in [29], using a smaller value of tol_2 results in, at best, a small reduction in the total number of fixed point steps, the advantage of which is outweighed by the extra work needed to solve the linear system to a higher accuracy, we have in some cases found that taking an even larger value of tol_2, say 0.5 gives slightly better results.

In the following we also accelerate the fixed point method using the Krylov acceleration procedure outlined in §2.6.9 for use with nonlinear multigrid (the process is the same with the multigrid step replaced by a fixed point step), an idea which was proposed by Chang and Chern in [34]. We carry out Krylov acceleration after each fixed point step using 5 stored previous iterates.

5.2 Linear Solvers

In this section I review the various linear iterative solvers which have been proposed in the literature for use in the fixed point method.

5.2.1 PCG with Incomplete Cholesky Preconditioner

Given that $A(\mathbf{u}_h^k)$ is symmetric positive definite, the preconditioned conjugate gradient method (PCG) can be used as an iterative solver. In [29] the incomplete Cholesky pre-

conditioner is used as preconditioner within the PCG method. In the following I generate the incomplete cholesky preconditioner using MATLAB's CHOLINC program with the drop tolerance option (see §A.6.3), obviously the larger the drop tolerance, the less it costs to generate and invert the preconditioner, but the less accurate the preconditioner is.

5.2.2 Geometric Multigrid

In [96] Vogel proposes a geometric multigrid method to solve the linear system within the fixed point method. The method I outline below differs slightly from Vogel's method because I am using a slightly different discretization scheme. Reverting, for now, to a grid function notation the linear operator L_h on step $k+1$ of the fixed point method at points not adjacent to the boundary is represented by the stencil

$$\begin{bmatrix} 0 & -\alpha_h \gamma D^h_{ij} & 0 \\ -\alpha_h D^h_{i-1,j} & 1+\alpha_h \Sigma_{ij} & -\alpha_h D^h_{ij} \\ 0 & -\alpha_h \gamma D^h_{i,j-1} & 0 \end{bmatrix} \tag{5.3}$$

where $\Sigma_{i,j} = (1+\gamma)D^h_{ij} + D^h_{i-1,j} + D^h_{i,j-1}$. The grid function D^h is dependant on u^k_h and is given at grid point (i,j) by

$$\left((\delta^+_x u^k_{i,j})^2 + (\gamma \delta^+_y u^k_{ij})^2 + \beta_h\right)^{-1/2}. \tag{5.4}$$

On a standard coarsened grid Ω^{2h} the coarse grid operator L_{2h} has stencil

$$\begin{bmatrix} 0 & -\alpha_{2h} \gamma D^{2h}_{ij} & 0 \\ -\alpha_h D^{2h}_{i-1,j} & 1+\alpha_{2h} \Sigma_{ij} & -\alpha_{2h} D^{2h}_{ij} \\ 0 & -\alpha_{2h} \gamma D^{2h}_{i,j-1} & 0 \end{bmatrix} \tag{5.5}$$

where $\alpha_{2h} = \alpha_h/2$ and $D^{2h} = I^{2h}_h D^h$.

The linear equation $L_h w_h = z_h$ is solved using the standard multigrid V-cycle method (Algorithm 3 with $\mu = 1$) with 2 pre-correction smoothing steps of red-black Gauss-Seidel and 2 post correction smoothing steps of black-red Gauss-Seidel. We use the cell-centered restriction operator given by the stencil

$$I^{2h}_h = \frac{1}{4} \begin{bmatrix} 1 & & 1 \\ & \cdot & \\ 1 & & 1 \end{bmatrix}^{2h}_h \tag{5.6}$$

and the interpolation operator is taken to be $4(I^{2h}_h)^T$. This method can be applied as a solver on its own, but to improve performance Vogel uses preconditioned conjugate gradient to accelerate it, hence the need for symmetry in the smoothing steps and the choice of transfer operators. The number of operations required to perform a single V-cycle will be $O(N)$ and experiments suggest, there is only a very small increase in the number conjugate gradient steps required as the size of the problem grows.

5.2.3 Algebraic Multigrid

The geometric multigrid method may perform reasonably well if β_h is not too small, say 10^{-2}, however for smaller values of β_h its performance tends to deteriorate quite rapidly. A more robust, but more expensive, option is to use a black box algebraic multigrid method (§2.7), this approach is proposed in [34] by Chang and Chern. Chang and Chern use an algebraic multigrid method which uses the interpolation operator of Chang, Wong and Fu detailed in §2.7.6, we have found that using direct interpolation gives similar results and this method is used for ease of implementation.

Cost of the AMG method

It is important to note that at this stage we are not confident that our implementation of the AMG setup phase in MATLAB is as efficient as it could be. Exactly how the AMG setup phase is implemented can have a significant effect on the cost in terms of cpu time. A naive implementation of the C/F-splitting algorithm using MATLAB's intersect and setdiff functions is very expensive in terms of cputime. We have improved our implementation several times by using various techniques to try and get around the most expensive costs, however in appendix B we present an estimate for the (equivalent) cost in flops of our current implementation based on various assumptions which reveal a potential $O(N^2)$ cost associated with performing the C/F splitting (Algorithm 9), it is our aim in the future to improve upon this.

The current cost of a setup phase in cputime is around 180 times the cost of a V-cycle for $N = 256^2$. Results presented in [92] using the RAMG05 code suggest that an assumption that a setup phase costs around 4 times the cost of a V-cycle may give a guide as to what can be achieved with an optimal implementation. Throughout the discussion below we give a guide as to what sort of reduction in cost can be expected with our recycling idea based on the relative cost of the setup to the V-cycle and the recycle (essentially the Galerkin step in the setup phase).

5.2.4 How Linear Solvers Perform

A detailed comparison of the various linear solvers used within the fixed point method is difficult and as far as I am aware has not been carried out in the literature. It is not my aim here to do such a comparison or to advocate one particular approach over the others, my work has focused on the particular case of fixed point with algebraic multigrid and a possible way to speed up this particular method. Any comparison between the various linear solvers in terms of cpu time would be unfair given that we our not entirely happy with our current implementation of the AMG method.

Although I do not attempt a detailed comparison of linear solvers, I do in the following

give a few results from my own experience, which demonstrate the performance of the various methods with respect to the parameter β_h.

Shown in Table 5.1 is the average number of conjugate gradient steps needed per fixed point step to reduce the linear residual by a factor of 0.1 for various values of the parameter β_h and various preconditioners (where a * is shown the number of steps required was excessive and no results are given). Results are run on a 256×256 noisy image and $\alpha_h = 30$ in all cases. The incomplete cholesky preconditioner with various values of *droptol* is considered, so is the geometric multigrid method. The incomplete Cholesky factor with *droptol* $= 0.1$ has roughly the same sparsity pattern as the lower triangular part of the matrix A or around 1% of the entries of the full Cholesky factor, Cholinc, with *droptol* $= 10^{-3}$ has between 2 and 4% of the entries of the full Cholesky factor depending on what stage the fixed point method is at and the value of β_h. I have found that the larger β_h is the larger the number of entries in the incomplete Cholesky factor. If *droptol* $= 10^{-6}$ the incomplete Cholesky factor has between 3-15% of the entries of the full Cholesky factor, again larger β_h leads to more entries.

Table 5.1: Average number of conjugate gradient steps needed per fixed point step for various preconditioners and values of β_h

β_h	10^{-2}	10^{-4}	10^{-8}	10^{-12}
Preconditioner				
Cholinc(0.1)	7.8	16.3	263.5	*
Cholinc(10^{-3})	1	1.3	11.5	88.3
Cholinc(10^{-6})	1	1	1	1.8
GMG(2/2)	2.1	3.7	18.6	75.4

The optimal value of *droptol* in the incomplete Cholesky factorisation will depend on the size of the problem and on the value of β_h but in general it will be smaller for smaller β_h, a strategy of taking *droptol* $= \sqrt{\beta_h}$ seems to give good results.

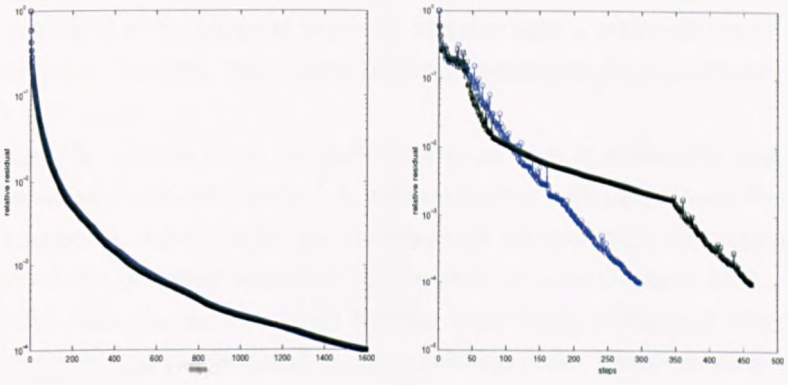
In Table 5.2 I give the average number of V-cycles needed per fixed point step, for the same image, using the AMG method and the GMG method (without preconditioning) with the same values of β_h as before, in both cases 2 pre and 2 post-correction smoothing steps were used.

I have found that geometric multigrid performs reasonably well for relatively large values of β_h (10^{-2} or greater), as β_h decreases, the performance deteriorates quite badly, this effect can be somewhat reduced by using GMG as a preconditioner, within the conjugate gradient method (Table 5.1). AMG tends to be much more robust with respect to small values of β_h than GMG, with only a small increase in the number of V-cycles required.

Table 5.2: Average number of V-cycles needed per fixed point step for GMG and AMG for various values of β_h

β_h	10^{-2}	10^{-4}	10^{-8}	10^{-12}
GMG(2/2)	3.1	9.4	202.9	*
AMG(2/2)	1	1.1	1.7	2.4

Figure 5.1: Convergence History of fixed point with AMG (circles) and pcg with Choline(10^{-6}) (squares) with no Krylov acceleration (left) and Krylov acceleration (right)



As well as being robust with respect to small β_h we have also found that AMG may work better in combination with Krylov acceleration than PCG with incomplete Cholesky preconditioner. In Figure 5.1 the convergence history of the fixed point method with preconditioned conjugate gradient using incomplete Cholesky with *droptol* $= 10^{-6}$ and AMG is shown for the case where no Krylov acceleration is used and the case where Krylov acceleration with 5 stored iterates is used. The results are for a 256×256 image and $\beta_h = 10^{-12}$. We see that the convergence histories are almost identical when no Krylov acceleration is used, but when Krylov acceleration is applied, the fixed point with AMG performs significantly better. This result is not observed in all cases and is generally more dramatic for smaller β_h.

As stated earlier these results simply reflect my own experience and are not meant to be an exhaustive study of linear solvers within the fixed point method.

In the rest of this chapter I focus purely on the fixed point with AMG method and propose a way to reduce the cost by performing less setup phases.

5.3 Recycling of AMG Setup Data Within the Fixed Point Method

The improved convergence properties of the algebraic multigrid method over the geometric multigrid method, comes at the cost of the need to perform an AMG setup phase (§2.7.2), in this section I outline a method to try and reduce the number of setups required over the whole fixed point process.

5.3.1 Motivation

If algebraic multigrid is used as the linear solver in the fixed point method, then on each step an algebraic multigrid setup phase is required, however only a relatively small reduction in the linear residual is required, which once the setup phase has been performed will require typically 1 or 2 V-cycles.

Observe that the matrix $A(\mathbf{u}_h^k)$ at each fixed point step is symmetric positive definite (SPD). The variational principle (see §2.7.3) guarantees that a Galerkin based V-cycle method in which restriction is chosen to be the transpose of interpolation will converge for SPD matrices provided the smoother converges (convergence is not necessarily fast). Furthermore at all fixed point steps the matrix $A(\mathbf{u}_h^k)$ has the same block tridiagonal structure. Based on these observations and the fact that accurate solution of the linear equation at each fixed point step is not necessary we propose that it may be possible to recycle the AMG setup data from an earlier fixed point step for use at later fixed point steps thus reducing the computational cost of the method.

By recycling of setup data we mean the following: Given a system $A\mathbf{w} = \mathbf{z}$, to solve and stored interpolation operators $\hat{I}_{(2)}^{(1)}, ..., \hat{I}_{(L)}^{(L-1)}$ generated from the entries of a matrix $\hat{A}$ which is similar to A (in our case $A = A(\mathbf{u}_h^k)$ and $\hat{A} = A(\mathbf{u}_h^l)$, $l < k$, is the matrix from a previous fixed point step) instead of generating a new C/F-splitting and interpolation operators based on the entries of A, we use the stored interpolation operators and generate the coarse grid matrices $A^{(2)}, ..., A^{(L)}$ using the Galerkin principle i.e for $p = 2, .., L$ we evaluate.

$$A^{(p)} = (\hat{I}_{(p)}^{(p-1)})^T A^{(p-1)} \hat{I}_{(p)}^{(p-1)}. \tag{5.7}$$

These coarse grid matrices together with the stored interpolation operators are then used in the V-cycle.

5.3.2 Preliminary Experiments

Below I present some preliminary experiments which motivate the final algorithm presented later. Test 1 looks at the effect on the overall cost of the fixed point method of recycling setup data after every q steps, while test 2 examines the effect recycling has on the efficiency

of the AMG solver.

Test 1. Table 5.3 shows convergence information for the fixed point method with AMG applied to the linear system at each step, we use standard algebraic coarsening with direct interpolation and point Gauss-Seidel as smoother with 2 pre and 2 post correction smoothing steps within a multigrid V-cycle. The AMG setup i.e generation of a coarse-fine splitting and interpolation and restriction operators from matrix entries, is carried out after every q fixed point steps. If $q \neq 1$ then the most recent setup data is used for the linear multigrid solver at a fixed point step. Enough V-cycles are run to reduce the inner residual by a factor 0.1. The table shows the number of fixed point steps required to reduce the outer (nonlinear residual) by a factor of 10^{-4} and the total cputime, for various values of q. Also shown is the total number of setups, V-cycles and recycles with corresponding cputime in seconds. The experiments are run on a 256×256 noisy image with SNR=3.6. I take $\alpha_h = 35$ and $\beta_h = 10^{-4}$.

Table 5.3: Fixed point with AMG data for various frequencies q of setup regeneration

q	FP Steps	cpu	Setups		V-cycles		Recycles	
			Number	Total Cpu	Number	Total cpu	Number	Total cpu
1	62	13716	62	13561	72	88	*	*
2	56	6122	28	5976	64	77	28	10
3	56	4215	19	4064	66	79	37	13
4	57	3410	15	3244	75	91	42	15
5	64	3023	13	2824	93	113	51	18
10	55	1552	6	1301	138	175	49	17
15	61	1513	5	1140	225	289	55	20
25	69	1237	3	655	364	476	66	24
50	71	1439	2	453	673	884	69	27

We make the following observations. Firstly as q increases the setup cost goes down but in general we require an increasing number of V-cycles because the recycled interpolation operators become less accurate, however it appears we can achieve some reduction in the number of setups for free. Indeed in the case of $q = 2$ and $q = 3$, we have used fewer V-cycles than in the $q = 1$ case (standard AMG). This is due to the fact that the number of outer fixed point steps has reduced slightly but the ratio of V-cycles to fixed point steps has also not increased. Even in the $q = 4$ case there is only a 4% rise in the number of V-cycles. For $q \geq 5$ there is approximately a linear relationship between the increase in the number of V-cycles per fixed point step (as compared to the $q = 1$ case) and q. It is of interest to

investigate the optimal frequency q_{opt}. To do this, we model the approximate increase in V-cycles by $0.15q + 0.7$ and assume that on average 1.2 V-cycles are performed on each fixed point step in the original (no-recycling) method. Denoting the cost of a setup phase by C_S, the cost of a V-cycle by C_V and the cost of a recycle by C_R we have that the average cost of a fixed point step when a setup is performed every q steps is

$$\frac{C_S}{q} + \left(1 - \frac{1}{q}\right) C_R + 1.2C_V(0.15q + 0.7). \tag{5.8}$$

Minimizing this quantity we get that the optimal q is

$$q_{opt} = \frac{10}{3\sqrt{2}} \sqrt{\frac{C_S - C_R}{C_V}}. \tag{5.9}$$

As mentioned earlier we aim to give a guide as to the sort of reduction in the cost of the fixed point method that can be achieved based on the relative costs of a setup, a recycle and a V-cycle. If we take $C_S = \rho_1 C_V$ and $C_R = \rho_2 C_V$ where ρ_1 is the ratio of setup cost to V-cycle cost and ρ_2 is the ratio of recycle cost to V-cycle cost then (5.9) becomes

$$q_{opt} = \frac{10}{3\sqrt{2}} \sqrt{\rho_1 - \rho_2}. \tag{5.10}$$

The factor by which the average cost of a fixed point step is decreased by is

$$\frac{\frac{C_S}{q_{opt}} + \left(1 - \frac{1}{q_{opt}}\right) C_R + 1.2C_V(0.15q_{opt} + 0.7)}{C_S + 1.2C_V}, \tag{5.11}$$

which is

$$\frac{\frac{3\sqrt{2}}{5}\sqrt{\rho_1 - \rho_2} + \rho_2 + 0.84}{\rho_1 + 1.2}. \tag{5.12}$$

We can then make a prediction of the best choice of q and the speed up in the fixed point method based on ρ_1 and ρ_2. The user can measure ρ_1 and ρ_2 in cpu time by running one AMG, or base them on complexity analysis. In our case the ρ_1 as measured in cpu is approximately 180 and ρ_2 is approximately 0.3, giving $q_{opt} \approx 31$ with a reduction in cost of around 93%. If we were to take $\rho_1 = 4$ we would have $q_{opt} \approx 5$ (on the cusp of where the above assumptions are valid) and expect a reduction in cost of around 45%.

This analysis is for the $\beta = 10^{-4}$ case, I note here that for smaller β the optimal value of q will in general be smaller. Although the AMG method itself is fairly robust with respect to changes in β, once you start recycling the performance of the multigrid solver deteriorates more rapidly for smaller β.

Test 2. To get a better idea of how effective the recycling of setups is, in Table 5.4, some information on the efficiency of the linear multigrid solver for the case where $q = 10$ is given. Data are given for the first and last fixed point steps at which a particular setup is used. Shown is the number of multigrid cycles required to reduce the inner (linear) residual by a factor of 10^{-4} (rather than 0.1 as we are interested in the efficiency of the inner solver rather

than the speed of the overall fixed point method) and the amount by which the residual is reduced on the first step (if no recycling is used, one V-cycle is usually enough to reduce the residual by a tenth, which is what we ordinarily require).

Table 5.4: Efficiency of AMG recycling for $q = 10$

FP step	number of inner multigrid steps	convergence factor on first step
1	2	5.3×10^{-3}
10	116	1.9×10^{-1}
11	15	4.1×10^{-2}
20	38	1.2×10^{-1}
21	16	7.4×10^{-2}
30	27	9.1×10^{-2}
31	13	6.5×10^{-2}
40	12	6.4×10^{-2}

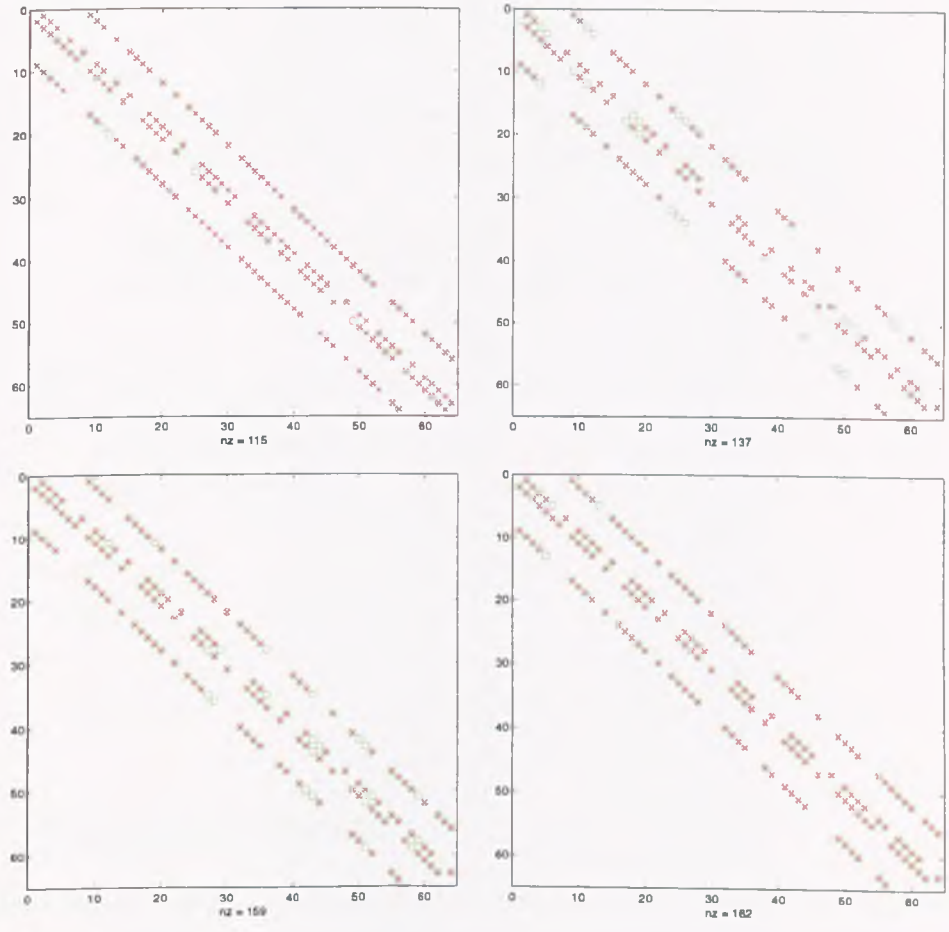
From Table 5.4 we observe that on early fixed point steps the multigrid method is significantly worse on the last step at which the recycled data is used compared to the first step (where the data was generated), but at later fixed point steps the performance on the first and last fixed point steps at which a particular setup data is used is very similar, the results shown are for the case $q = 10$, but similar results have been observed for other values of q. Figure 5.2 shows the matrix entries (the matrix at the finest level) corresponding to strong connections at various fixed point steps along with the entries which corresponded to strong connections 10 fixed point steps ago, for ease of presentation the image is only 8×8 but we have observed similar results for larger images. On step 11 there are a large number of points which initially corresponded to strong connections but are now weak connections. At steps 21 and 31 there is less change in the connectivity pattern (compared to 10 steps ago) than there was at step 11 and the majority of points that have changed have gone from being weak to being strong. By step 41 there is almost no change in the pattern of strong connections compared to step 31.

The above test results suggest that more setup phases are required at early fixed point steps and less later on i.e the recycling should be more adaptive.

5.3.3 The New Method

Instead of arbitrarily carrying out a new setup phase every q FP steps we base the decision on whether new setup data is required on the convergence history at the previous FP step. If the multigrid method takes more than ss V-cycles to reduce the linear residual by a factor of 0.1 on a particular FP step we generate new setup data at the next step. This allows for

Figure 5.2: Strong connections (circles) and strong connections 10 steps ago (crosses) at steps 11, 21, 31 and 41 of the fixed point method



more setup phases on early fixed point steps when they are needed and less at later fixed point steps. This new method should strike a balance between using as few AMG setups as possible and not causing a dramatic increase in the overall number of V-cycles required. The algorithm for the new method, which from now on I shall refer to as fixed point with AMG-R, is given in Algorithm 17. Here the notation $AMG1$ is as defined in (2.249).

Algorithm 17 Fixed Point with AMG-R

Set $\mathbf{u}_h^0 = \mathbf{z}_h$, $ms^0 = ss + 1$, $k = 0$
While $\|\mathbf{z}_h - A(\mathbf{u}_h^k)\mathbf{u}_h^k\|_2 > 10^{-4}\|\mathbf{z}_h - A(\mathbf{u}_h^0)\mathbf{u}_h^0\|_2$
    Evaluate $A^{(1)} = A(\mathbf{u}_h^k)$.
    Set $\mathbf{z}^{(1)} = \mathbf{z}_h$, $\mathbf{v}^{(1)} = \mathbf{u}_h^k$, $\mathbf{r}_0^{(1)} = \mathbf{z}^{(1)} - A^{(1)}\mathbf{v}^{(1)}$.
    If $ms^k > ss$
        Perform AMG setup and generate $A^{(2)}, ..., A^{(L)}$.
        Store the Interpolation operators from the AMG setup.
    else
        Generate $A^{(2)}, ..., A^{(L)}$ using stored
        interpolation operators from most recent setup.
    end
    Set $ms^{k+1} = 0$
    While $\|\mathbf{r}^{(1)}\|_2 > 0.1\|\mathbf{r}_0^{(1)}\|_2$
        $\mathbf{v}^{(1)} \leftarrow AMG1^{(1)}(\mathbf{v}^{(1)}, \mathbf{z}^{(1)}, 2, 2)$
        $ms^{k+1} \leftarrow ms^{k+1} + 1$
    end
    Set $\mathbf{u}_h^{k+1} = \mathbf{v}^{(1)}$
    $k \leftarrow k + 1$
    end
end

5.4 Numerical Results

In the following I compare the fixed point method with AMG-R, with the standard fixed point with AMG method. I give results for 2 256×256 images (Lenna and Fingers) each with $SNR \approx 3.5$. In each case 4 different values of β_h are tested, in the case of Lenna $\alpha_h = 30$ in the case of the Fingers $\alpha_h = 35$. AMG-R in each case is run with two different values of ss, $ss = 3$ and $ss = 10$. Shown in Tables 5.5-5.6 are the number of fixed point steps required for convergence, the total cpu time in seconds and also the total number of setups and V-cylces required, with corresponding cpu times.

I have presented results for the cases $ss = 3$ and $ss = 10$. I give results for the case

Table 5.5: Comparison of Fixed Point with AMG-R against Fixed Point with AMG for Lenna image

β_h	Linear Solver	AMG	AMG-R(3)	AMG-R(10)
10^{-2}	FP Setps	31	31	31
	Total cpu	7499	570	398
	Setups: No/cpu	31/7421	2/475	1/246
	V-cycles: No/cpu	31/42	40/51	80/106
	Recycles:No/cpu	*	29/11	30/13
10^{-4}	FP Setps	66	56	89
	Total cpu	14912	899	1005
	Setups: No/cpu	66/14737	3/690	2/464
	V-cycles: No/cpu	83/104	106/130	339/414
	Recycles:No/cpu	*	53/19	87/31
10^{-8}	FP Setps	172	146	148
	Total cpu	37744	2123	1937
	Setups: No/cpu	172/37165	7/1533	5/1104
	V-cycles: No/cpu	322/393	316/383	514/623
	Recycles:No/cpu	*	139/49	143/50
10^{-12}	FP Setps	310	291	272
	Total cpu	67878	3602	3340
	Setups: No/cpu	310/66747	11/2381	8/1739
	V-cycles: No/cpu	652/795	666/809	1001/1214
	Recycles:No/cpu	*	280/97	264/92

Table 5.6: Comparison of Fixed Point with AMG-R against Fixed Point with AMG for Fingers image

β_h	Linear Solver	AMG	AMG-R(3)	AMG-R(10)
10^{-2}	FP Setps	33	31	31
	Total cpu	7979	549	396
	Setups: No/cpu	33/7895	2/454	1/239
	V-cycles: No/cpu	33/45	42/52	86/112
	Recycles:No/cpu	*	29/11	30/12
10^{-4}	FP Setps	62	56	76
	Total cpu	14094	867	846
	Setups: No/cpu	62/13923	3/662	2/449
	V-cycles: No/cpu	72/99	108/128	246/293
	Recycles:No/cpu	*	53/18	74/26
10^{-8}	FP Setps	154	173	174
	Total cpu	34520	2351	1994
	Setups: No/cpu	154/33967	8/1677	5/1061
	V-cycles: No/cpu	279/372	372/437	589/692
	Recycles:No/cpu	*	165/56	169/57
10^{-12}	FP Setps	301	310	297
	Total cpu	67427	3646	3095
	Setups: No/cpu	301/66195	11/2342	7/1485
	V-cycles: No/cpu	655/875	719/868	990/1191
	Recycles:No/cpu	*	299/103	290/100

$ss = 10$ because I have found that generally this gives the best results in terms of cpu time. Given that I have concerns over the implementation of the AMG setup phase I also include the case $ss = 3$ as a method which should limit the increase in the number of V-cycles to be as small as possible. A value $ss = 3$ is chosen because we have observed that this is the maximum number of V-cycles required per fixed point step in the standard fixed point with AMG method. We see that in both cases, there is a significant decrease in the number of setup phases required. In the case of $ss = 3$ there is on average a 95% reduction in the number of setups compared to the fixed point with AMG method with on average only a 22% increase in the number of V-cycles. In the case of $ss = 10$ there is on average a 97% decrease in the number of setups, with an average 142% increase in the number of V-cycles. Note that in the preliminary tests where we performed a setup every q steps regardless, a 96% reduction in the number of setups ($q = 25$) led to a 406% increase in the number of V-cycles. In both cases the largest reduction in setups occurs for the smallest value of β_h, 10^{-12}, the increase in the number of V-cycles is in general also smaller for smaller β_h.

Based on these timings, the cost of the fixed point with AMG method in cpu time is reduced by around 95% by recycling setup data. Taking the average reduction in setups and increase in V-cycles from the $ss = 3$ case and assuming that 2 V-cycles are used per fixed point step with no-recycling the expected reduction in cost for any ρ_1 and ρ_2 (as defined earlier) would be

$$\frac{0.05\rho_1 + 0.95\rho_2 + 2.44}{\rho_1 + 2}. \tag{5.13}$$

If we take $\rho_2 = 0.3$, then we can expect some speed up in the fixed point method provided $\rho_1 > 0.7$ i.e the cost of a setup is a least 0.7 times the cost of a V-cycle. Taking $\rho_1 = 4$ we would expect to at least halve the overall cost of the fixed point method.

For information I also show in Table 5.7 on which fixed point steps AMG setups are actually performed, results are shown for the fingers image only. As expected the majority

Table 5.7: Fixed Points steps on which AMG setups are performed for fixed point with AMG-R(3) and AMG-R(10) run on the fingers image, with various values of β_h

	AMG-R(3)	AMR-R(10)
β_h	AMG Setup on steps:	AMG Setup on steps:
10^{-2}	1,8	1
10^{-4}	1,5,14	1,8
10^{-8}	1,5,14,23,32,41,57,109	1,8,23,38,76
10^{-12}	1,5,14,30,40,51,63,76,98,130,275	1,8,29,45,61,88,139

of setups occur early on in the fixed point process.

Finally I present two more experiments to test the robustness of the method. In the first test I compare fixed point with AMG against fixed point with AMG-R for a noisier version of the Lenna image seen earlier, in this case the SNR=0.8 and the value of α_h is increased to 45. In the second test I run comparisons for a larger version of the Fingers Image. The size of the image is 512 × 512 and a similar amount of noise is present as in the 256 × 256 case, α_h is again chosen to be 35. In both cases I compare the methods for $\beta_h = 10^{-4}$ and $\beta_h = 10^{-8}$ only. Results for the noisy Lenna Image are given in Table 5.8 and results for the larger fingers image are given in Table 5.9.

Table 5.8: Comparison of Fixed Point with AMG-R against Fixed Point with AMG for Noisier Lenna image

β_h	Linear Solver	AMG	AMG-R(3)	AMG-R(10)
10^{-4}	FP Setps	69	74	98
	Total cpu	15091	1148	1097
	Setups: No/cpu	69/14894	4/910	2/476
	V-cycles: No/cpu	97/121	134/160	401/470
	Recycles:No/cpu	*	70/25	96/35
10^{-8}	FP Setps	199	187	167
	Total cpu	42813	4210	2003
	Setups: No/cpu	199/42098	16/3460	5/1128
	V-cycles: No/cpu	399/495	439/544	567/693
	Recycles:No/cpu	*	171/64	162/59

In the case of the noisier Lenna image we see that again both versions of the AMG-R method achieve a significant decrease in the number of setup phases, however the performance of the AMG-R(3) method is worse than in the less noisy case particularly for the $\beta_h = 10^{-8}$ case, while the performance of the AMG-R(10) method is actually slightly better than in the less noisy case. In the case of the 512 × 512 fingers image, the performance (in terms of decrease in setups and increases in V-cycles) of the AMG-R methods is almost identical to the 256 × 256 case. The 10 fold increase in cpu time of the AMG method as compared to the 256 × 256 case can be potentially improved with a better implementation.

5.5 Conclusion

We have proposed a method for accelerating the fixed point method, when AMG is used as the inner linear solver, by recycling the AMG setup data. In the final algorithm an AMG setup phase was performed only when the number of V-cycles required on the previous fixed point

Table 5.9: Comparison of Fixed Point with AMG-R against Fixed Point with AMG for 512×512 Fingers image

β_h	Linear Solver	AMG	AMG-R(3)	AMG-R(10)
10^{-4}	FP Setps	62	60	60
	Total cpu	146754	7926	6043
	Setups: No/cpu	62/146035	3/6856	2/4564
	V-cycles: No/cpu	72/407	128/698	199/1104
	Recycles:No/cpu	*	57/81	58/84
10^{-8}	FP Setps	163	175	209
	Total cpu	37997	20820	16512
	Setups: No/cpu	163/377239	8/17516	5/11429
	V-cycles: No/cpu	301/1732	406/2219	672/3757
	Recycles:No/cpu	*	167/233	204/290

step exceeded some user-defined parameter ss. Experiments have shown that a significant decrease in the total number of setups can be achieved, for only a modest increase in the number of V-cycles performed, for a wide range of values of the parameter β_h. The reduction in cpu time achieved by the method is subject to caveats regarding our implementation of the AMG setup phase, nevertheless, even with a very efficient implementation of the setup phase ($\rho_1 = 4$) at least a halving of the cpu time can still be achieved. The optimal choice of the parameter ss will again depend on implementation costs but tests suggest it will increase with increasing noise levels.

Chapter 6

Multigrid Methods for Staircase Reducing Denoising

Useful Section References: §2.6, §3.2, §3.4, §3.5, §3.6, §4.1, §4.2, §5.1, §5.2, §A.6.3, §A.6.4, §A.6.5

Main Reference Material: [5, 6, 12, 19, 27, 30, 40, 41, 59, 62, 63],[67]-[71], [78, 92, 102, 96, 97]

In Chapter 3 we saw that although Total-Variation (TV) denoising has excellent edge-capturing properties it does suffer from the staircasing effect in which piecewise smooth regions in the original image tend to be recovered as piecewise constant. Over recent years there have been many attempts to devise denoising methods which capture edges as well as Total-Variation denoising but do not suffer from the staircasing effect. Unlike in the TV case where the model is well established and many possible iterative solvers have been proposed, the opposite is true in staircase reduction, many different models have been proposed, with less attention paid to developing efficient iterative solvers for the resulting equations. The aim in this chapter is firstly to try and develop nonlinear multigrid solvers similar to the one seen in Chapter 4 for several staircasing reduction models, which have an Euler-Lagrange equation of a similar form and then also to compare with other possible iterative solvers, specifically time marching and fixed point type methods which can also be easily generalized to the new problems. Results presented at the end of the chapter, show that nonlinear multigrid is an effective solver in several of the cases tested, although cannot be implemented effectively in all cases. One model in particular is found to both produce good quality reconstructions and be solved efficiently using nonlinear multigrid, which is shown to be faster and more robust

than several other iterative solvers. Much of the work presented here is also contained in [84].

The rest of the chapter is organized as follows. I first give a brief review of the many methods proposed in the literature to reduce the staircasing effect and then focus on a small number of these and attempt to implement nonlinear multigrid methods similar to the one seen in Chapter 4 for the solution of the resulting equations. Discretization of the resulting equations is discussed in §6.4 and algorithms for the iterative solvers considered given in §6.5, various implementation issues and some numerical results are presented in §6.6. Finally some conclusions are drawn in §6.7.

6.1 Reducing Staircasing: An Overview

In this section I review some of the ways to reduce staircasing that have been proposed in the literature, the specific models that I study here are introduced in the next section.

6.1.1 Higher Order Models

One way to reduce staircasing is to in some way introduce higher order derivatives into the regularization term. In [19] Chambolle and Lions do this by minimizing the inf-covolution of the TV norm and a second order functional

$$\min_{u_1,u_2} \int_\Omega |\nabla u_1| + \mu|\nabla(\nabla u_2)| + \lambda/2(u_1 + u_2 - z)^2. \tag{6.1}$$

Here u is decomposed into a smooth function u_2 and a function containing the discontinuities u_1.

Another way to use higher order derivatives is introduced in [30] by Chan et al in which the non-convex functional

$$\int_\Omega \alpha\sqrt{|\nabla u|^2 + \beta} + \mu\frac{(\Delta u)^2}{(\sqrt{|\nabla u|^2 + 1})^3} + 1/2(u - z)^2 \tag{6.2}$$

is minimized. Here the $(|\nabla u|^2 + 1)^{-3/2}$ term multiplying the higher order term ensures that true edges (with very large gradient) are not penalized while staircasing is reduced by the presence of the higher order term.

In [69] instead of combining the TV norm and second order derivatives within one regularization functional Lysaker and Tai use two regularization functionals.

$$E_1(u) = \int_\Omega |\nabla u| + \lambda_1/2(u-z)^2 \text{ and } E_2(v) = \int_\Omega (v_{xx}^2 + v_{xy}^2 + v_{yx}^2 + v_{yy}^2)^{1/2} + \lambda_2/2(v-z)^2. \tag{6.3}$$

Their approach is to use an iterative procedure in which they simultaneously apply an explicit time marching method to the Euler-Lagrange equation of each functional. After each step the current iterates u^k and v^k are combined in a convex combination to give $w = \theta^k u^k + (1-\theta^k)v^k$.

u^k and v^k are then overwritten with w in preparation for the next step. θ^k is chosen so that it is 1 only at the largest jumps (edges) allowing smaller jumps due to staircasing to be suppressed by the higher order PDE. In an earlier paper the same authors and Lundervold [70] considered E_2 on its own and another functional $\int_\Omega |u_{xx}| + |u_{yy}| + \lambda/2(u-z)^2$ which was not rotation invariant.

6.1.2 Combining TV and H^1

Another popular approach to reducing staircasing is to try and combine the ability of TV denoising to preserve edges with the ability of H^1 denoising to preserve smooth regions. In the next section I outline 4 such approaches for which I will attempt to use nonlinear multigrid to solve the resulting PDEs, here I mention several other methods which could be said to fall into this category. The first is the inf-convolution of the TV and H^1 regularization functionals proposed as an alternative to (6.1) in [19] the resulting minimization problem is equivalent to:

$$\min_u \int_{|\nabla u| \geq \epsilon} |\nabla u| + \epsilon/2 \int_{|\nabla u| < \epsilon} |\nabla u|^2 + \int_\Omega \lambda/2(u-z)^2. \tag{6.4}$$

Another approach proposed in [59] by Ito and Kunisch is to minimize the functional

$$\int_\Omega \alpha\psi(|\nabla u|^2) + 1/2(u-z)^2, \tag{6.5}$$

where ψ is chosen so that it behaves like $|\nabla u|$ for large values of $|\nabla u|$ and (in contrast to other models seen later) for small values of $|\nabla u|$ and behaves like $|\nabla u|^2$ for mid range values of $|\nabla u|$.

6.1.3 Other Ways to Reduce Staircasing

As mentioned in §3.6 Marquina and Osher [71] preconditioned the right hand side of the parabolic equation used in [78] with $|\nabla u|$ which had a staircase reducing effect. In a similar vein is the algebraic scaling approach used in [60] which is equivalent to using

$$u_t = \min\left(\frac{a_{max}}{2}(|\nabla u|^2 + \beta)^{1/2}, 1\right) \nabla . \left(\frac{\nabla u}{\sqrt{|\nabla u|^2 + \beta}}\right) + \lambda(z-u), \tag{6.6}$$

where a_{max} is a parameter to be chosen.

I also mention the Gauss-Curvature driven diffusion approach (not related to any optimization problem) proposed in [67] which has several desirable properties, including staircase reduction.

$$u_t = \nabla . \left(\left| \frac{u_{xx}u_{yy} - u_{xy}^2}{(1 + u_x^2 + u_y^2)^2} \right| \nabla u \right). \tag{6.7}$$

6.2 A class of PDE models from combining TV and H^1 norms

In this section I introduce the four Models for staircase reduction that I use. The emphasis will be to solve these models efficiently using nonlinear multigrid. All the models involve a minimisation problem of the form

$$\min_u \int_\Omega \alpha\Phi(|\nabla u|) + 1/2(u-z)dxdy. \tag{6.8}$$

which has Euler-Lagrange equation

$$-\alpha\nabla.\left(\Phi'(|\nabla u|)\frac{\nabla u}{|\nabla u|}\right) + (u-z) = 0 \tag{6.9}$$

with homogenous Neumann boundary condition $\frac{\partial u}{\partial n} = 0$. If the Euler-Lagrange equation is degenerate for $|\nabla u| = 0$ we must include a perturbing parameter β as in the TV case.

6.2.1 Model 1

Note that the TV case corresponds to $\Phi(g) = g$ and the H^1 case $\Phi(g) = g^2$, one simple approach proposed in various papers [6, 62], would be to take

$$\Phi(|\nabla u|) = \frac{1}{p}|\nabla u|^p \quad 1 < p < 2. \tag{6.10}$$

In this case

$$\Phi'(|\nabla u|) = |\nabla u|^{p-1} \text{ and } \Phi'(|\nabla u|)/|\nabla u| = \frac{1}{|\nabla u|^{2-p}}. \tag{6.11}$$

In order to recover edges reasonably well p should be close to 1, say 1.1.

6.2.2 Model 2

A more sophisticated approach would be to in some way make p adaptive. To this end Blomgren [5] and Blomgren, Chan, Mulet [6] proposed the following choice which results in a non-convex minimisation problem

$$\Phi(|\nabla u|) = |\nabla u|^{p(|\nabla u|)}, \tag{6.12}$$

where $p(g)$ is a monotonically decreasing function, $\lim_{g\to 0} p(g) = 2$ and $\lim_{g\to\infty} p(g) = 1$ i.e TV-like regularization ($p = 1$) is used at edges, H^1-like regularization ($p = 2$) is used in flat regions and in between p takes some value between 1 and 2. We have

$$\Phi'(|\nabla u|) = p(|\nabla u|)|\nabla u|^{p(|\nabla u|)-1} + p'(|\nabla u|)|\nabla u|^{p(|\nabla u|)}\log(|\nabla u|). \tag{6.13}$$

In [5] Blomgren suggests the following choice for p.

$$p(g) = \begin{cases} ag^3 + bg^2 + cg + d & g < sg_{max} \\ 1 & g \geq sg_{max} \end{cases}, \tag{6.14}$$

where the third order polynomial is chosen so that $p(0) = 2$, $p'(0) = 0$, $p(sg_{max}) = 1$, $p'(sg_{max}) = 0$, g_{max} is the maximum realizable gradient and $0 < s \leq 1$. Resolving the conditions on p gives $a = \frac{2}{(sg_{max})^3}$, $b = \frac{-3}{(sg_{max})^2}$, $c = 0$ and $d = 2$. If we assume that our image is a square $n \times n$ image with values in the range $[0, 255]$, then $g_{max} = 255\sqrt{2}(1/h)$ where h is the grid spacing (see later). We note here that in a later paper, Chan, Esedoglu, Park and Yip [27] suggested taking p to be a monotonically decreasing function from 2 to 0 e.g $p(g) = \frac{2}{1+2g}$, here we focus on the case that p takes values between 2 and 1.

6.2.3 Model 3

A slightly simplified alternative to (6.12) would be to make p dependant only on the position in the image (and thus 'less' nonlinear), i.e take

$$\Phi(|\nabla u|) = \frac{1}{p(|\nabla u^*|)}|\nabla u|^{p(|\nabla u^*|)}, \tag{6.15}$$

where u^* is some initial image from which the position of edges and smooth regions in the true image can be approximated. With this choice the resulting minimization problem is convex and we have

$$\Phi'(|\nabla u|) = |\nabla u|^{p(|\nabla u^*|)-1} \text{ and } \Phi'(|\nabla u|)/|\nabla u| = \frac{1}{|\nabla u|^{2-p(|\nabla u^*|)}}. \tag{6.16}$$

This approach was originally proposed as a possible alternative to (6.12) by Blomgren in [5] with p a function of $|\nabla G * z|$ where G is a Gaussian used to smooth the noisy image z. More recently this approach was used by Karkkainen and Majava in [63]. In [63] p is based on the gradient of the TV solution u_{TV}, the choice of p is as follows

$$p(|\nabla u_{TV}|) = \begin{cases} 2 & |\nabla u_{TV}| < g_1 \\ p_1(|\nabla u_{TV}|) & g_1 \leq |\nabla u_{TV}| \leq g_2 \\ 1 & |\nabla u_{TV}| > g_2 \end{cases}. \tag{6.17}$$

Where $p_1(g)$ is a second order polynomial satisfying $p_1(g_1) = 1.5$, $p_1(g_2) = 1$ and $p'(g_2) = 0$. The idea here is that a value of 1.5 is enough to recover smooth regions effectively with larger values possibly oversmoothing the image. In order that a nonlinear conjugate gradient solver can be implemented effectively p takes values 2 for $|\nabla u_{TV}| < g_1$ where g_1 is small, p then jumps to a value of 1.5 and then decreases smoothly as $|\nabla u_{TV}|$ increases until $|\nabla u_{TV}| = g_2$, g_2 being small enough so that $p = 1$ at all edges in the image. The values of g_1 and g_2 are chosen using a histogram of $|\nabla u_{TV}|$ values.

Another similar, but slightly different approach is that used by Chen, Levine and Rao in [40]. In this case

$$\Phi(|\nabla u|) = \begin{cases} \frac{1}{p(|\nabla u^*|)}|\nabla u|^{p(|\nabla u^*|)} & |\nabla u| \leq \epsilon \\ |\nabla u| - \frac{\epsilon p(|\nabla u^*|) - \epsilon^{p(|\nabla u^*|)}}{p(|\nabla u^*|)} & |\nabla u| > \epsilon \end{cases}, \tag{6.18}$$

where

$$p(|\nabla u^*|) = 1 + \frac{1}{1 + k|\nabla u^*|^2} \tag{6.19}$$

and $u^* = G*z$. The difference here is that the threshold for a switch to pure TV regularization is based on the gradient of u rather than on the gradient of the initial image u^*. The function p is a monotonically decreasing rational function which is 2 at $|\nabla u^*| = 0$ and tends to 1 as $|\nabla u^*|$ tends to infinity. In [40] the existence and uniqueness of solutions of this model and the behaviour of the gradient descent method are studied.

6.2.4 Model 4

Another approach proposed in [6, 5] tries to combine TV and H^1 in a convex combination. In this case Φ is of the form

$$\Phi(|\nabla u|) = \pi(|\nabla u|)|\nabla u| + (1 - \pi(|\nabla u|))|\nabla u|^2 \tag{6.20}$$

with $\lim_{g\to 0} \pi(g) = 0$ and $\lim_{g\to\infty} \pi(g) = 1$. In this case

$$\Phi'(|\nabla u|) = \pi'(|\nabla u|)(|\nabla u| - |\nabla u|^2) + \pi(|\nabla u|)(1 - 2|\nabla u|) + 2|\nabla u|. \tag{6.21}$$

One possibility, suggested in [5] is to take $\pi(g) = 2 - p(g)$ where p is the polynomial outlined in (6.14).

Next we shall consider how to solve these 4 models.

6.3 Solving the PDEs

As mentioned earlier less focus has been given to the efficient solution of the models of the previous section than their effectiveness in reducing staircasing. In [6] a fixed point type method is proposed to solve model 2 and model 4 but no numerical results are given. In [63] a nonlinear conjugate gradient method is used to solve model 3 with the particular choice of p outlined above. Our aim is to implement a nonlinear multigrid method similar to the one we used to solve the TV problem in Chapter 4 for the solution of the Euler-Lagrange equations of the 4 models outlined in the previous section and compare with explicit time marching and fixed point type methods. In the case of model 3 we also for comparison implement a nonlinear conjugate gradient solver (see §A.6.5). The Euler-Lagrange equation in all cases has the general form

$$-\alpha\nabla.\left(D(\sqrt{|\nabla u|^2 + \beta})\nabla u\right) + u = z. \tag{6.22}$$

Remark 6.3.1 *In the case of model 1 and model 3 D is similar to the TV case with the added advantage in model 3 that when $|\nabla u|$ is small $p(|\nabla u^*|)$ should be close to 2, preventing jumps in the diffusion coefficient as large as in the TV case. For models 2 and 4 the Euler-Lagrange equation is more nonlinear than in the TV case.*

In the next 3 sections we first outline our discretization scheme, we then give the algorithms for each of the iterative methods and then give details of implementation and numerical results.

6.4 Discretization

The domain Ω (see below for the choice of Ω) is partitioned into $n \times m$ rectangular cells of size $h \times k$ to produce a discrete cell-centered grid Ω^h. Denoting the discrete Euler-Lagrange equation by $N_h(u_h) = z_h$ (where u_h and z_h are grid functions on Ω^h) we have:

$$(N_h(u_h))_{i,j} = u_{i,j} - \alpha_h \left[\delta_x^- \left(D_{ij}(g_{ij})\delta_x^+ u_{i,j}\right) + \gamma\delta_y^- \left(D_{ij}(g_{ij})\gamma\delta_y^+ u_{i,j}\right)\right] = z_{ij}, \tag{6.23}$$

where

$$g_{i,j} = \frac{1}{h}\sqrt{(\delta_x^+ u_{i,j})^2 + (\gamma\delta_y^+ u_{i,j})^2 + \beta_h}. \tag{6.24}$$

$$D_{ij}(g_{ij}) = \begin{cases} \frac{1}{h}\left[g_{ij}^{-(2-p)}\right] & \text{model 1} \\ \frac{1}{h}\left[p(g_{ij})g_{ij}^{p(g_{ij})-1} + p'(g_{ij})g_{ij}^{p(g_{ij})}log(g_{ij})\right]g_{ij}^{-1} & \text{model 2} \\ \frac{1}{h}\left[g_{ij}^{-(2-p_{i,j})}\right] & \text{model 3} \\ \frac{1}{h}\left[\pi'(g_{ij})(g_{ij} - g_{ij}^2) + \pi(g_{ij})(1 - 2g_{ij}) + 2g_{ij}\right]g_{ij}^{-1} & \text{model 4} \end{cases}. \tag{6.25}$$

$$\alpha_h = \alpha/h, \ \beta_h = h^2\beta \text{ and } \gamma = h/k \tag{6.26}$$

and

$$\delta_x^{\pm} u_{i,j} = \pm\left(u_{i\pm1,j} - u_{i,j}\right) \qquad \delta_y^{\pm} u_{i,j} = \pm\left(u_{i,j\pm1} - u_{i,j}\right). \tag{6.27}$$

Note that D is actually only dependant on (i,j) in the case of model 3. We also have boundary condition.

$$u_{i,0} = u_{i,1}, u_{i,m+1} = u_{i,m}, u_{0,j} = u_{1,j}, u_{n+1,j} = u_{n,j}. \tag{6.28}$$

Once again we note that (6.23) is equivalent to $(\nabla J_h)_{i,j} = 0$ where

$$J_h(u_h) = \sum_{i,j} h\alpha_h \Phi_{i,j}(g_{i,j}) + 1/2(u_{i,j} - z_{i,j})^2. \tag{6.29}$$

is the discrete functional and

$$\Phi_{ij}(g_{ij}) = \begin{cases} g_{i,j}^p & \text{model 1} \\ g_{i,j}^{p(g_{i,j})} & \text{model 2} \\ g_{i,j}^{p_{i,j}} & \text{model 3} \\ \pi(g_{i,j})g_{i,j} + (1 - \pi(g_{i,j}))g_{i,j}^2 & \text{model 4} \end{cases}. \tag{6.30}$$

Unlike in the TV case where the choice of Ω is not important provided α_h and β_h are chosen to be the same, whatever the value of h, there is not in all cases here a straightforward relationship between the case $\Omega = (0, n) \times (0, m)$ i.e $(h, k) = (1, 1)$ and the case $\Omega = (0, 1) \times (0, 1)$ i.e $(h, k) = (1/n, 1/m)$. We have chosen the former to be consistent with the majority of papers that we have seen on this subject. In the case of model 1 there is a straightforward relationship, choosing $\alpha_h = a$ when $h = 1/n$ is equivalent to choosing $\alpha_h = a(n^{p-1})$ when $h = 1$.

6.5 Algorithms

For clarity I give the algorithms for the time marching (Algorithm 18), fixed point (Algorithm 19), nonlinear conjugate gradient (Algorithm 20) and nonlinear multigrid (Algorithm 21) methods below.

6.5.1 Time Marching

Algorithm 18 Time Marching

Choose initial guess u_h^0
Set $k = 0$.
While $\|(z_h - N_h(u_h^k))\|_2 >$ tol
$u_h^{k+1} = u_h^k + \Delta t \left[z_h - N_h(u_h^k)\right]$
$k = k + 1$
end

The time step Δt is determined by experiment as the largest value which gives stability of the algorithm, *tol* is typically $10^{-4}\|(z_h - N_h(z_h))\|_2$. The time step can also be determined automatically via a backtracking line search on the discrete functional i.e steepest descent (§A.6.4).

6.5.2 Fixed Point Method

The linear operator $L_h(u_h^k)$ on step $k + 1$ is given by the stencil:

$$\begin{bmatrix} 0 & -\alpha\gamma D_{ij}(g_{ij}^k) & 0 \\ -\alpha D_{i-1,j}(g_{i-1,j}^k) & 1 + \alpha\Pi_{ij} & -\alpha D_{ij}(g_{ij}^k) \\ 0 & -\alpha\gamma D_{i,j-1}(g_{i,j-1}^k) & 0 \end{bmatrix}. \tag{6.31}$$

where $\Pi_{ij} = (1 + \gamma)D_{ij}(g_{ij}^k) + D_{i-1,j}(g_{i-1,j}^k) + \gamma D_{i,j-1}(g_{i,j-1}^k)$. The linear solver used in most cases is a geometric multigrid method similar to the one used in the TV case (§5.2.2). We only require a relatively small decrease in the linear residual (typically a halving) as this

Algorithm 19 Fixed Point

Choose initial guess u_h^0

Set $k = 0$.

While $\|(z_h - N_h(u_h^k))\|_2 > \text{tol}$

Set u_h^{k+1} to be the result of applying some iterative method to the linear system:

$L_h(u_h^k)w_h = z_h$

$k = k + 1$

end

seems to give the best results in terms of overall cpu time. For the FPGS smoother we use (unless otherwise stated) 3 steps of pointwise lexicographical Gauss-Seidel as linear solver.

6.5.3 Nonlinear Conjugate Gradient

Algorithm 20 Nonlinear Conjugate Gradient

Choose Initial Guess u_h^0.

Evaluate $r_h^0 = z_h - N_h(u_h^0)$.

Set $d_h^0 = -r_h^0$, $k = 0$.

While $\|(z_h - N_h(v_h))\|_2 > \text{tol}$

Choose ρ such that $u_h^k + \rho d_h^k$ satisfies the strong Wolfe conditions (A.94) and (A.96) with respect to (6.29).

$u_h^{k+1} = u_h^k + \rho d_h^k$.

$r_h^{k+1} = z_h - N_h(u_h^{k+1})$.

$\beta^{PR} = \frac{(r_h^{k+1}, r_h^{k+1} - r_h^k)_2}{(r_h^k, r_h^k)_2}$

$\beta^{PR} = \max(\beta^{PR}, 0)$.

$d_h^{k+1} = r_h^{k+1} + \beta^{PR} d_h^k$.

end

The value of λ and σ used in the Wolfe conditions (A.94) and (A.96) are 10^{-4} and 0.5 respectively.

6.5.4 Nonlinear Multigrid

The algorithm for the nonlinear multigrid method is similar to the TV case (Algorithm 14), for clarity I present it here.

In Algorithm 21 $v_h \leftarrow FPGS(v_h, N_h, z_h)$ denotes the updating of v_h via one step of the FPGS smoother. N_{2h} is the coarse grid analogue of N_h which results from standard coarsening i.e the nonlinear operator which results from discretizing the Euler-Lagrange equation using a cell-centered grid with grid spacing $(2h, 2k)$. We use V-cycles and the value of ν_1 and

Algorithm 21 Nonlinear Multigrid Method

Set v_h to be some initial guess.

While $\|(z_h - N_h(v_h))\|_2 > \text{tol}$

$v_h \leftarrow FAS1^h(v_h, N_h, z_h, \nu_1, \nu_2)$

end

where FAS_μ^h is defined recursively as

$$v^h \leftarrow FAS1^h(v_h, N_h, z_h, \nu_1, \nu_2).$$

1. If Ω^h =coarsest grid, solve $N_h u_h = z_h$ using Fixed Point Method and stop.
   Else For $l = 1, .., \nu_1$ $v_h \leftarrow FPGS(v_h, N_h, z_h)$
2. $v_{2h} = I_h^{2h} v_h$
   $\bar{v}_{2h} = v_{2h}$
   $z_{2h} = I_h^{2h}(z_h - N_h v_h) + N_{2h} v_{2h}$
3. $v_{2h} \leftarrow FAS1^{2h}(v_{2h}, N_{2h}, z_{2h}, \nu_1, \nu_2)$
4. $v_h \leftarrow v_h + I_{2h}^h(v_{2h} - \bar{v}_{2h})$
5. For $l = 1, .., \nu_2$ $v_h \leftarrow FPGS(v_h, N_h, z_h)$

ν_2 depends on the model (see the results section for details). We use standard cell-centered interpolation and restriction operators seen earlier (§2.6.3).

6.6 Implementation Issues and Numerical Results

In this section we present some numerical results and give details of some of the issues regarding our implementation of iterative methods for each of the four models. Tests are run on the test hump image seen in Figure 6.2, which has both smooth regions, high intensity edges and low intensity edges and the more realistic Lenna image shown in Figure 6.3. In each case we have tried to choose parameters which give the optimal reconstruction, focusing on the need to reduce staircasing. What the optimal reconstruction is, is somewhat subjective, as a guide we have used mesh and image plots as well as Peak signal to noise ratio PSNR (see §3.7). The PSNR does not always give a clear guide as to whether one image is less staircased than another as can be seen in the hypothetical 1-d example in Figure 6.1, so we also take into account the value of $PSNR_{grad}$ which we define as $1/2(PSNR(u_x, u_x^0) + PSNR(u_y, u_y^0))$ this should measure how well the derivatives of the reconstruction match those of the true image. All methods were implemented in MATLAB.

In Figure 6.4, we present some plots showing the results of applying each of the four models

Figure 6.1: A simple 1-d example of a staircased reconstruction (squares) which will have a higher PSNR than the smooth reconstruction (stars), the smooth reconstruction in this case has exactly the same gradient as the true solution (circles)

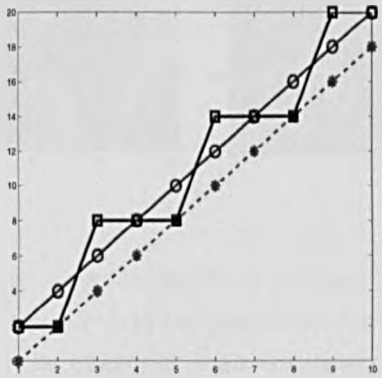


Figure 6.2: Mesh plots of true (left) and noisy (right) Hump image

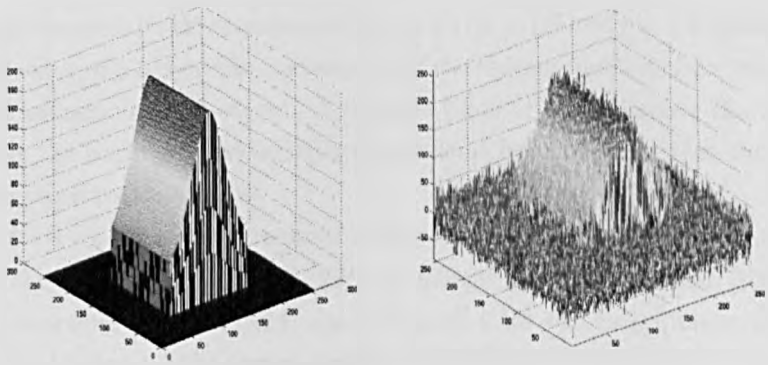


Figure 6.3: True (left) and noisy (right) Lenna image

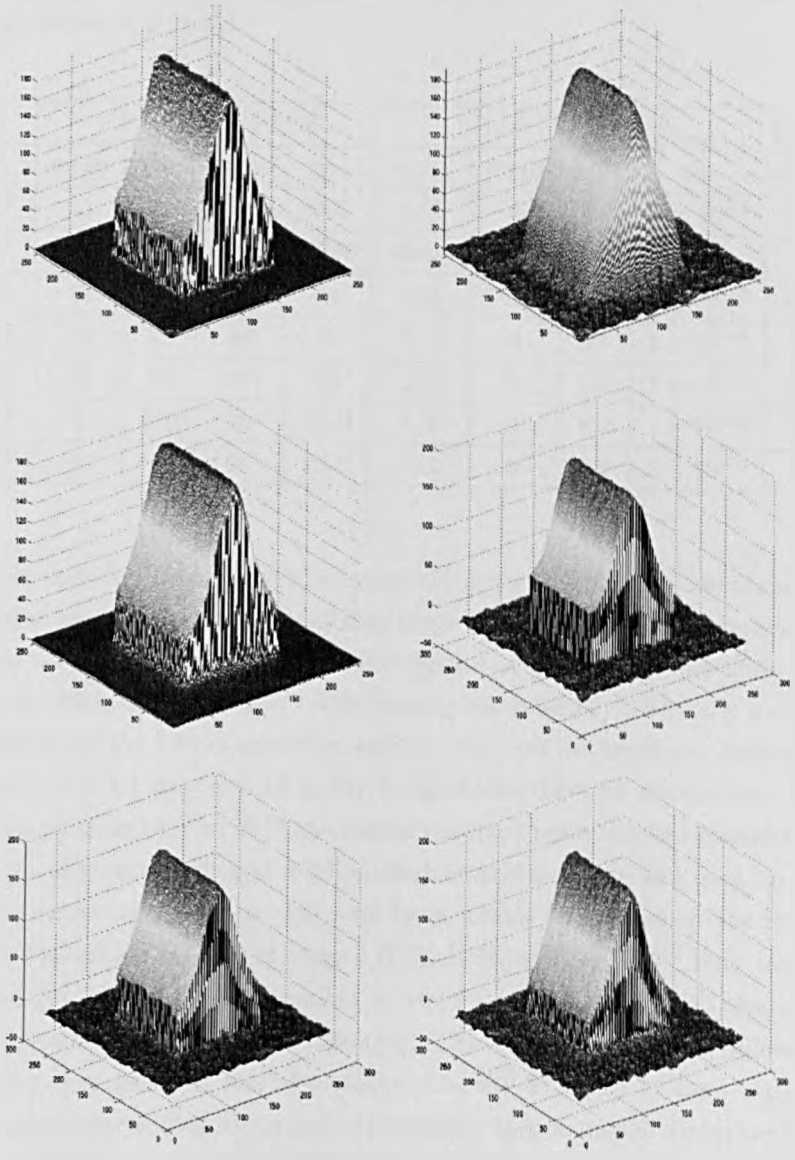
to the test hump image, we also show the results of applying TV and H^1 regularization. We remark that it is not our intention to carry out a detailed comparison of the various staircase reducing methods in terms of the quality of the reconstructed images, however we make a few general comments. To some extent all the models can recover better the smooth regions of the image than the original TV model (3.15) but in our experience models 2 and 3 seem to give better overall results than model 1 (as would be expected) and model 4 in which there is some oversmoothing of the edges (particularly the low intensity edges), this is predicted in [5]. With models 2 and 3 for the test image shown we have been able (with suitable choices of parameters) to reduce the staircasing present in the TV reconstructed image while still recovering well the high and low intensity edges in the image.

6.6.1 Model 1

For this model we consider three choices of p, $p = 1.1$, $p = 1.5$ and $p = 1.9$ mainly to highlight the effect the value of p has on the convergence of the various methods (the latter two choices will of course oversmooth the edges). A suitable value of α_h to remove the noise is chosen for each value, the larger p is, the smaller α_h needs to be. The effect that the parameter β_h has on convergence is also studied.

In Table 6.1 we show results (number of steps required for convergence and cpu time in seconds) for the Fixed Point method (FP), Nonlinear multigrid method (NLMG) and the explicit time marching method (TM) run on model 1 for the hump image with 3 different values of p, 1.1, 1.5 and 1.9 the corresponding values of α_h are 52, 24 and 15. Also shown are results for the smoother (FPGS) run on its own. Shown are results for various values of β_h. In all cases the initial guess is taken to be the noisy image z and the stopping criteria is a reduction in the residual by a factor of 10^{-4}. As linear solver in the fixed point method we use

Figure 6.4: From top left to bottom right, the images recovered using TV, H^1, model 1 ($p = 1.1$), model 2, model 3 and model 4



a linear multigrid method with 2 pre and 2 post correction smoothing steps of Gauss-Seidel relaxation and stop the iterations when the linear residual has been reduced by a factor of 0.5. Shown in the table are the choices of ν_1 and ν_2 which give the optimal nonlinear multigrid method for each case, also shown is the value of the time step in the time marching method.

Table 6.1: Comparison of Fixed Point, Time Marching and Nonlinear Multigrid for Model 1 with various choices of p and β

p	β_h	FP		FPGS		NLMG			TM		
		steps	cpu	steps	cpu	ν_1/ν_2	steps	cpu	Δt	steps	cpu
1.1	10^{-2}	43	73	748	680	5/5	4	34	5×10^{-4}	9502	2540
	10^{-4}	73	216	4389	4036	10/10	4	66	*	*	*
1.5	10^{-2}	14	19	78	61	1/1	6	13	1×10^{-3}	4054	536
	10^{-4}	16	23	94	74	1/1	6	13	1×10^{-3}	4053	536
	10^{-10}	16	23	119	93	1/1	6	13	5×10^{-4}	8150	1131
1.9	10^{-2}	6	8.8	29	23.9	1/1	3	6.9	1×10^{-2}	303	56
	10^{-10}	6	8.8	29	23.9	1/1	3	6.9	1×10^{-2}	303	56

We observe that the closer p is to 2 the easier the problem is to solve, less steps are required for each of the methods and less smoothing steps are required in the nonlinear multigrid method. We see that for $p = 1.9$ the convergence of the various methods is seemingly invariant to the value of β_h. For $p = 1.5$ decreasing the value of β_h has only a small effect on the FP method and the FPGS smoother and no effect on the nonlinear multigrid method. In the case that $p = 1.1$ the value of β_h has a significant effect on convergence. We see that as β_h is decreased from 10^{-2} to 10^{-4} the cost of the fixed point method increases by 3 times. The cost of the nonlinear multigrid method doubles and more pre and post correction steps are needed to ensure convergence. We have been unable to implement the time marching method in a reasonable number of steps. If β_h is reduced to 10^{-10} only the fixed point method converges in a reasonable number of steps (in this case a PCG linear solver with Cholesky preconditioner gives the best results). This breakdown of the nonlinear multigrid convergence for very small β_h was also observed in the TV ($p = 1$) case. Apart from this last case the nonlinear multigrid method significantly speeds up the smoother FPGS and is faster than the time marching and fixed point methods.

6.6.2 Model 2

For this model $p(|\nabla u|)$ is chosen to be the polynomial (6.14). There were several problems that occurred during the implementation of iterative solvers for this model. The first problem is that the functional is non-convex and the initial guess seems to have an effect on the quality of the final image. If we take the noisy image z as initial guess we appear to converge to a minimum which is still highly oscillatory (in this case we use steepest descent as solver). To achieve the reconstruction of the test image shown in Figure 6.4 we had to take the solution to the TV problem as initial guess, the following discussion relates to experiments run using this initial guess.

Another point to note is that unlike in the TV case (and some of the other models considered here) the D_{ij} terms can take negative values, as a consequence the linear system in the fixed point method is not necessarily positive definite. We have been unable to implement our smoother FPGS successfully, this is due to divergence of the inner gauss-seidel steps, instead we have used a slight modification of this smoother which we will denote FPGS2. Instead of updating u^{k+1} by applying 3 Gauss-Seidel steps to the linear system $L(u_h^k)w_h = z_h$ we apply 3 Gauss-Seidel steps to the linear system $(\lambda + L(u_h^k))w_h = z_h + \lambda u_h^k$ (essentially we add a λu term to both sides of the Euler-Lagrange equation and lag the right hand side term). Taking λ large enough will ensure diagonal dominance of the inner linear system and hence positive definiteness, which ensures convergence of the Gauss-Seidel steps. In addition we have also used this approach when implementing the fixed point method. We tried to implement the fixed point method in its original form but had problems finding a suitable inner solver (linear multigrid did not converge and PCG was not an option) we settled on the minimum residual method but found that the outer fixed point steps stagnated, this was also the case when we used a direct solver to solve the linear system. Under-relaxation of the fixed point iterates was required to ensure convergence, with the under-relaxation parameter in some cases being quite small. Using the modified fixed point method, we can use linear multigrid or PCG as the inner linear solver and the outer steps also converge.

We implemented the time marching method, the modified fixed point method and the nonlinear multigrid method with FPGS2 smoother on the test hump image using a value of $s = 0.2$, $\alpha_h = 10$ and $\lambda = 7$, in this case only 2 pre and 2 post correction smoothing steps were required in the nonlinear multigrid method which converged in 9 steps and was around 1.75 times as fast as the modified fixed point method and over 5 times as fast as the time marching method. However when we tried to implement this model for the Lenna image we could not achieve a reasonable quality reconstruction, the image tended to look too blurred or be contaminated with undesirable artifacts. In addition we found that the nonlinear multigrid method is not effective in that the convergence stagnates unless a large number (10 or more) of smoothing steps is used and the total number of smoothing steps in this case is more than if the smoother were run on its own. The convergence of the modified

Table 6.2: Comparison of Fixed Point, Time Marching, Nonlinear Multigrid and Nonlinear Conjugate Gradient (NLCG) for Model 3 on the hump image (left) and Lenna image (right)

	Hump		Lenna	
Method	Steps	cpu(s)	Steps	cpu(s)
FP	9	12.2	10	12.4
FPGS	32	18.6	22	13.1
NLMG	2	6.3	3	9.8
TM	211	32.5	169	24.8
NLCG	42	21.7	35	19.5

fixed point method also seems somewhat unstable and typically the number of steps required by the modified fixed point and time marching methods is considerably larger than the case of the hump image above. We note that some of the problems with the iterative methods described above also occur in the case of the hump image for larger values of s (although these do not produce good reconstructions). More work is needed on this model before we can draw any firm conclusions.

Finally we note that the value of β_h seems to have no effect on convergence for this model and so it is taken to be very small (10^{-10}) in the implementation.

6.6.3 Model 3

We have implemented model 3 with the choice of $p(|\nabla u^*|)$ described by (6.17). We have been able to implement a working nonlinear multigrid method (with the usual FPGS smoother) as well as the fixed point and time marching methods. Since a nonlinear conjugate gradient (NLCG) solver is used in [63] for this model we also implement a version here for comparison.

For the parameters g_1 and g_2 in (6.17) we take $g_1 = g^*_{max}/50$ (as in [63]) and $g_2 = sg^*_{max}$ where $0 < s < 1$ and is chosen to give the best visual results, g^*_{max} is the maximum value of $g^*_{i,j}$ over all (i,j) where the $g^*_{i,j}$ is the discretization of $|\nabla u^*|$ at grid point (i,j), u^* in this case being the TV solution u_{TV}.

In Table 6.2 results of running FP, NLMG, TM and NLCG on model 3 for the hump test image are shown. In this case we take $s = 0.2$ and $\alpha_h = 30$, β_h in this case appears to have no effect on convergence and is taken to be 10^{-10}. We take z as the initial guess and the same stopping criteria as above is used. In the nonlinear multigrid method 2 pre and 2 post correction smoothing steps are used. For the fixed point method linear multigrid is used as the linear solver again with the same stopping criteria as in model 1. The time step in the time marching method is $\Delta t = 8.0 \times 10^{-3}$.

We observe that the nonlinear multigrid method reduces the cost of the smoother alone

by approximately 65%. Nonlinear multigrid is around 2.1 times faster than the fixed point method, around 5.2 times as fast as the time marching method and around 3.4 times as fast as the nonlinear conjugate gradient method.

In our second test, we compare the performance of fixed point, time marching and nonlinear multigrid on the more realistic Lenna image. In this case we take $s = 0.9$ and $\alpha_h = 11$. The implementation is as above, except that the time step $\Delta t = 2.2 \times 10^{-2}$ is used in the time marching method. The usual initial guess and stopping criteria are used, results are given in Table 6.2 (right).

In this case the speed up in the smoother achieved by the nonlinear multigrid method is around 35% (note that the smoother itself is competitive with the fixed point method), the nonlinear multigrid method is around 1.3 times as fast as the fixed point, around 2.4 times faster than the time marching method and around twice as fast as the nonlinear conjugate gradient solver.

Other Choices of p and u^*

One of the advantages we have found with using Model 3 over Models 2 and 4 is that iterative solvers and in particular the nonlinear multigrid method are robust with respect to different choices of p, since changing p does not alter the nature of the Euler-Lagrange equation, only the exponent of $g_{i,j}$ at each grid point. Focusing on the hump image we outline below the performance of the iterative solvers for several different choices of p that we have considered. We consider the following choices for p

1. A general quadratic which satisfies $p(0) = q$, $p(sg^*_{max}) = 1$ and $p'(g^*_{max}) = 0$, where $1 < q \leq 2$. Specifically here we consider $q = 2$ and $q = 1.5$ we note that the latter choice is equivalent to the p from [63] with $g_1 = 0$.

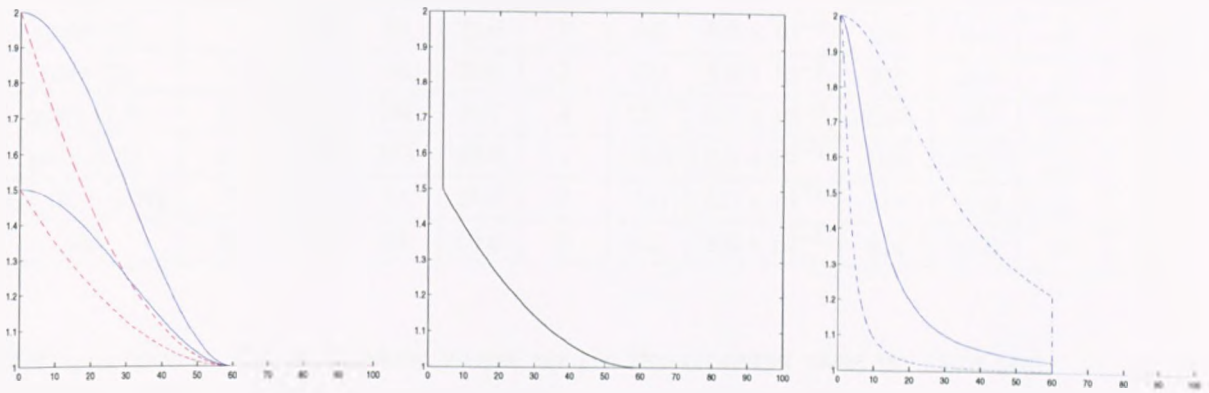
2. A general third order polynomial satisfying $p(0) = q$, $p'(0) = 0$, $p(sg^*_{max}) = 1$ and $p'(g^*_{max}) = 0$ again taking $q = 2$ and $q = 1.5$, the former choice here is like the polynomial used in model 2.

3. A rational function like that used by Chen et al in [41] but with the threshold for switch to TV regularization based on $|\nabla u^*|$. See equation (6.19) above.

Note that in all cases $p = 1$ whenever $g^*_{i,j} > sg^*_{max}$ with s being a user defined parameter. Figure 6.5 illustrates the nature of each choice with a plot of p against g, also shown is the choice from [63] used above. Note that in the case of the rational (right plot) the choice of k in (6.19) is important, in the experiments below we take $k = .01$.

In Table 6.3, information is given on the cost of running the three iterative solvers with each choice of p on the hump image. In each case we take $s = 0.2$, $\alpha_h = 30$ (this choice is reasonable in all cases, but the optimal α may vary with p) and $\beta_h = 10^{-10}$. In all cases

Figure 6.5: Plots of p against g for polynomial (full line) and quadratic (dashed line) from 2 and 1.5 (left), choice of Karkkainen and Majava (centre) and rational with various values of k (right).



we use 2 pre and 2 post correction smoothing steps in the nonlinear multigrid method, the fixed point method is implemented as above and the time step for the time marching method is given in the table. Results for the Karkkainen, Majava (km) type choice are reproduced for comparison. We observe that all the choices which have $p(0) = 2$ behave similarly in terms of convergence, while the costs associated with the choices satisfying $p(0) = 1.5$ are larger. Note that for these more difficult problems, the advantage of the nonlinear multigrid method over the other methods has grown. The time marching method suffers particularly badly, the multigrid method being up to 100 times faster as compared to around 7 times for other choices, while the nonlinear conjugate gradient is as much as 16 times slower than the nonlinear multigrid method, as compared to three and a half times slower for other choices. We remark that, the choice of Karkkainen and Majava was used originally to avoid difficulties in implementing their nonlinear conjugate gradient method for such choices. Taking q even closer to 1 will lead to a further increase in cost and eventually to a sensitivity to small values of β, although as we see below, there is likely to be no advantage in terms of image quality.

Remark 6.6.1 *The costs of one step of the various methods for the km choice is slightly less than for other choices, this is because the cpu time associated with evaluating* $g_{ij}^{p_{i,j}-1}$ *is slightly less for* $p_{i,j} = 1$ *or 2 than it is for* $1 < p_{i,j} < 2$ *and the km choice has* $p_{i,j} = 2$ *at more places.*

Finally we make some comment about the quality of the reconstructed image in this case. In Figure 6.6 we show a 1-dimensional slice through the reconstructed hump image, for each choice of p used above as well as the original image and the TV solution. We see that all results are clearly less staircased than the TV solution which has the lowest $PSNR$ and

Table 6.3: Comparison of iterative solvers for various choices of p on hump image

	FP		FPGS		NLMG		TM			NLCG	
p	steps	cpu	steps	cpu	steps	cpu	Δt	steps	cpu	steps	cpu
quad (2)	7	11.3	35	22.0	2	6.8	8.0×10^{-3}	222	46.4	43	27.0
poly (2)	7	13.3	35	21.9	2	6.9	8.0×10^{-3}	223	48.5	41	25.9
quad (1.5)	17	28.3	158	99.7	4	13.7	6.0×10^{-4}	6254	1380	258	227.3
poly (1.5)	16	27.0	153	96.8	4	13.6	6.0×10^{-4}	6135	1322	198	171.9
rat ($k = 0.01$)	7	11.2	34	21.8	2	7.0	8.0×10^{-3}	218	47.0	42	26.3
km	9	12.2	32	18.6	2	6.3	8.0×10^{-3}	211	32.5	42	21.7

$PSNR_{grad}$ values. The best looking images are the two recovered using the third order polynomial choice of p (third row), the $q = 2$ choice seems to look the smoothest and indeed has the highest $PSNR_{grad}$ value of all choices (a 15% increase on TV), the $q = 1.5$ choice achieves the highest $PSNR$ value of all choices (a 4% increase on the TV). Although it is hard to see from the figure their is less loss of intensity at the peak of the hump, than in the $q = 2$ case and the background is slightly less oscillatory as well. We note that the second highest $PSNR$ value was achieved with the polynomial with $q = 2$ and the second highest $PSNR_{grad}$ value with the polynomial with $q = 1.5$. Given that the $q = 1.5$ case is slightly less smooth on the sides of the hump than the $q = 2$ choice there is likely to be no advantage in taking q even smaller.

Of the other choices the km choice achieved the lowest $PSNR$ and $PSNR_{grad}$ values (apart from TV), we see that visually the reconstruction looks similar to the quadratic with $q = 1.5$, but the background is slightly more noisy.

Remark 6.6.2 *The hump image was designed to have simply smooth regions and edges, for the Lenna image which also includes a lot of fine detail, PSNR and $PSNR_{grad}$ values achieved with all choices, including TV, are very similar, visually a reduction in staircasing can be seen with all p choices used above (provided suitable parameters are chosen) but it is difficult to say which is best.*

As well as using different p, different choices of u^* can also be used, all the results presented above are for $u^* = u_{TV}$, but we have observed similar convergence properties for $u^* = G * z$ where G is a Gaussian used to smooth the original noisy image. This choice does of course avoid the cost of having to solve the TV problem first. Taking $u^* = G * z$ may give slightly better visual results for the Lenna image, while for the hump $u^* = u_{TV}$ appears to be best. We note that u^* can also be iterated, although we have found no real advantage in doing more than two iterations.

Figure 6.6: From left to right and down, 1-d slice of true image and images recovered using TV, quadratic 2 and 1.5, polynomial 2 and 1.5, rational and km

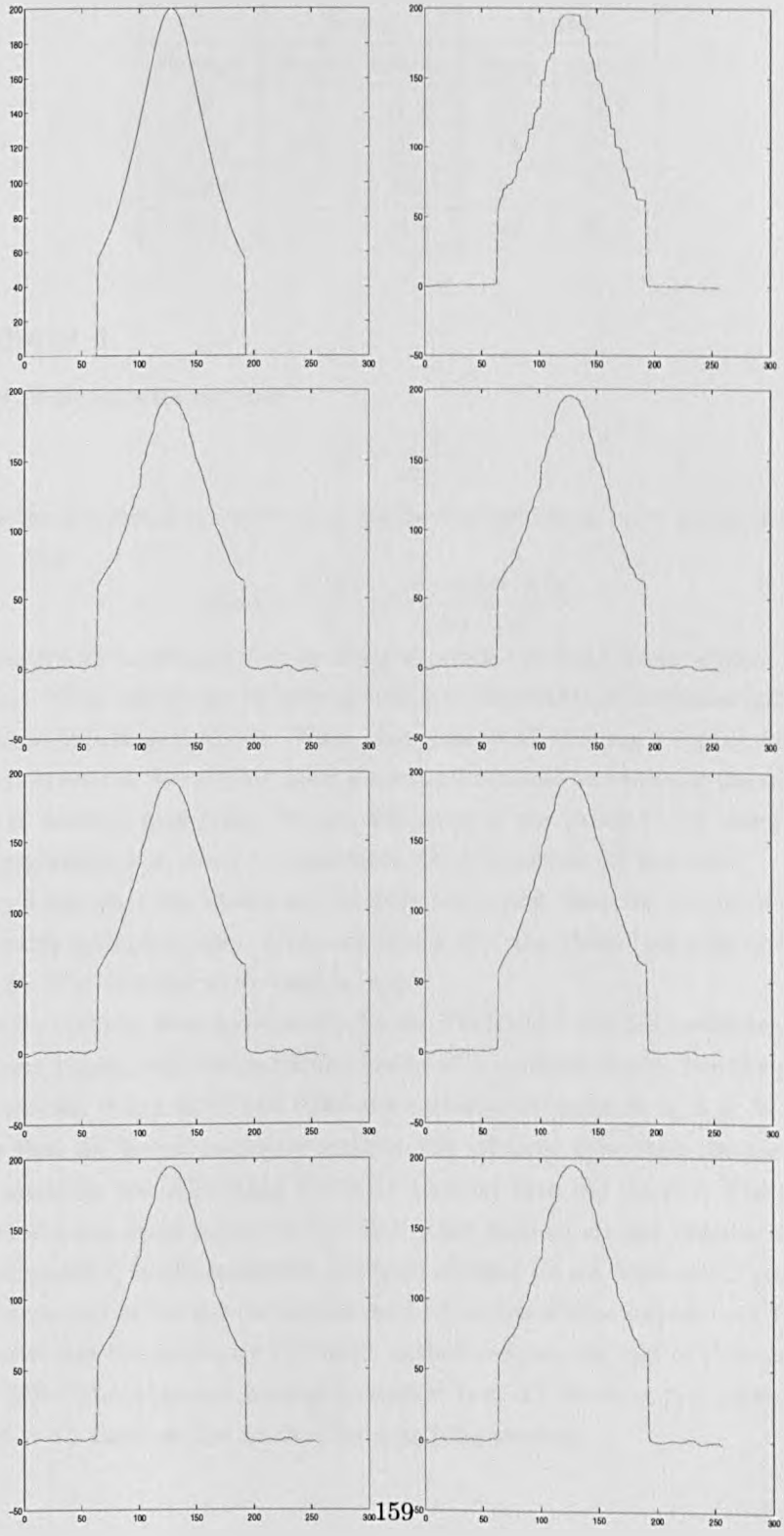


Table 6.4: Comparison of Fixed Point, time Marching and Nonlinear Multigrid for Model 4 on the hump image (left) and Lenna image (right)

	Hump		Lenna	
Method	Steps	cpu(s)	Steps	cpu(s)
FP	16	17.9	22	24.7
FPGS	140	31.3	78	17.5
NLMG	6	8.0	8	10.3
TM	378	34.2	245	21.8

6.6.4 Model 4

We consider (6.20) only for the case

$$\pi(x) = \frac{\epsilon x}{\epsilon x + q}. \tag{6.32}$$

In this case the functional is convex (see [6] for the conditions on π required for a convex functional). Also

$$D(x) = \frac{\Phi'(x)}{x} = \frac{(\epsilon + q)(\epsilon x + 2q)}{(\epsilon x + q)^2}. \tag{6.33}$$

which is positive for nonnegative x ensuring a positive definite linear system in the fixed point method. With this choice we have successfully implemented nonlinear multigrid, fixed point and time marching methods. With other choices of $\pi(x)$ e.g $2 - p(x)$ where p is the third order polynomial, we may not have a convex functional and some of the same issues as in the case of Model 2 may arise. We are not aware of the choice (6.32) being used before but in our experience it is easier to implement iterative solvers for this case.

We have found that the choice of ϵ is more important than the choice of q in obtaining a reasonable reconstruction. With our choice of π the Euler-Lagrange equation is not degenerate for $|\nabla u| = 0$ and so we take $\beta_h = 0$.

In Table 6.4 (left) we show some results for the FP, NLMG and TM methods run on model 4 for the hump image, with the particular choice of π outlined above. For the parameters ϵ and q in π we take values 0.001 and 0.005 respectively, the value of α_h is 9. We have found in this case that the fastest multigrid method was achieved if we took the parameter *it* in the FPGS smoother (see Algorithm 13) to be 1 rather than the usual 3. The initial guess, stopping criteria and linear solver for the fixed point method are the same as in the case of model 1 and model 3. In the nonlinear multigrid method we use 2 pre and 2 post correction smoothing steps and in the time marching method we use a time step $\Delta t = 1.3 \times 10^{-2}$.

We observe that the nonlinear multigrid method reduces the cost of the smoother alone by around 75%. The nonlinear multigrid method is ≈ 2.2 times as fast as the fixed point method and ≈ 4.3 times as fast as the time marching method.

Figure 6.7: Close up of Lenna Image recovered using model 3 (left) and model 4 (centre), with TV result (right) for comparison, notice the reduction in staircasing on the face and shoulder

We also applied model 4 to the Lenna image, results are shown in Table 6.4 (right). The value of q and ϵ are as above, but $\alpha_h = 5$. The implementation is as above, except that the time step in the time marching method is $\Delta t = 2.7 \times 10^{-2}$.

In this case the FPGS smoother on its own performs quite well and is actually slightly faster than the fixed point method with linear multigrid inner solver, this suggests that we are oversolving the linear equation in the fixed point method and we can view the FPGS smoother with just one gauss-seidel sweep as an optimal fixed point method. The nonlinear multigrid method is 1.7 times faster than FPGS. The time marching method is actually quite competitive in this case at around twice the cost of the nonlinear multigrid method.

Remark 6.6.3 *Although model 4 did not perform that well on the hump image with over-smoothing of some edges, we have observed for more realistic images like the Lenna image, where the intensity of edges is more uniform, this model does not perform that badly in comparison with model 3 as can be seen from the plots in Figure 6.7.*

6.7 Conclusion

We studied several staircasing-reducing regularization methods and compared three different iterative solvers for the solution of the resulting equations, a nonlinear multigrid method similar to the one we used in Chapter 4 for the TV problem, a fixed point method similar to the method used by Vogel and Oman for solving the TV problem and an explicit time marching method. In the case of model 3 a nonlinear conjugate gradient solver was also compared as this approach was used in [63]. By studying the simple model 1 we observed that the closer the exponent p of the $|\nabla u|$ term was to 2 the less steps were required by each of the iterative methods and the less smoothing steps were required for each step of the

multigrid method, in addition while the convergence of the multigrid method breaks down for small β_h when p is close to 1 for larger p the convergence properties are invariant to the value of β_h. In this case large values of p oversmooth the edges, however the convergence properties seen for large fixed p also seem to carry over to the case of model 3 where the value of p varies with position in the image and is based on the gradient of some initial image u^*, taking a value 1 at edges and a value between 1 and 2 in smooth regions and flat regions. We tested the specific choice of $p(|\nabla u^*|)$ used in [63] and several other more general cases. In all cases the nonlinear multigrid method was the fastest iterative solver with its advantage over the other methods growing for the harder choices. We have also observed that this model produces good quality reconstructions reducing the staircasing effect while still recovering sharp edges.

While the non-convex model 2, in which p is chosen to be a monotonically decreasing function of $|\nabla u|$, can also with a suitable initial guess produce good quality reconstructions for the simple hump image we encountered some problems when attempting to implement iterative solvers for this problem. We had some success with a nonlinear multigrid method using a modified version of the FPGS smoother and with a modified fixed point method when denoising our simple hump image, however there seems to be a lack of robustness of the iterative solvers with respect to different images and changes in parameters. In addition we had problems achieving good quality reconstructions for the Lenna image.

In the case of model 4 we found a choice for π which resulted in a convex functional and did not require the use of a perturbing parameter β in the Euler-Lagrange equation. In this case fewer inner Gauss-Seidel steps were used in the FPGS smoother as this resulted in the fastest multigrid method. Running this model on the hump image revealed that it may have a problem recovering low intensity edges however reconstructions of the Lenna image looked reasonable. Other choices of π which lead to non-convex functionals are likely to prove more of a challenge.

Overall taking into account both the quality of the reconstructed image and the efficiency and robustness of iterative solvers, we favour model 3 with the nonlinear multigrid solver and a third order polynomial choice for p.

Chapter 7

Models and Solvers for Image Deblurring

Useful Section References: §2.3, §3.2,§3.5, §3.6, §4.2, §5.1, §5.2, §A.6.3
Main Reference Material: [20, 22, 24, 29, 33, 34, 36, 38, 55, 56, 71, 79, 83],[93]-[97]

Recall from Chapter 3 that the full TV deblurring and denoising problem involves the minimization of the functional

$$\int_\Omega \sqrt{|\nabla u|^2 + \beta} + 1/2(\mathcal{K}u - z)^2 dxdy. \tag{7.1}$$

and the corresponding Euler-Lagrange equation is

$$-\alpha\nabla.\left(\frac{\nabla u}{\sqrt{|\nabla u|^2 + \beta}}\right) + \mathcal{K}^*\mathcal{K}u = \mathcal{K}^*z. \tag{7.2}$$

Discretization of (7.2) leads to an equation of the form

$$\alpha L(\mathbf{u})\mathbf{u} + K^T K\mathbf{u} = K^T\mathbf{z}. \tag{7.3}$$

where L is an $N \times N$ sparse banded matrix and K is a Block Toeplitz matrix with Toeplitz blocks (see §7.2). The cost of inverting and multiplying by K can be reduced to $O(NlogN)$ by making use of fast fourier transform techniques. In general, the more noise there is the larger α should be and the more expensive it is to solve the nonlinear equation using the iterative methods currently available.

In this chapter details are given of our current work on deblurring, previously presented in [38]. Two current areas of investigation are detailed, both of which in some way try

and make use of iterative methods used for the pure denoising problem, within a deblurring and denoising context. The first is a new deblurring model which builds upon the two-step method presented in [38]. The model attempts in some way to decouple the denoising element of the image restoration from the deblurring element. At this stage work on this model is still ongoing and so no numerical results are presented here. The second area involves the development of two new iterative solvers for the total-variation deblurring problem. The new methods are shown to be somewhat complementary to the existing fixed point method, in that they perform best when α is large as opposed to the fixed point method, which performs best for small α. Consequently the new methods are most effective for problems involving low or moderate levels of blur and high noise levels.

The rest of the chapter is organized as follows, in section 7.1 some useful results on the Discrete Fourier Transform and Toeplitz and Circulant Matrices are reviewed, in §7.2 I outline the discretiztion of the Euler-Lagrange equation, in §7.3 some of the iterative methods currently used to solve the discrete Euler-Lagrange equation are surveyed, in §7.4 the two step deblurring model is introduced, in §7.5 the new iterative methods for the TV problem are presented, in §7.6 some numerical results relating to the methods of §7.5 are presented and finally I draw some conclusions and comment on future work in §7.7.

7.1 Toeplitz and Circulant Matrices and the DFT

This section gives a brief introduction to the Discrete Fourier transform, its fast implementation via the fast fourier transform and its connection to Toeplitz and circulant matrices.

7.1.1 The Discrete Fourier Transform

Definition 7.1.1 (The Discrete Fourier Transform)
Given a vector $\boldsymbol{f} = (f_0, f_1, ..., f_{n-1}) \in \mathbb{C}^n$ *the discrete fourier transform (DFT) of* $\boldsymbol{f}$ *is defined by*

$$[DFT\{\boldsymbol{f}\}]_k = \sum_{j=0}^{n-1} f_j e^{-i2\pi jk/n} \quad k = 0, 1, .., n-1$$

where $i = \sqrt{-1}$. *In matrix notation this is* $[DFT\{\mathbf{f}\}]_k = [\hat{F}_n \mathbf{f}]_k$ *where* $\hat{F}_n$ *has entries* $[\hat{F}_n]_{kj} = e^{-i2\pi jk/n}$.
The inverse DFT (IDFT) of $\boldsymbol{g} = (g_0, ..., g_{n-1}) \in {}^n$ *is defined by*

$$[IDFT\{\boldsymbol{g}\}]_k = \frac{1}{n} \sum_{j=0}^{n-1} g_j e^{i2\pi kj/n} = [\hat{F}_n^{-1} \mathbf{g}]_k$$

where $\hat{F}_n^{-1} = \frac{1}{n} \hat{F}_n^*$.

Definition 7.1.2 (Two-dimensional DFT)
The two-dimensional DFT (DFT2) of an array $f \in \mathbb{C}^{n\times m}$ *with entries* f_{ij} $0 \le i \le n-1$ $0 \le j \le m-1$ *is defined by*

$$[DFT2\{f\}]_{kl} = \sum_{r=0}^{n-1}\sum_{s=0}^{m-1} f_{rs}e^{-i2\pi(kr/n+ls/m)} \quad 0 \le k \le n-1, 0 \le l \le m-1$$

The inverse DFT2 (IDFT2) of an array $g \in \mathbb{C}^{n\times m}$ *is defined by*

$$[IDFT2\{g\}]_{kl} = \frac{1}{nm}\sum_{r=0}^{n-1}\sum_{s=0}^{m-1} g_{rs}e^{i2\pi(kr/n+ls/m)} \quad 0 \le k \le n-1, 0 \le l \le m-1$$

7.1.2 The Fast Fourier Transform

If the 1-dimensional DFT is implemented using standard matrix vector multiplication on a vector of size n then the computational cost is $O(n^2)$. The fast fourier transform (FFT) devised by Cooley and Tukey reduces this computational cost to $O(n \log n)$. The ideas behind the fast fourier transform algorithm are outlined below. Consider a vector $\mathbf{f} = (f_0, f_1, ..., f_{n-1}) \in \mathbb{C}^n$. Assume that n is a power of 2 and define $p = n/2$ and

$$\omega_n = e^{-i2\pi/n}. \tag{7.4}$$

The Discrete Fourier Transform of $\mathbf{f}$ can then be written as

$$[DFT\{\mathbf{f}\}]_k = \sum_{j=0}^{2p-1} f_j\omega_{2p}^{jk},\ k = 0, 1, \ldots, n-1. \tag{7.5}$$

This can be split into two sums

$$[DFT\{\mathbf{f}\}]_k = \sum_{j=0}^{p-1} f_{2j}\omega_{2p}^{2jk} + \sum_{j=0}^{p-1} f_{2j+1}\omega_{2p}^{(2j+1)k},\ k = 0, 1, .., n-1 \tag{7.6}$$

Now $\omega_{2p}^{2jk} = \omega_p^{jk}$ so

$$[DFT\{\mathbf{f}\}]_k = \sum_{j=0}^{p-1} f_{2j}\omega_p^{jk} + \sum_{j=0}^{p-1} f_{2j+1}\omega_p^{jk}\omega_{2p}^{k},\ k = 0, 1, .., n-1. \tag{7.7}$$

If we define $\mathbf{f}^{even} = (f_0, f_2, .., f_{n-2})$ and $\mathbf{f}^{odd} = (f_1, f_3, .., f_{n-1})$ then we see that for $0 \le k \le p-1$

$$[DFT\{\mathbf{f}\}]_k = [DFT\{\mathbf{f}^{even}\}]_k + \omega_{2p}^k[DFT\{\mathbf{f}^{odd}\}]_k\ . \tag{7.8}$$

Furthermore

$$\omega_p^{p+k} = e^{-i2\pi(p+k)/p} = \omega_p^k \tag{7.9}$$

and

$$\omega_{2p}^{p+k} = e^{-i2\pi(p+k)/2p} = -\omega_{2p}^k \tag{7.10}$$

so

$$[DFT\{\mathbf{f}\}]_{k+p} = [DFT\{\mathbf{f}^{even}\}]_k - \omega_{2p}^k [DFT\{\mathbf{f}^{odd}\}]_k \tag{7.11}$$

for $0 \leq k \leq p-1$. The discrete fourier transform of $\mathbf{f}$ can therefore be written in terms of the discrete fourier transforms of 2 vectors half its size

$$[DFT\{\mathbf{f}\}]_k = [DFT\{\mathbf{f}^{even}\}]_k + \omega_n^k [DFT\{\mathbf{f}^{odd}\}]_k \tag{7.12}$$

$$[DFT\{\mathbf{f}\}]_{k+n/2} = [DFT\{\mathbf{f}^{even}\}]_k - \omega_n^k [DFT\{\mathbf{f}^{odd}\}]_k \tag{7.13}$$

$$k = 0, 1, ..., n/2 - 1.$$

Since we are assuming that n is a power of 2 this process can be applied recursively until we have the Discrete Fourier Transform of $\mathbf{f}$ entirely in terms of discrete fourier transforms of single elements (vectors of size 1).

Computational Cost

Let $FFT(n)$ denote the number of operations required to evaluate $DFT\{\mathbf{f}\}$ where $\mathbf{f}$ is of size n, using the fast fourier transform and assume that $n = 2^q$. Given $DFT\{\mathbf{f}^{even}\}$ and $DFT\{\mathbf{f}^{odd}\}$ only $O(n/2)$ operations (the evaluation ω_n^k , the multiplication of ω_n^k by $[DFT\{\mathbf{f}^{odd}\}]_k$ and 2 additions for each $k = 0, 1, .., n/2$) are required to evaluate $DFT\{\mathbf{f}\}$ also $FFT(1) = 0$ therefore

$$\begin{aligned} FFT(n) &= O(n/2) + 2FFT(n/2) \\ &= O(n/2) + 2(O(n/4) + 2FFT(n/4)) \\ &= O(n/2) + 2(O(n/4) + 2(O(n/8) + FFT(n/8))) \\ &\vdots \\ &= O(qn/2) \end{aligned}$$

where $q = \log_2 n$.

A similar process to that used to evaluate the DFT can be used to evaluate IDFT. The two dimensional DFT can be evaluated by performing a 1-D DFT on each of the columns of f and then performing 1-D DFT's on the rows of the resulting array. Making use of the FFT the cost of DFT2 is $O(nm \log_2(nm))$. The FFT is very useful for efficiently computing matrix vector products of Circulant and Toeplitz matrices as we will see later.

7.1.3 Toeplitz and Circulant Matrices

Definition 7.1.3 (Convolution Product)
The convolution product of vectors $t = (t_{1-n}, ...t_0, t_1, ..., t_{n-1})$ *and* $\boldsymbol{f} \in \mathbb{C}^n$ *is defined by*

$$[t \star \boldsymbol{f}]_i = \sum_{j=0}^{n-1} t_{i-j} f_j \;\; i = 0, ..., n-1$$

The two-dimensional convolution product of an array t with components $t_{ij}, 0 \leq i \leq n-1, 0 \leq j \leq m-1$ *and an array* $f \in \mathbb{C}^{n \times m}$ *is defined by*

$$[t \star f]_{ij} = \sum_{k=0}^{n-1} \sum_{l=0}^{m-1} t_{i-k,j-l} f_{k,l} \; 1-n \leq i \leq n-1, 1-m \leq j \leq m-1$$

Definition 7.1.4 (n-periodic)
A discrete vector $\mathbf{t}$ *is called n-periodic if* $t_i = t_j$ *when* $i = j \bmod n$. *Given a vector* $\mathbf{t} = (t_0, ..., t_{n-1}) \in \mathbb{C}^n$ *the periodic extension of* $\mathbf{t}$ *of size* $2n-1$ *is the n-periodic vector* $t^{ext} = (t^{ext}_{1-n}, .., t^{ext}_0, ..., t^{ext}_{n-1})$ *for which* $t^{ext}_i = t_i \;, i = 0, .., n-1$

The following theorem relates the convolution product and the discrete Fourier Transform

Theorem 7.1.1 *If* $\mathbf{t}$ *and* $\mathbf{f} \in \mathbb{C}^n$ *and* t^{ext} *is the periodic extension of* $\mathbf{t}$ *of size* $2n-1$ *then*

$$t^{ext} \star \mathbf{f} = IDFT[DFT\{t\}. * DFT\{\mathbf{f}\}]. \tag{7.14}$$

Proof
Define $\omega = e^{-i2\pi/n}$ so that $e^{-i2\pi jk/n} = \omega^{jk}$. From Definition 7.1.1

$$DFT[\{\mathbf{t}^{ext} \star \mathbf{f}\}]_k = \sum_{j=0}^{n-1} \left([t^{ext} \star f]_j\right) \omega^{jk}. \tag{7.15}$$

From Definition 7.1.3 we see that this is

$$DFT[\{\mathbf{t}^{ext} \star \mathbf{f}\}]_k = \sum_{j=0}^{n-1} \left(\sum_{i=0}^{n-1} t^{ext}_{j-i} f_i \right) \omega^{jk}$$

$$= \sum_{i=0}^{n-1} f_i \left(\sum_{j=0}^{n-1} t^{ext}_{j-i} \omega^{jk} \right). \tag{7.16}$$

If we set $j = i + l$ so that $j = 0$ when $l = -i$ and $j = n-1$ when $l = n-1-i$ we have

$$DFT[\{\mathbf{t}^{ext} \star \mathbf{f}\}]_k = \sum_{i=0}^{n-1} f_i \left(\sum_{l=-i}^{n-1-i} t^{ext}_l \omega^{(l+i)k} \right)$$

$$= \sum_{i=0}^{n-1} f_i \omega^{ik} \sum_{l=-i}^{n-1-i} t^{ext}_l \omega^{lk}. \tag{7.17}$$

Now $\mathbf{t}^{ext}$ is n-periodic so $t^{ext}_{-p} = t^{ext}_{n-p}$ for $p = 1, .., n-1$. Also if k is an integer

$$\omega^{(l+n)k} = e^{-i2\pi(l+n)k/n} = e^{-i2\pi lk/n} e^{-i2\pi k} = e^{-i2\pi lk/n} = \omega^{lk} \tag{7.18}$$

so

$$\sum_{l=-i}^{n-1-i} t^{ext}_l \omega^{lk} = \sum_{l=0}^{n-1-i} t^{ext}_l \omega^{lk} + \sum_{l=n-i}^{n-1} t^{ext}_l \omega^{lk}$$

$$= \sum_{l=0}^{n-1} t_l^{ext} \omega^{lk} = [DFT\{\mathbf{t}\}]_k \text{ for all } i \in (0,1,...,n-1) \tag{7.19}$$

therefore

$$DFT[\{\mathbf{t}^{ext} \star \mathbf{f}\}]_k = \left(\sum_{i=0}^{n-1} f_i \omega^{ik} \right) [DFT\{\mathbf{t}\}]_k = [DFT\{\mathbf{f}\}]_k [DFT\{\mathbf{t}\}]_k. \tag{7.20}$$

and the theorem is proved. ∎

A similar result in two dimensions shows that

$$t^{ext} \star f = IDFT2[DFT2\{t\}. * DFT2\{f\}], \tag{7.21}$$

where t^{ext} is the (n,m)-periodic extension of size $(2n-1) \times (2m-1)$ of the $n \times m$ array t.

Definition 7.1.5 (Toeplitz Matrix)
An $n \times n$ matrix T is said to be toeplitz if it has the form

$$T = \begin{bmatrix} t_0 & t_{-1} & . & . & t_{2-n} & t_{1-n} \\ t_1 & t_0 & t_{-1} & & & t_{2-n} \\ . & t_1 & t_0 & & & . \\ . & & . & . & & \\ t_{n-2} & & & . & . & t_{-1} \\ t_{n-1} & t_{n-2} & . & . & t_1 & t_0 \end{bmatrix}$$

i.e it is constant along its diagonals.

A Toeplitz matrix T can be uniquely determined by its first row and column only. The $2n-1$ vector $\mathbf{t} = (t_{1-n}, t_{2-n}, .., t_{-1}, t_0, t_1, ..., t_{n-1})^T$ is therefore defined as the *generator* of T, this is denoted $T = \mathbf{toeplitz(t)}$. If $\mathbf{f} = (f_0, f_1, .., f_{n-1})^T \in \mathbb{C}^n$ then

$$[T\mathbf{f}]_i = \sum_{j=0}^{n-1} t_{i-j} f_j = [\mathbf{t} \star \mathbf{f}]_i. \tag{7.22}$$

Definition 7.1.6 (Circulant Matrix)
An $n \times n$ matrix is said to be circulant if it has the form

$$C = \begin{bmatrix} c_0 & c_{n-1} & . & . & c_2 & c_1 \\ c_1 & c_0 & c_{n-1} & & & c_2 \\ . & c_1 & c_0 & & & . \\ . & & . & . & & \\ c_{n-2} & & & . & . & c_{n-1} \\ c_{n-1} & c_{n-2} & . & . & c_1 & c_0 \end{bmatrix}$$

A circulant matrix is a matrix which is Toeplitz and which has the added property that each column is the previous column downshifted by 1 with the last entry of the previous column wrapping around to become the first entry. A circulant matrix can therefore be determined by its first column only. We define $\mathbf{c} = (c_0, c_1, ..., c_{n-1})^T$ as the generator of C and write $C = \mathbf{circulant(c)}$. Of course since C is Toeplitz we can also define $C = \mathbf{toeplitz}(\mathbf{c}^{ext})$ where $\mathbf{c}^{ext} = (c_1, ..., c_{n-1}, c_0, c_1, ..., c_{n-1})^T$ is the n-periodic extension of $\mathbf{c}$ of size $2n-1$. This means that for a vector $\mathbf{f} = (f_0, f_1, .., f_{n-1})^T \in \mathbb{C}^n$

$$C\mathbf{f} = \mathbf{c}^{ext} \star \mathbf{f}. \tag{7.23}$$

From theorem 7.1.1 we see that this is

$$C\mathbf{f} = IDFT\{DFT\{\mathbf{c}\}. * DFT\{\mathbf{f}\}\}, \tag{7.24}$$

which can be done using fast fourier transforms. This gives us a computationally efficient way to compute $C\mathbf{f}$ which requires storage of the first column of C only. If we rewrite equation (7.24) in matrix form we have

$$\begin{aligned} C\mathbf{f} &= \hat{F}_n^{-1}(\hat{F}_n\mathbf{c}. * \hat{F}_n\mathbf{f}) \\ &= \hat{F}_n^{-1} diag(\hat{F}_n\mathbf{c})\hat{F}_n\mathbf{f} \end{aligned}, \tag{7.25}$$

which implies that

$$C = \hat{F}_n^{-1} diag(DFT\{\mathbf{c}\})\hat{F}_n = F_n^* diag(DFT\{\mathbf{c}\})F_n, \tag{7.26}$$

where $F_n = \frac{1}{\sqrt{n}}\hat{F}_n$ is a unitary matrix known as the fourier matrix. Thus any circulant matrix can be diagonalized by the Fourier matrix F and the eigenvectors of all circulant matrices of size n are the same. Furthermore the eigenvalues of a circulant matrix are the entries of $DFT\{\mathbf{c}\}$.

An $m \times m$ block matrix with $n \times n$ blocks is said to be Block Toeplitz with Toeplitz blocks (BTTB) if each of its blocks is Toeplitz and it has a Toeplitz structure at the block level as well. A BTTB matrix has generator $t = (\mathbf{t}_{1-m}, .., \mathbf{t}_0, .., \mathbf{t}_{m-1})$ where $\mathbf{t}_j$ is the generator of the block T_j. If $\mathbf{f} \in \mathbb{C}^{nm}$ and T is an $nm \times nm$ BTTB matrix, then

$$T\mathbf{f} = vec(t \star array(\mathbf{f})), \tag{7.27}$$

where *vec* stacks an $n \times m$ array columnwise into a vector and *array* is its inverse. A Block Circulant matrix with Circulant blocks (BCCB) is defined analogously and has generator $c = (\mathbf{c}_0, .., \mathbf{c}_{m-1})$, stacking this array columnwise into a vector gives the first column of the BCCB matrix. Analogous to the circulant case the matrix vector product of a BCCB matrix C and a vector $\mathbf{f}$ of size nm can be written as

$$C\mathbf{f} = vec(IDFT2(DFT2(c). * DFT2(array(\mathbf{f})))), \tag{7.28}$$

where .* denotes componentwise multiplication. This relationship allows us to calculate matrix vector products of C in $O(nm\log(nm))$ operations using FFT's with storage of the generator of C only. Note also that the cost of inverting a BCCB matrix is also $O(nm\log(nm))$.

If C_a and C_b are two BCCB matrices with generators a and b, then clearly $C_a + C_b$ is a BCCB matrix with generator $a + b$, in addition $C_d = C_a C_b$ is also a BCCB matricx, furthermore the first column of C_d i.e $vec(d)$ is $C_a vec(b) = vec(IDFT2\{DFT2\{a\}.*DFT2\{b\}\})$ so $d = IDFT2\{DFT2\{a\}.*DFT2\{b\}\}$.

To efficiently compute matrix vector products involving BTTB matrices a process known as circulant extension, in which the BTTB matrix T is embedded in a larger BCCB matrix is used. Algorithm 22 outlines how to compute $T\mathbf{f}$ if T is BTTB with generator t.

Algorithm 22 BTTB Matrix Vector Products Using Circulant Extension

1. Extend t to a $2n \times 2m$ array $\tilde{t}$ by adding a top row and left column of zeros.

2. Partition $\tilde{t}$ into four $n \times m$ subblocks and then reorder the subblocks to form the generator for a BCCB matrix $\tilde{C}$

$$\tilde{t} = \begin{bmatrix} \tilde{t}_{11} & \tilde{t}_{12} \\ \tilde{t}_{21} & \tilde{t}_{22} \end{bmatrix} \qquad \tilde{c} = \begin{bmatrix} \tilde{t}_{22} & \tilde{t}_{21} \\ \tilde{t}_{12} & \tilde{t}_{11} \end{bmatrix}$$

3. Take $f = array(\mathbf{f})$ and embed it in a $2n \times 2m$ array $\tilde{f}$

$$\tilde{f} = \begin{bmatrix} f & 0 \\ 0 & 0 \end{bmatrix}$$

4. Compute

$$g = IDFT2(DFT2(\tilde{f}).*DFT2(\tilde{c}))$$

5. Take the leading $n \times m$ subblock, g_{11}, of the array g. $T\mathbf{f}$ is then given by

$$T\mathbf{f} = vec(g_{11})$$

7.1.4 T. Chans Circulant Approximation

For a general $n \times n$ matrix A, T. Chans circulant approximation [22] $c(A)$ is the $n \times n$ circulant matrix C which minimizes $\|C - A\|_F$, where $\|.\|_F$ denotes the Frobenius norm defined by $\|A\|_F = \sqrt{\sum_{i,j} |a_{i,j}|^2}$. The generator of $c(A)$ is given by the formula

$$c_l = \frac{1}{n} \sum_{j-k=l(mod n)} a_{j,k}, \quad l = 0, .., n-1. \tag{7.29}$$

The c_l are just the average of the diagonals of A with the diagonals extended to length n by a wrap around.

Example

For the 4 x 4 matrix shown below the entries which are used to calculate c_1 are shown in bold.

$$\begin{pmatrix} 1 & 3 & 6 & 8 \\ \mathbf{2} & 7 & 5 & 4 \\ 3 & \mathbf{1} & 5 & 4 \\ 2 & 6 & \mathbf{5} & 3 \end{pmatrix}$$

c_1 is therefore given by $\frac{1}{4}(2+1+5+8) = 4$. ∎

For the specific case when A is an $n \times n$ toeplitz matrix the generator for T.Chans circulant approximation $c(A)$ can be defined from the generator of A, $\mathbf{a} = (a_{1-n}, .., a_0, ..., a_{n-1})^T$ by

$$c_j = \frac{(n-j)a_j + ja_{j-n}}{n} \quad j = 0, ..., n-1. \tag{7.30}$$

$c(A)$ can be therefore be computed in $O(n)$ operations. Similarly if A is sparse with just a few nonzero diagonals $c(A)$ can also be computed in $O(n)$ operations.

A BCCB approximation $c_2(A)$ of a block matrix A can be computed by first approximating each block in A by a circulant matrix as above and then applying the same approach at the block level.

7.2 Discretization

In this section I outline the discretization of the Euler-Lagrange equation for the TV deblurring problem (7.2).

The diffusion term in the Euler-Lagrange equation is discretized in exactly the same way as in the denoising problem, so using the usual matrix notation we have $\alpha_h BE(\mathbf{u}_h)^{-1}B\mathbf{u}_h$ (as defined in §3.5.1) which I will denote $L(\mathbf{u}_h)\mathbf{u}_h$, all that remains is to discretize the term $\mathcal{K}^*(\mathcal{K}u - z)$. To avoid confusion with the kernel function, the grid spacing in the y-direction will be denoted by p rather than the usual k. By midpoint quadrature we have

$$(\mathcal{K}u)_{ij} = \sum_{s=1}^{m}\sum_{r=1}^{n} k(x_i - x_r, y_j - y_s)u_{rs}hp = [\hat{k} \star u]_{ij}, \tag{7.31}$$

where $\hat{k} = hp(\mathbf{k}_{1-m}, .., \mathbf{k}_0, .., \mathbf{k}_{m-1})$ and $\mathbf{k}_j = (k((1-n)h, jp), .., k(0, jp), .., k((n-1)h, jp))^T$. From (7.28) the discrete version of $\mathcal{K}u$ is $K\mathbf{u}$ where K is a BTTB matrix with generator $\hat{k}$. The discrete version of the adjoint operator $\mathcal{K}^*$ is just K^T. Hence we have the discrete Euler-Lagrange equation

$$\alpha L(\mathbf{u})\mathbf{u} + K^T K\mathbf{u} = K^T\mathbf{z}. \tag{7.32}$$

In practical applications it is usually the case that we are given a $n \times m$ 'PSF array' H which results from imaging a point source, assumed to be located at the central pixel $(n/2+1, m/2+1)$ [55] (rather than having an explicit formula for k). The array H contains all the nonzero entries of $\hat{k}$ and is the central $n \times m$ block of $\tilde{k}$, where $\tilde{k}$ is the $2n \times 2m$ array which results from adding a top row and left column of zeros to $\hat{k}$. Given H we can replace steps 1 and 2 in algorithm 22 with

1. Partition H into four $n/2 \times m/2$ subblocks

$$H = \begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix} \tag{7.33}$$

   note that $\hat{k}_{0,0}$ is the top left entry of $H_{2,2}$.

2. Form the $2n \times 2m$ BCCB generator array

$$\tilde{c} = \begin{bmatrix} H_{22} & 0 & H_{21} \\ 0 & 0 & 0 \\ H_{12} & 0 & H_{11} \end{bmatrix} \tag{7.34}$$

7.3 Solving The Euler-Lagrange Equation

Several of the iterative methods used to solve the denoising problem (§3.6), can also be applied in the deblurring case.

The simplest approach is just to use the artificial time-marching method i.e find the steady state of

$$u_t = \alpha \nabla . \left(\frac{\nabla u}{\sqrt{|\nabla u|^2 + \beta}} \right) - \mathcal{K}^*(\mathcal{K}u + z) \tag{7.35}$$

using an explicit time-marching method. The update of $\mathbf{u}$ is done as follows

$$\mathbf{u}^{r+1} = \mathbf{u}^r - \Delta t \left(L(\mathbf{u}^r)\mathbf{u}^r + K^T(K\mathbf{u}^r - \mathbf{z}) \right). \tag{7.36}$$

As in the denoising case the time step Δt has to be small due to stability constraints, resulting in very slow convergence, which can be somewhat improved by multiplying the right hand side of (7.35) by $|\nabla u|$. Each step of this method simply requires evaluation of the entries of $L(\mathbf{u}^r)$ plus matrix vector multiplications by L, K and K^T.

The 'lagged diffusivity' fixed point method of Vogel and Oman can also be used in the deblurring case. In this method the following linear equation is solved to update $\mathbf{u}$ on each step.

$$\alpha L(\mathbf{u}^r)\mathbf{u}^{r+1} + K^T K \mathbf{u}^{r+1} = K^T \mathbf{z}. \tag{7.37}$$

The linear system is symmetric positive definite and is usually solved using the preconditioned conjugate gradient method (PCG), the main cost of each step of the PCG method is a matrix

vector multiply by $\alpha L(\mathbf{u}^r) + K^T K$ and the inversion of a preconditioner M. One option would be to take $M = c_2(\alpha L(\mathbf{u}^r)) + c_2(K)^T c_2(K)$ where $c_2(A)$ denotes T.Chans BCCB approximation of A. For an $n \times m$ image the construction cost of the preconditioner will be $O(nm)$ and its inversion using FFT's will have cost $O(nm \log nm)$. A similar approach with similar costs, using cosine transforms and optimal cosine transform preconditioners is presented in [24]. These cosine transform preconditioners give better approximations to the elliptic operator. Another option proposed by Vogel and Oman [95] is the product preconditioner $M = \gamma^{-1}(\tilde{K}^T\tilde{K} + \gamma I)^{1/2}(\gamma I + \alpha L)(\tilde{K}^T\tilde{K} + \gamma I)^{1/2}$, where $\tilde{K}$ is a circulant approximation to K. The aim here is to 'split up' the preconditioning of the elliptic and the convolution terms. $(\tilde{K}^T\tilde{K} + \gamma I)^{1/2}$ is inverted using FFT's and $(\gamma I + \alpha L)$ is inverted using PCG with multigrid preconditioner. In [24] 'diagonal scaling' of the cosine transform and product preconditioners is employed to try and better capture the large variations in the $D_{i,j}$ terms within the elliptic operator. Two level preconditioners are employed in [79] by Riley. Finally I mention an attempt in [20] to use a linear multigrid solver to solve the linear equation. This work only looked at the 1-dimensional (signal processing) case and an efficient implementation of the proposed method could not be obtained.

The primal-dual Newton method also applies in the deblurring case, the system of equations to be solved becomes

$$\begin{aligned} -\alpha\nabla.w + \mathcal{K}^*(\mathcal{K}u - z) &= 0 \\ w\sqrt{|\nabla u|^2 + \beta} - \nabla u &= 0 \\ |w(x,y)| &\leq 1 \forall (x,y) \end{aligned} \quad . \tag{7.38}$$

The inner solve in this case will involve solving something of the form

$$(H(\mathbf{u}_h) + K^T K)\mathbf{du}_h = \mathbf{g}(\mathbf{u}_h). \tag{7.39}$$

where $H(\mathbf{u}_h)$ is a discretization of

$$\left[-\alpha\nabla.\left(\frac{1}{\sqrt{|\nabla u|^2+\beta}}\left(I - \frac{1}{2}\frac{w\nabla u^T + \nabla u w^T}{\sqrt{|\nabla u|^2+\beta}}\right)\nabla\right) + I\right]. \tag{7.40}$$

In [33] PCG with cosine transform preconditioners is suggested for the inner solve.

7.4 An Alternative Two-Step Deblurring Model

In [38] we proposed the following two-step method for denoising and deblurring an image.

1. First approximate $\mathcal{K}u_0$ (where u_0 is the true image) by solving the denoising problem

$$\min_w \int_\Omega \alpha_1\sqrt{|\nabla w|^2 + \beta} + 1/2(w - z)^2 dxdy \tag{7.41}$$

which has Euler-Lagrange equation

$$-\alpha_1\nabla.\left(\frac{\nabla w}{\sqrt{|\nabla w|^2+\beta}}\right) + w = z \tag{7.42}$$

2. Approximate u_0 by solving the deblurring problem

$$\min_u \int_\Omega \alpha_2 \sqrt{|\nabla u|^2 + \beta} + 1/2(\mathcal{K}u - w)^2 dxdy \tag{7.43}$$

which has Euler-Lagrange equation

$$-\alpha_2 \nabla \cdot \left(\frac{\nabla u}{\sqrt{|\nabla u|^2 + \beta}} \right) + \mathcal{K}^*(\mathcal{K}u - w) \tag{7.44}$$

The idea is to try and decouple the denoising from the deblurring. The cost of the pure denoising problem (7.41) is relatively cheap and the value of α_2 needed in (7.43) should be less than the equivalent value of α needed in (7.1) since some of the noise has been removed by the first step, reducing the cost of solving the resulting Euler-Lagrange equation. We proposed using nonlinear multigrid to solve the discretization of (7.42), for the solution of the discretization of (7.44) fixed point or primal-dual Newton can be used.

To try and make this approach more formal we have recently considered the following minimization problem, which is not equivalent to the above approach but is in a similar vein.

$$\min_{u,w} J(u,w), \text{ where } J(u,w) = \int_\Omega \alpha_2 |\nabla u|_\beta + \alpha_1 |\nabla w|_\beta + 1/2(w - z)^2 + \gamma/2(\mathcal{K}u - w)^2 dxdy \tag{7.45}$$

Here u will be an approximation to the true image u_0 and w an approximation to $\mathcal{K}u_0$. The minimization problem is solved via an iterative procedure in which u is fixed and the nonlinear PDE

$$-\alpha_1 \nabla \cdot \left(\frac{\nabla w}{|\nabla w|_\beta} \right) + (1 + \gamma)w - (z + \mathcal{K}u) = 0 \tag{7.46}$$

solved to update w, the new value of w held fixed and

$$-\frac{\alpha_2}{\gamma} \nabla \cdot \left(\frac{\nabla u}{|\nabla u|_\beta} \right) + \mathcal{K}^*(\mathcal{K}u - w) = 0 \tag{7.47}$$

solved to update u. The process is repeated until $|J(u^{old}, w^{old}) - J(u^{new}, w^{new})|$ is less than some specified tolerance. Once again nonlinear multigrid (or other solvers used for the TV denoising problem) can be used in the first step, and iterative solvers for the TV deblurring problem used in the second step. Obviously the solve on each step can be initialized using the approximation from the previous step and typically only a small reduction in the nonlinear residual on each inner step is required, to produce the most efficient method.

At this stage we are still in the process of investigating the effect of the various parameters in the model, on both image quality and overall efficiency of the method. Some preliminary results suggest that reconstructed images of similar quality to the result of TV deblurring with fixed point solver can be achieved in comparable cpu time, particularly for problems with moderate blurring and moderate or high levels of noise. We have found that taking γ relatively large (i.e requiring that $\mathcal{K}u$ is a good match to w) generally improves both image

quality and overall computational time. Clearly from the above equations, the larger γ is the easier it is to solve both (7.46) and (7.47), the disadvantage is that more outer steps of the iteration are required.

We do not present here any numerical results or draw any firm conclusions regarding this model as further testing is required fisrt, however we believe that there is potentially an advantage in decoupling the pure denoising from the deblurring and exploiting efficient solvers for the resulting equations.

7.5 New Solvers for TV Deblurring

In the following we propose two alternative iterative solvers for the discrete Euler-Lagrange equation of the TV deblurring problem. In the first method we propose updating $\mathbf{u}$ on each step by solving the linear equation

$$\sigma \mathbf{u}^{r+1} + \alpha L(\mathbf{u}^r)\mathbf{u}^{r+1} = K^T\mathbf{z} - K^TK\mathbf{u}^r + \sigma \mathbf{u}^r. \tag{7.48}$$

In the second method we update $\mathbf{u}$ by solving the nonlinear equation

$$\sigma \mathbf{u}^{r+1} + \alpha L(\mathbf{u}^{r+1})\mathbf{u}^{r+1} = K^T\mathbf{z} - K^TK\mathbf{u}^r + \sigma \mathbf{u}^r. \tag{7.49}$$

The idea here is to avoid the inversion $L + K^TK$ via PCG needed on each step of the fixed point method (7.37) by moving the $K^TK\mathbf{u}$ term over to the right hand side, one matrix vector multiply by K^TK is then needed on each outer step instead. We also add a $\sigma\mathbf{u}$ term to both sides to stabilize the method. If the K^TK term is dominant this approach may not be sensible (this is confirmed by experiment in the next section) but when α is relatively large there may be some benefit.

In the first method we can use any of the linear solvers used in the fixed point method for the pure denoising problem e.g linear multigrid or preconditioned conjugate gradient method to solve the linear equation on each step. In the second method we propose to use a nonlinear multigrid method, like the one used in Chapter 4 to solve the nonlinear equation on each step. From now on we refer to (7.48) as method 1 and (7.49) as method 2.

Remark 7.5.1 *A method similar to (7.48) was proposed independently by Chang et al [36], in their method an additional diagonal term $D\mathbf{u}$ is added to both sides, where D is a diagonal matrix with the same diagonal as K^TK, this can be computed efficiently using the BTTB structure of K, algebraic multigrid is used as the linear solver. In our experience the addition of this diagonal term has very little effect on the convergence properties of the method.*

Numerical results relating to the implementation of these two new methods are given in the next section.

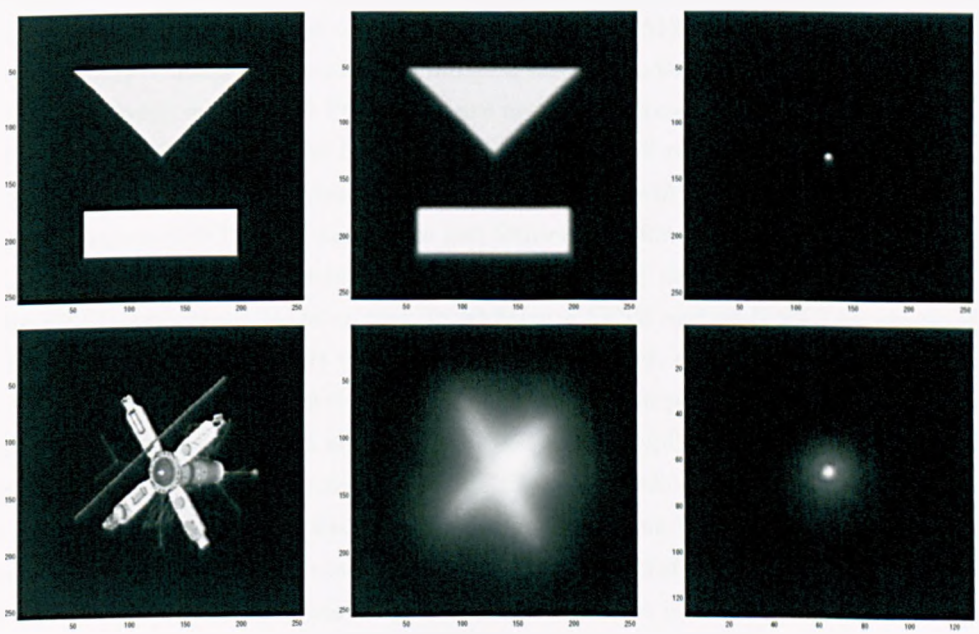
7.6 Numerical Results

Tests in this section are run on two blurred images (size 256×256), the first image is the triangle with a relatively low level of blur, the second image is the satellite image with a higher level of blur. In the first case we generate an $n \times n$ PSF array using the formula

$$H_{i,j} = ce^{0.1(|i-i'|+|j-j'|)}, \tag{7.50}$$

where (i', j') is the central pixel $(n/2+1, n/2+1)$ and c is a normalization constant. In the second case the PSF array is given as part of the data set. Various levels of noise will be tested. Figure 7.6 shows the true images the PSF arrays and the resulting blurred images (no noise is added so that the blurring effect can be seen clearly).

Figure 7.1: True image (left), blurred image (center) and PSF array (right) for triangle image (top) and satellite image (bottom)



7.6.1 Choice of σ

Before coming to the main tests I first mention the significance of the choice of the parameter σ in (7.48) and (7.49). In every test I have performed so far both method 1 and method 2 converge provided $\sigma \geq 0.5$, taking any smaller value of σ leads to a divergent method. In addition the minimum number of steps for convergence appears to be achieved with the smallest possible value of σ i.e $\sigma = 0.5$, in all tests below I have used $\sigma = 0.5$, except in one

case in which a slightly larger value of σ led to better convergence properties of the inner nonlinear solver and thus an overall faster method. The smaller α is, the more significant the choice of σ appears to be.

7.6.2 Cost Analysis

In the next subsection I will compare the new methods with the fixed point method, below I analyse some of the costs involved in each method. For ease of implementation I use PCG with circulant preconditioner as the linear solver in the fixed point method although other preconditioning methods may potentially give better convergence results, in method 1 I am using linear multigrid (c.f §5.2.2) for the inner linear solver and in method 2, nonlinear multigrid (c.f. Chapter 4) for the inner nonlinear solver.

In Table 7.1 the cost in cpu time and flops of one PCG step with circulant preconditioner, one linear multigrid (LMG) step with 2 pre and 2 post correction steps of gauss-seidel relaxation and one nonlinear multigrid (NLMG) step with 5 pre and 5 post-correction smoothing steps is shown in the 256×256 case and also in the 512×512 case. Typically in the results seen next, only 1 linear and nonlinear multigrid step is needed per outer step of methods 1 and 2 respectively, while 20-40 PCG steps are needed per fixed point step.

Each inner PCG step within the fixed point method will require a matrix multiplication by the matrix K^TK, multiplying by each of K and K^T will involve performing a 2D fast fourier transform (FFT2) and an inverse fast fourier transform (IFFT2) on an array of size $2n \times 2m$ ($DFT2\{\tilde{c}\}$ in Algorithm 22 can be computed at the beginning of the algorithm (using FFT2) and stored for later use), in addition a FFT2 and an IFFT2 on arrays of size $n \times m$ will be required to invert the circulant preconditioner, two FFT2 and one IFFT2 will also be required in constructing the generator of the circulant preconditioner at the beginning of the PCG steps. In method 1 and method 2 just one multiplication by K^TK is required on each outer step. The cost of performing an FFT2/IFFT2 pair on vectors of size $2N$ and N is also shown in the table. Note that as the size of the problem increases we expect that these costs will grow faster than the cost of performing a multigrid step ($O(N \log N)$ versus $O(N)$), this can be seen in the flop costs in the table. What is also interesting is that although the cost in cputime of performing the FFT2/IFFT2 pair (using MATLAB's fft function) is low, the cost in cputime of performing FFT's on $2n \times 2m$ array's has increased ten fold as the size of the problem increases from 256^2 to 512^2. We don't know why this is, at this stage, but it does suggest a possible increasing advantage (in terms of computational time) as the size of the problem increases, of methods which limit the number FFT's required.

Remark 7.6.1 *The costs associated with implementing other preconditioning techniques, such as optimal cosine preconditioners and the product preconditioner will be similar to the circulant case in terms of the number of FFT's required (an additional inner solve on $\gamma I + \alpha L$*

Table 7.1: Costs associated with implementing fixed point method and methods 1 and 2

N	256^2		512^2	
	Flops	cpu	Flops	cpu
LMG(2/2)	1.28×10^7	1.2	5.11×10^7	5.5
NLMG(5/5)	8.71×10^7	7.2	3.49×10^8	29.4
PCG	1.28×10^8	0.55	5.54×10^8	5.2
FFT2/IFFT2 (2N)	5.11×10^7	0.22	2.22×10^8	2.3
FFT2/IFFT2 (N)	1.29×10^7	0.052	5.69×10^7	0.22

is required in the product preconditioner) . Other preconditioners may of course reduce the overall number of PCG steps required per step.

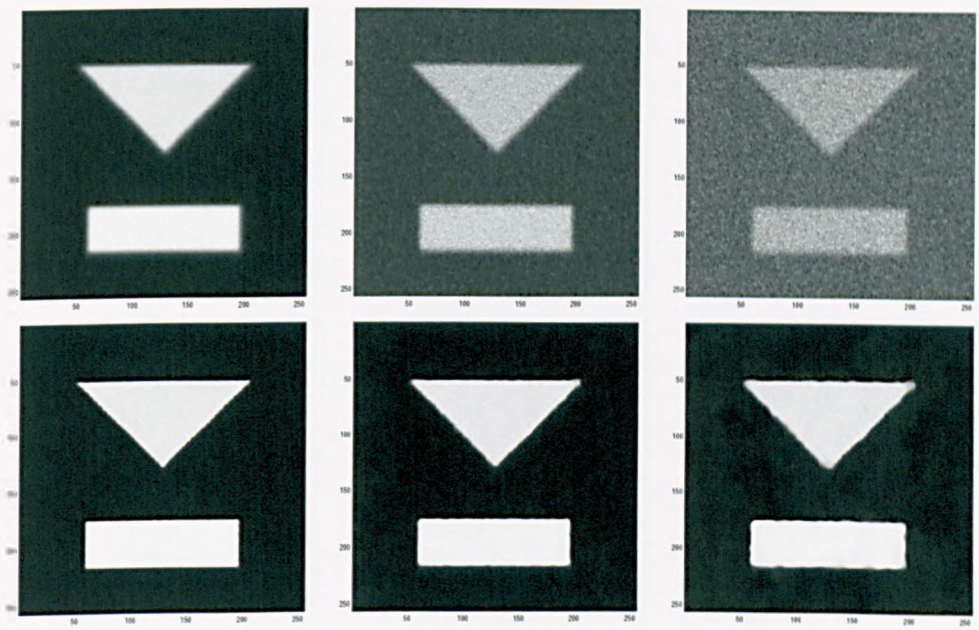
7.6.3 Comparison of new Methods with the Fixed Point Method

In the following we compare our two alternative solvers (7.48) and (7.49) with the fixed point method. We test method 1, method 2 and the fixed point method on the blurred triangle image seen previously with 3 different levels of noise (Figure 7.2), very low noise (snr$\approx$ 1000), medium noise (snr$\approx$ 10) and relatively high noise (snr$\approx$ 2) and on the blurred satellite image with similar levels of noise (Figure 7.3). Suitable choices of α_h in the case of the triangle are 0.1, 5 and 20 for the low, medium and high noise levels respectively. In the case of the satellite image where the blurring is heavier, the best results, in terms of image quality are achieved for much smaller values of α_h. In this case α_h is taken to be 10^{-3}, 0.2 and 0.5 for the low, medium and high noise levels respectively. We run each of the methods until the outer nonlinear residual has been reduced by a factor of 10^{-4}. In the fixed point method we use PCG with circulant preconditioner as the linear solver and stop the iterations when the linear residual has been reduced by a factor of 0.1. In method 1 we use a linear multigrid method with 2 pre and 2 post correction smoothing steps and stop the iterations when the linear residual has been reduced by a factor of 0.1. In method 2 we use nonlinear multigrid with 5 pre and 5 post-correction smoothing steps of the FPGS smoother as the nonlinear solver on each step, we stop the inner iterations when the residual of the inner nonlinear equation has been reduced by a factor of 0.1. In all cases we take $\beta_h = 10^{-2}$ and take the observed image z as initial guess, in both method 1 and method 2 we take $\sigma = 0.5$, except in the case of the triangle image with heaviest noise, in which σ is taken as 0.7 in order that the nonlinear multigrid method converges without the need to increase the number of smoothing steps.

The number of outer steps required for convergence, the cpu time in seconds and the average number of inner steps (ais) needed on each outer step for each method run on the

triangle image is shown in Table 7.2, the results for the satellite image are shown in table 7.3.

Figure 7.2: Blurred triangle with 3 levels of noise, low noise (left) medium noise (centre) and high noise (right), with images recovered using total variation deblurring (second row)



We see that as the noise level and hence the value of α needed increases so does the number of steps required for convergence of the fixed point method, on the other hand as α increases the number of steps required for convergence of method 1 and method 2 reduces suggesting that these approaches and the fixed point approach may be complementary to each other.

The fixed point method performs best for the case of the heavily blurred satellite image, with low noise level, with just four outer steps required for convergence, the largest number of outer steps is required for the triangle image with high noise level, although we note that as the level of noise is increased, the increase in fixed point steps is more dramatic in the case of the satellite image. Generally the number of inner PCG steps stays fairly stable although in the case of the satellite image the number of PCG steps required in the low noise case is more than half that required at the other two noise levels.

We see that the new methods perform best for the triangle image with the medium and high levels of noise. In both cases the number of outer steps decreases as the level of noise increases and the decrease is larger in the case of the triangle. In the case of the triangle the decrease in the number of steps in method 2 is at least twice the decrease in the number of

Figure 7.3: Blurred satellite with 3 levels of noise, low noise (left) medium noise (centre) and high noise (right), with images recovered using total variation deblurring (second row)

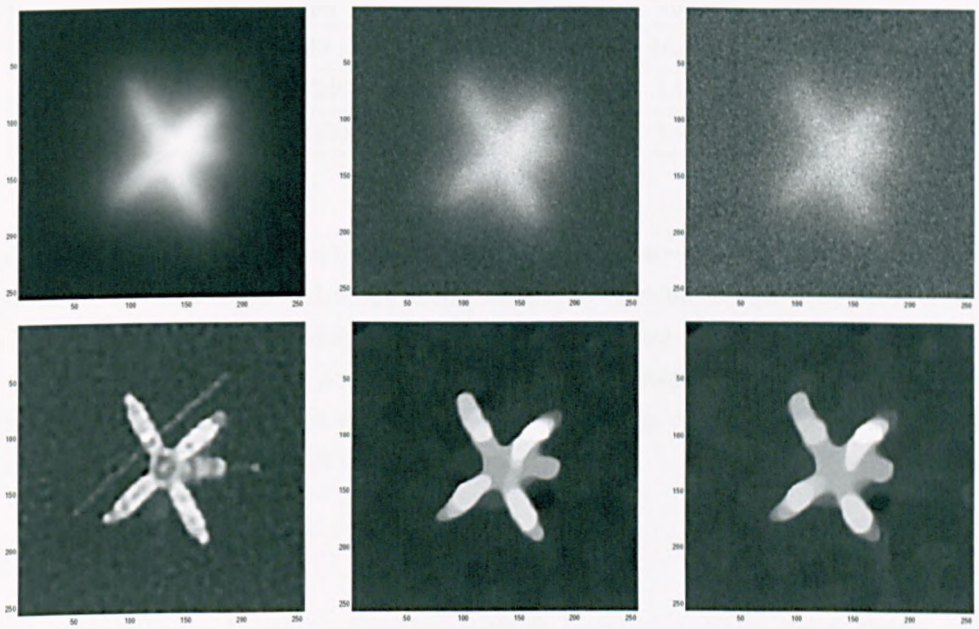


Table 7.2: Comparison of Method 1, Method 2 and Fixed Point method for blurred triangle with 3 levels of noise.

Noise level	Low			Medium			High		
α_h	0.1			5			20		
	steps	cpu	ais	steps	cpu	ais	steps	cpu	ais
Fixed Point	91	1343	26.0	347	5940	31.1	386	6575	31.0
Method 1	1614	3763	1.0	521	1244	1.1	452	1227	1.2
Method 2	1635	13616	1.0	264	2843	1.3	87	2601	4.0

Table 7.3: Comparison of Method 1, Method 2 and Fixed Point method for blurred satellite with 3 levels of noise.

Noise level	Low			Medium			High		
α_h	10^{-3}			0.2			0.5		
	steps	cpu	ais	steps	cpu	ais	steps	cpu	ais
Fixed Point	4	48	17.8	117	2828	46.1	222	4758	40.7
Method 1	4419	10249	1.0	2725	6569	1.0	2149	5170	1.1
Method 2	4418	37409	1.0	2671	22567	1.0	2009	16561	1.0

steps for method 1, however in the satellite case the decrease is roughly the same for both methods. In all cases method 1 outperforms method 2 in terms of cpu time. In general just one inner multigrid step is required on each step of method 1 and method 2, however for the triangle image with heavy noise 4 nonlinear multigrid steps are required on each step of method 2, this increase in inner steps negates any advantage over method 1 gained from the faster decrease in the number of outer steps.

7.7 Conclusion

I have presented two alternative iterative methods for total variation image deblurring. In contrast to the fixed point method inversion of a linear system $L + K^T K$ where K is BTTB is not required on each step, instead either a linear or a nonlinear inner solve using methods employed for the pure denoising problem is required. The new methods perform best when α is large, as opposed to the fixed point method which performs best for small α, this makes them most suitable for deblurring problems in which high levels of noise are combined with moderate levels of blur. Testing suggests that the linear variant (method 1) is overall more efficient than the nonlinear variant (method 2). Further testing of the methods against fixed point with other preconditioners and primal-dual Newton are planned in the future.

Also presented was an alternative model for image deblurring, which attempted to decouple the denoising from the deblurring. Work on analysing and implementing this new model is still ongoing.

Chapter 8

Future Work

In this chapter I briefly discuss some possible future research directions.

Starting with the TV denoising problem, there are several future areas of research that arise from our work on nonlinear multigrid (chapter 4) and AMG-R within the fixed point method (chapter 5). The main area of improvement for the nonlinear multigrid method is its performance with respect to small values of β. To try and combat this we would like to investigate the possibility of using more accurate interpolation operators for transferring the error, possibly like those considered for use in cascadic multigrid by Ohesen in [74]. Another possibility would be to use some sort of matrix dependant interpolation operator for accurately coarsening the linearized (fixed point type) operator and use this within the multilevel nonlinear method of Yavneh and Dardyk (see [103] for details and §2.6.10 for a brief introduction). As regards the fixed point with AMG-R method, there is as mentioned earlier a need to improve our implementation in MATLAB of the AMG method, or possibly, to implement in another programming language (note that predictions for cost reduction made in chapter 5 were based on relative cost and so take into account the possible improvement of the implementation). Potentially the recycling idea could also be used in the context of preconditioned conjugate gradient methods, to avoid the construction of a preconditioner on each fixed point step. Another possible avenue of research is the combination of AMG-R with some other linear solver e.g PCG, which can be used on early fixed point steps (where most AMG setups are needed) with AMG-R taking over for later steps.

Moving on to the work on staircase reduction, there are still some outstanding issues regarding the effective implementation of nonlinear multigrid for some of the models considered in chapter 6, particularly model 2. We may also consider the use of multigrid methods for solving some of the higher order models, it is likely that different smoothers will be needed for these problems.

For the work on image deblurring, as mentioned in chapter 7 work on the two-step model and the new solvers for the TV problem is still ongoing. In addition we would ultimately like

to develop a nonlinear multigrid solver for the full TV deblurring problem. Using a version of method 1 (7.48) with gauss-seidel solver as smoother and using a smoother based on circulant extension have not so far produced good results, further exploration of these ideas is planned. We may also possibly consider, for the deblurring problem, the use of multilvel optimization techniques used by Chan and Chen for the denoising problem (see [25] for details and §A.7.2 for an introduction).

Finally I mention that we may consider multigrid solvers for other imaging problems, for example the active contour segmentation method of Chan and Vese [31] involves solution of the Euler-Lagrange equation

$$\delta_\epsilon(\phi)\left[\mu p\left(\int_\Omega \delta_\epsilon(\phi)|\nabla\phi|\right)^{p-1}\nabla.\left(\frac{\nabla\phi}{|\nabla\phi|}\right)-\nu-\lambda_1(u-c_1)^2-\lambda_2(u-c_2)^2\right]=0 \qquad (8.1)$$

where ϕ is the contour used to segment the image u and δ_ϵ is a regularization of the dirac measure. We can see that this equation has some similarities with the nonlinear equation associated with total variation denoising.

Appendix A

Optimization

Main Reference Material: [8, 10, 15, 25, 42, 49, 50, 51, 54, 64, 65, 73, 81, 82, 87, 97, 105]

In this appendix I give a brief review of some useful results and techniques in optimization. In section A.1 I introduce unconstrained and constrained minimization and the definitions of local and global minima. Section A.2 considers the conditions for the existence of minima, while section A.3 reviews the necessary and sufficient conditions for minima of unconstrained optimization problems, section A.4 gives some specific results relating to convex optimization problems. The KKT conditions for constrained optimization problems are touched upon in section A.5 for optimization problems on $\mathbb{R}^n$ only. Section A.6 looks at optimization methods for unconstrained problems for both quadratic and more general nonlinear problems on $\mathbb{R}^n$, focusing mainly on the steepest descent and conjugate gradient methods, finally in section A.7 several multigrid optimization methods are reviewed.

A.1 Optimization Problems

A.1.1 Unconstrained Optimization

In the majority of this chapter we will consider unconstrained optimization problems of the form

$$\min_{u \in H} f(u), \tag{A.1}$$

where H is a Hilbert space (or more generally an open subset of a normed space) and $f(u)$ is assumed to be a proper functional i.e there exits some $\tilde{u} \in H$ such that $f(\tilde{u}) < +\infty$. A solution to such a problem can either be a local minimum or a global minimum, the definition of which are given below:

Definition A.1.1 (Local Minimum)
$\bar{u} \in H$ is a local minimum of f if $f(\bar{u}) \leq f(\bar{u}+h)$ for all $h \in H$ such that $\|h\| < \delta$.

Definition A.1.2 (Global Minimum)
$\bar{u} \in H$ is a global minimum of f if $f(\bar{u}) \leq f(u)$ for all $u \in H$.

If we replace $\leq$ by $<$ in both definitions, require that h in the first definition is nonzero and replace $u \in H$ by $u \in H \backslash \{\bar{u}\}$ in the second definition we get the definitions of a strict local/global minimum. Local and global maxima are defined in a similar way.

In general finding the global minima if many local minima exist is difficult.

A.1.2 Constrained Optimization

In addition certain constraints may be placed on the solution u. Constraints usually take the form of equality or inequality constraints. If V and W are Hilbert spaces and V is ordered, with order relation $\preceq$, then a typical constrained optimization problem will be

$$\min_{u \in H} f(u) \tag{A.2}$$

subject to

$$p(u) \preceq 0 \tag{A.3}$$

$$q(u) = 0 \tag{A.4}$$

where $p : H \to V$ and $q : H \to W$. Specifically we will only consider here the case that $H = \mathbb{R}^n$, $V = \mathbb{R}^m$ and $W = \mathbb{R}^l$, so that for $\mathbf{x} \in \mathbb{R}^n$ the constraints take the form $p_i(\mathbf{x}) \leq 0$ for $i = 1, .., m$ and $q_i(\mathbf{x}) = 0$ for $i = 1, .., l$ where p_i and q_i map $\mathbb{R}^n$ into $\mathbb{R}$.

The definition of local and global minimisers have to be modified so that the h in definition A.1.1 is such that $\bar{u} + h \in C$ where C is the constraint set and $u \in C$ in definition A.1.2.

A.2 Existence of Minima

The solution to a minimization problem does not always exist, for example in the simple case $x \in \mathbb{R}$, no minimum exists for either $f(x) = x^3$ or $f(x) = e^{-x}$. In the following I give the conditions required for the existence of a minimum of a functional $f(u) : H \to \mathbb{R}$. First some definitions will be needed.

Definition A.2.1 (Bounded Sequence)
A sequence $\{v_n\}$ in a Hilbert space H is bounded if there exists a number M such that $\|v_n\|_H \leq M$ for all n.

Definition A.2.2 (Weak Lower Semicontinuity)
A functional $f : H \to \mathbb{R}$, where H is a Hilbert space is weakly lower semicontinuous if

$$f(u_*) \leq \lim_{n\to\infty} \inf f(u_n) \tag{A.5}$$

whenever u_n converges weakly to u_*.

Definition A.2.3 (Coercive Functional)
A functional $f : H \to \mathbb{R}$ is coercive if $f(u_n) \to \infty$ whenever $\|u_n\|_H \to \infty$.

The following Lemma will also be useful.

Lemma A.2.1 *Let $\{u_k\}$ be a bounded sequence in a Hilbert space H, then there exists a weakly convergent subsequence $\{u_{kl}\}$ i.e $(v, u_{kl})_H$ converges to $(v, \bar{u})_H$ for all $v \in H$.*

Finally we come to the main theorem

Theorem A.2.1 *If $f : H \to \bar{\mathbb{R}} = [-\infty, \infty]$ is weakly lower semicontinuous and coercive, then there exists $\bar{u} \in H$ which is a global minimizer of f over H.*

Proof
Let $\theta = \inf_{u\in H} f(u)$, then there exists a sequence $\{u_k\}$ such that $f(u_k)$ converges to θ. Given that f is coercive the sequence $\{u_k\}$ is bounded which by Lemma A.2.1 implies that there is a subsequence $\{u_{kl}\}$ which converges weakly to some $\bar{u} \in H$. From the weak lowersemicontinuity of f we have

$$f(\bar{u}) \leq \lim_{l\to\infty} \inf f(u_{kl}) = \theta \tag{A.6}$$

Hence $\bar{u}$ is a global minimum of f. ∎

More generally we can say that a f has a minimizer over any closed convex set S, since closed convex sets in a Hilbert space are weakly closed i.e if $\{u_n\}$ is a bounded sequence in S there will exist a weakly convergent subsequence who's weak limit is contained in S.

A.3 Optimality Conditions For Unconstrained Minimization

For functions on the real numbers we know that $f(x)$ has a stationary point when $\frac{df}{dx} = 0$ and that a stationary point $\bar{x}$ is a minimum if $\frac{d^2f}{dx^2}(\bar{x}) > 0$. In this section the optimality conditions for an unconstrained minimization problem of the form $\min_{u\in H} f(u)$ where H is a Hilbert space, will be given. These conditions will be based on two concepts of derivatives which will be introduced below.

A.3.1 The Frechet Derivative

The Frechet Derivative is a generalization of the ordinary derivative of a function on the real numbers to operators on normed spaces.

Definition A.3.1 (The Frechet Derivative)
An operator $f : N \to M$, where N and M are normed spaces is said to be Frechet differentiable at $\bar{u} \in N$ if there exists a $\delta > 0$ such that for all $h \in N$ satisfying $\|h\| < \delta$

$$f(\bar{u} + h) - f(\bar{u}) = Lh + \epsilon(h). \tag{A.7}$$

where $L : N \to M$ is a continuous linear operator and $\lim_{h \to 0} \frac{\|\epsilon(h)\|}{\|h\|} = 0$. L is known as the Frechet derivative of f at $\bar{u}$ and is often denoted $f'(\bar{u})$

Remark A.3.1 *Clearly if f is a linear operator then $f(\bar{u} + h) = f(\bar{u}) + f(h)$ and $f'(\bar{u}) = f$ i.e f is it's own Frechet derivative. It can also be shown that the usual rules of elementary calculus e.g the product rule generalize to the Frechet derivative.*

The second Frechet derivative is defined in a similar way, $f''(\bar{u})$ will be a linear operator mapping N into the space of bounded linear operators from N into M, such that for all $k \in N$ satisfying $\|k\| < \delta$, for some δ

$$f'(\bar{u} + k) - f'(\bar{u}) = f''(\bar{u})k + \epsilon(k). \tag{A.8}$$

with $\lim_{k \to 0} \frac{\|\epsilon(k)\|}{\|k\|} = 0$. Note that if ψ is a linear operator which maps N into $B(N, M)$ then if $k \in N$ ψk is a linear operator from N into M which means that if $h \in N$, $(\psi k)h \in M$ and thus ψ is a bilinear mapping from $N \times N$ into M. The bilinear map $f''(\bar{u})$ is symmetric and can therefore be uniquely determined if $f''(\bar{u})(h, h)$ is known for all h.

An equivalent definition of the second Frechet defivative of $f : N \to M$ at $\bar{u}$ is the bilinear mapping from $N \times N$ into M such that for all $h \in N$ with $\|h\| < \delta$ for some δ

$$f(\bar{u} + h) = f(\bar{u}) + f'(\bar{u})h + 1/2 f''(\bar{u})(h, h) + \epsilon(h). \tag{A.9}$$

where $\lim_{h \to 0} \frac{\|\epsilon(h)\|}{\|h\|^2} = 0$. This result follows from a generalized version of Taylors formula in Banach spaces.

If $f : H \to \mathbb{R}$ where H is a Hilbert space, then if f is Frechet differentiable at $\bar{u}$ by the Reisz representation theorem (see §2.2) there will be a unique vector $grad(\bar{u}) \in H$ such that

$$f'(\bar{u})h = (h, grad(\bar{u})). \tag{A.10}$$

It can also be shown that if f is twice Frechet differentiable at $\bar{u}$ then there will be a unique self-adjoint operator $Hess(\bar{u}) \in B(H, H)$ such that

$$f''(\bar{u})(h, k) = (h, Hess(\bar{u})k). \tag{A.11}$$

It can be shown that if $f : \mathbb{R}^n \rightarrow \mathbb{R}$ then

$$[grad(\bar{\mathbf{x}})]_i = \nabla f(\bar{\mathbf{x}})_i = \frac{\partial f}{\partial x_i}(\bar{\mathbf{x}}). \tag{A.12}$$

simply by substituting $\mathbf{h} = h_i \mathbf{e}_i$ into (A.7), where $\mathbf{e}_i$ is the vector with 1 in position i and zeros everywhere else, and letting $h_i \rightarrow 0$ and similarly if the Hessian exists then it has entries

$$Hess(\bar{\mathbf{x}})_{ij} = \frac{\partial^2 f}{\partial x_i \partial x_j}(\bar{\mathbf{x}}). \tag{A.13}$$

A.3.2 The Gateaux Derivative

Definition A.3.2 (The Gateaux Variation)
If $f : N \rightarrow M$ where N and M are normed spaces, $\delta f(\bar{u}, h)$ is called the Gateaux variation (or first variation) of $f(u)$ at $\bar{u} \in N$ if for $t \in \mathbb{R}$

$$\delta f(\bar{u}, h) = \frac{d}{dt} f(\bar{u} + th)|_{t=0}. \tag{A.14}$$

exists for all $h \in N$.

Note that

$$\begin{aligned} \frac{d}{dt} f(\bar{u} + th)|_{t=0} &= \lim_{\tau \rightarrow 0} \frac{f(\bar{u} + (t+\tau)h) - f(u + th)}{\tau}\Big|_{t=0} \\ &= \lim_{\tau \rightarrow 0} \frac{f(\bar{u} + \tau h) - f(u)}{\tau} \end{aligned} \tag{A.15}$$

This is known as the directional derivative of f in the direction of h. From this definition and the definition of the Frechet Derivative it is clear that if the Frechet Derivative $f'(\bar{u})$ exists then $\delta f(\bar{u}, h) = f'(\bar{u})h$, existence of the Gateaux variation does not imply existence of the Frechet derivative.

The Gateaux variation if it exists is unique, it is also homogenous of the first degree i.e for all $h \in H$ and $\lambda \in \mathbb{R}$, $\delta f(\bar{u}, \lambda h) = \lambda \delta f(\bar{u}, h)$, this fact can be used to show that the following theorem holds

Theorem A.3.1 *$\delta f(\bar{u}, h)$ is the Gateaux variation of $f(u)$ at $\bar{u}$ if and only if there exists a $\delta > 0$ such that for all $h \in H$ such that $\|h\| < \delta$*

$$f(\bar{u} + h) - f(\bar{u}) = \delta f(\bar{u}, h) + \epsilon(h), \tag{A.16}$$

where $\delta f(\bar{u}, h)$ is homogenous of the first degree and $\lim_{t \rightarrow 0} \epsilon(th)/t = 0$.

For more details see [82].

Definition A.3.3 (Gateaux Derivative)
If the Gateaux variation $\delta f(\bar{u}, h) = D_G f(\bar{u})h$ where $D_G f(\bar{u})$ is a continuous linear map, then $D_G f(\bar{u})$ is called the Gateaux derivative of f at $\bar{u}$.

In the following it is shown that if the Gateaux derivative exists and is continuous in the operator norm, then the Frechet Derivative also exists and is equivalent to the Gateaux derivative. The result is proved for the functional case only, for a more general proof for operators see [42]. The following Lemma which is a generalization of the mean value theorem will be required.

Lemma A.3.1 (Mean Value Theorem)
If $f : N \to \mathbb{R}$ where N is a normed space and f is Gateaux differentiable on N, then for each u and v in N, there exists some ξ on the line segment between u and v such that

$$f(v) - f(u) = D_G f(\xi)(v - u) \tag{A.17}$$

Proof
Define $g(t) = f(u + t(v - u))$, then

$$\begin{aligned} g'(t) &= \lim_{\tau \to 0} \frac{f(u + (t + \tau)(v - u)) - f(u + t(v - u))}{\tau} \\ &= \lim_{\tau \to 0} \frac{f(u + t(v - u) + \tau(v - u)) - f(u + t(v - u))}{\tau} \\ &= \delta f(u + t(v - u), v - u) = D_G f(u + t(v - u))(v - u). \end{aligned} \tag{A.18}$$

By the ordinary mean value theorem there exists $\theta \in (0, 1)$ such that $g'(\theta) = g(1) - g(0)$ i.e there exits ξ between u and v such that

$$D_G f(\xi)(v - u) = f(v) - f(u) \tag{A.19}$$

■

Theorem A.3.2 *Let $f : N \to \mathbb{R}$, then if f is Gateaux differentiable on N and the Gateaux derivative is continuous in the operator norm at $\bar{u}$, the Frechet derivative of f exists at $\bar{u}$ and coincides with the Gateaux derivative.*

Proof
From Lemma A.3.1 there exists ξ between $\bar{u}$ and $\bar{u} + h$ such that

$$D_G f(\xi)h = f(\bar{u} + h) - f(\bar{u}) \tag{A.20}$$

for any $h \in N$. We therefore have

$$f(\bar{u} + h) - f(\bar{u}) = D_G f(\bar{u})h + [D_G f(\xi)h - D_G f(\bar{u})h] \tag{A.21}$$

Defining $\epsilon(h) = D_G f(\xi)h - D_G f(\bar{u})h$ we have

$$\frac{|\epsilon(h)|}{\|h\|} = \frac{|D_G f(\xi)h - D_G f(\bar{u})h|}{\|h\|} \leq \|D_G f(\xi) - D_G f(\bar{u})\|. \tag{A.22}$$

Given that $D_G f$ is continuous in the operator norm at $\bar{u}$ the right hand side of (A.22) goes to zero as $\xi \to \bar{u}$ i.e $|\epsilon(h)|/\|h\| \to 0$ as $h \to 0$ and hence $D_G f(\bar{u})$ is the Frechet derivative of f at $\bar{u}$. ■

Definition A.3.4 (Second Variation)
$\delta^2 f(\bar{u}, h)$ is called the second Gateaux variation (or second variation) of $f(u)$ at $\bar{u}$ if for $t \in \mathbb{R}$

$$\delta^2 f(\bar{u}, h) = \frac{d^2}{dt^2} f(\bar{u} + th)|_{t=0} \tag{A.23}$$

exists for all $h \in H$.

We have

$$\delta^2 f(\bar{u}, h) = \frac{d^2}{dt^2} f(\bar{u} + th)_{t=0}$$

$$= \lim_{\tau \to 0} \left[\frac{\lim_{\rho \to 0} \frac{f(\bar{u}+(t+\rho+\tau)h) - f(u+(t+\tau)h)}{\rho} - \lim_{\rho \to 0} \frac{f(\bar{u}+(t+\rho)h) - f(u+th)}{\rho}}{\tau} \right] \Bigg|_{t=0}$$

$$= \lim_{\tau \to 0} \frac{\delta f(\bar{u} + \tau h, h) - \delta f(\bar{u}, h)}{\tau}. \tag{A.24}$$

If f is twice Frechet differentiable, then $\delta^2 f(\bar{u}, h) = f''(\bar{u})(h, h)$. More generally $f''(\bar{u})(h, v) = \frac{\partial^2}{\partial t \partial s} f(\bar{u} + th + sv)|_{t=0, s=0}$.

A.3.3 Necessary Conditions

Below the first and second order necessary conditions for local minima will be given.

Theorem A.3.3 (First Order Necessary Condition)
If $f : H \to \mathbb{R}$ is Gateaux differentiable at $\bar{u}$ and $\bar{u}$ is a local minimum of f then $D_G f(\bar{u}) = 0$.

Proof
Since $\bar{u}$ is a local minimum of f we have for any $v \in H$ and $t \in \mathbb{R}^+$ sufficiently small

$$\frac{f(\bar{u} + tv) - f(\bar{u})}{t} \geq 0 \tag{A.25}$$

Letting $t \to 0$ we have $D_G f(\bar{u})v \geq 0$ for all $v \in H$. Taking $v = w$ where w is arbitrary we have $D_G f(\bar{u})w \geq 0$, taking $v = -w$ and using the fact that $D_G f(\bar{u})$ is linear we have $D_G f(\bar{u})w \leq 0$, which implies that $D_G f(\bar{u}) = 0$. ∎

Remark A.3.2 *Clearly if f is Frechet differentiable at $\bar{u}$ then $f'(\bar{u}) = 0$ is a necessary condition for a local minimum, the slightly weaker result given above is more useful in practice. It can also be shown [82] using theorem A.3.1 that $\delta(\bar{u}, h) = 0$ is a necessary condition for a local minimum even if $\delta(\bar{u}, h) = 0$ is not linear.*

The following theorem provides a second order necessary condition.

Theorem A.3.4 (Second Order Necessary Condition)
If f is twice Frechet differentiable at $\bar{u}$ and $\bar{u}$ is a local minimum of f then $f''(\bar{u})$ is positive semidefinite.

Proof

Substituting $h = tv$ into (A.9) with $t \in \mathbb{R}$ and taking account of the first order condition from above we have

$$\frac{f(\bar{u}+tv)-f(\bar{u})}{t^2} = 1/2f''(\bar{u})(v,v) + \|v\|_2^2\rho(\bar{u},tv), \tag{A.26}$$

where $\rho(\bar{u},h) \to 0$ as $h \to 0$. If we assume that $f''(\bar{u})(v,v) < 0$ for some v then we have that the left hand side is negative for sufficiently small t and hence $\bar{u}$ is not a local minimum, which is a contradiction. ∎

A.3.4 Sufficient Condition

A sufficient condition for a strict local minimum is given below.

Theorem A.3.5 (Sufficient Condition)
If $f : H \to \mathbb{R}$ is twice Frechet differentiable at $\bar{u}$, $f'(\bar{u}) = 0$ and $f''(\bar{u})(v,v) \geq \theta$ for all $v \in H$ with $\theta > 0$ independent of v, then $\bar{u}$ is a strict local minimum of f.

Proof

From A.9 and the conditions stated above we have for any $v \in H$ and $t \in \mathbb{R}$ sufficiently small

$$\begin{aligned} f(\bar{u}+tv) - f(\bar{u}) &= 1/2t^2f''(\bar{u})(v,v) + t^2\|v\|^2\rho(\bar{u},tv) \\ &\geq 1/2\theta t^2 + t^2\|v\|_2^2\rho(\bar{u},tv) \end{aligned} \tag{A.27}$$

therefore

$$\frac{f(\bar{u}+tv)-f(\bar{u})}{t^2\|v\|^2} \geq \frac{\theta}{2\|v\|^2} + \rho(\bar{u},tv) \tag{A.28}$$

For t small enough the right hand side is positive and hence $\bar{u}$ is a strict local minimum of f. ∎

Note that the requirement that $f''(\bar{u})(v,v) \geq \theta$ for all $v \in H$ with $\theta > 0$ independent of v (strong positivity) is equivalent to the hessian $Hess(\bar{u})$ being a positive definite operator if H is finite dimensional (see §2.2.3).

A.4 Convex Optimization

A special type of optimization problem is the problem of minimizing a convex functional on a convex set, as we shall see below, the first order necessary condition for a minimum is also sufficient for these problems.

Definition A.4.1 (Convex Set)
A set S is convex if for all $u, v \in S$

$$\lambda u + (1-\lambda)v \in S \tag{A.29}$$

for all $\lambda \in [0,1]$.

Definition A.4.2 (Convex Functional)
A functional $f : S \to \mathbb{R}$, where S is a convex set, is convex if for all $u, v \in S$

$$f(\lambda u + (1-\lambda)v) \leq \lambda f(u) + (1-\lambda)f(v) \tag{A.30}$$

for all $\lambda \in [0,1]$.

A function is strictly convex if the strict inequality holds for $u \neq v$ and $\lambda \in (0,1)$. The following lemma will be useful in the proofs that follow.

Lemma A.4.1
Suppose that $f : S \to \mathbb{R}$ where S is a nonempty open convex set and that f is Frechet differentiable on S, then f is convex if and only if

$$f(v) \geq f(u) + f'(u)(v-u) \forall u, v \in S. \tag{A.31}$$

Proof
Assume that f is convex, then

$$f(\lambda v + (1-\lambda)u) \leq \lambda f(v) + (1-\lambda)f(u) \tag{A.32}$$

for all $u, v \in S$ and $\lambda \in [0,1]$, or equivalently

$$\frac{f(u + \lambda(v-u)) - f(u)}{\lambda} \leq f(v) - f(u). \tag{A.33}$$

From the fact that f is Frechet differentiable we have that the term in the numerator of the left hand side is $\lambda f'(u)(v-u) + \lambda \|v-u\| \rho(u, \lambda(v-u))$ where $\rho \to 0$ as $\lambda(v-u) \to 0$, therefore letting $\lambda \to 0$ in (A.33) gives the desired result.

Conversely assume that (A.31) is satisfied and let $u = \lambda w + (1-\lambda)z$ where $w, z \in S$ and $\lambda \in [0,1]$, then given that

$$f(w) \geq f(u) + f'(u)(w-u) \tag{A.34}$$

$$f(z) \geq f(u) + f'(u)(z-u) \tag{A.35}$$

we have that

$$\lambda f(w) + (1-\lambda)f(z) \geq f(u) + f'(u)(\lambda(w-u) + (1-\lambda)(z-w))$$

$$= f(u) + f'(u)\left[\lambda w - \lambda u + (1-\lambda)z - (1-\lambda)u\right]$$
$$= f(u) + f'(u)\left[\lambda w + (1-\lambda)z - u\right] = f(u)$$
$$= f(\lambda w + (1-\lambda)z). \tag{A.36}$$

■

Theorem A.4.1 *If S is a nonempty open convex set and $f : S \to \mathbb{R}$ is twice Frechet differentiable then f is convex if and only if $f''(u)$ is positive semidefinite for all $u \in S$.*

Proof

Suppose that f is convex. Let $u \in S$ and v be an arbitrary vector in $H \supset S$. For $t \in \mathbb{R}^+$ sufficiently small $u + tv \in S$. Given that f is twice Frechet differentiable we have

$$f(u+tv) = f(u) + tf'(u)v + 1/2t^2 f''(u)(v,v) + t^2\|v\|^2\rho(u,tv). \tag{A.37}$$

and $\rho(u,tv) \to 0$ as $tv \to 0$. From lemma A.4.1 we have that

$$f(u+tv) \geq f(u) + tf'(u)v. \tag{A.38}$$

Combining $(A.37)$ and $(A.38)$ we get

$$1/2f''(u)(v,v) + \|v\|^2\rho(u,tv) \geq 0. \tag{A.39}$$

Letting $t \to 0$ gives the desired result.

Conversely assume that $f''(u)$ is positive definite for all $u \in S$. Let $w, z \in S$ be arbitrary, Taylors theorem with the Lagrange form of the remainder gives

$$f(z) = f(w) + f'(w)(z-w) + 1/2f''(\zeta)(z-w, z-w). \tag{A.40}$$

The term ζ is a convex combination of w and z, which means that the truncation term is ≥ 0 and hence (A.31) is satisfied implying that f is convex. ■

Theorem A.4.2 *If S is a nonempty open convex set, $f : S \to \mathbb{R}$ is convex and $\bar{u}$ is a local minimum of f, then $\bar{u}$ is a global minimum of f.*

Proof

Assume that $\bar{u}$ is not a global minimum of f i.e there exists $v \in S$ such that

$$f(v) < f(\bar{u}). \tag{A.41}$$

Now let $z = \lambda\bar{u} + (1-\lambda)v$ where $\lambda \in (0,1)$ we have

$$\begin{aligned} f(z) = f(\lambda\bar{u} + (1-\lambda)v) &\leq \lambda f(\bar{u}) + (1-\lambda)f(v) \\ &< \lambda f(\bar{u}) + (1-\lambda)f(\bar{u}) = f(\bar{u}) \end{aligned}, \tag{A.42}$$

which implies that $f(\bar{u})$ is not a local minimum, which is a contradiction. ■

Theorem A.4.3 *If S is a nonempty open convex set, $f : S \to \mathbb{R}$ is strictly convex and $\bar{u}$ is a global minimum of f then $\bar{u}$ is unique.*

Proof

Assume that there exists a $v \in S$ such that $f(\bar{u}) = f(v)$. Let $z = \lambda\bar{u} + (1-\lambda)v$ where $\lambda \in (0,1)$, then

$$f(z) = f(\lambda\bar{u} + (1-\lambda)v) < \lambda f(\bar{u}) + (1-\lambda)f(\bar{u}) = f(\bar{u}). \tag{A.43}$$

Implying that $f(\bar{u})$ is not a minimum, which is a contradiction. ∎

Theorem A.4.4 (Necessary and Sufficient Condition)

If S is a nonempty open convex set and $f : S \to \mathbb{R}$ is convex and Frechet differentiable on S, then $f'(\bar{u}) = 0$ if and only if $\bar{u}$ is a global minimum of f.

Proof

The necessity was proved earlier for any f, to prove the sufficiency assume that $f'(\bar{u}) = 0$ then by lemma A.4.1

$$f(v) \geq f(\bar{u}) \; \forall v \in S \tag{A.44}$$

giving the desired result. ∎

A.4.1 Duality Theory

Duality theory is often used in convex optimization problems to rewrite an optimization problem in terms of its dual representation. Below a very brief introduction to this subject is given, for more details see [8, 15, 97]

Definition A.4.3 (Convex Conjugate)

If $f : H \to \bar{\mathbb{R}}$ where H is a Hilbert space, the convex conjugate $f^* : H^* \to \bar{\mathbb{R}}$ is defined by

$$f^*(v) = \sup_{u \in H} [(v,u) - f(u)] \tag{A.45}$$

where H^* is the dual space to H.

The convex conjugate, as it's name suggests is always convex, even if f isn't. Similarly f^{**} is defined as $(f^*)^*$ and acts on the space H^{**}. If H is reflexive, then $H^{**} = H$ and furthermore $f^{**} = f$ if and only if f is convex. Therefore if f is a convex functional defined on a convex set S, the following dual representation for f can be given

$$f(u) = \sup_{v \in S^*} (u,v) - f^*(v), \tag{A.46}$$

where S^* is defined by

$$S^* = \left\{ v \in H^* \middle| \sup_{u \in S} [(v,u) - f(u)] < \infty \right\} \tag{A.47}$$

and $S^{**} = S$.

A.5 Optimality Conditions for Constrained Problems

Below I state, without proof the first order Karush-Kuhn-Tucker (KKT) conditions for constrained minimization problems on $\mathbb{R}^n$ of the form

$$\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x}) \tag{A.48}$$

subject to

$$p(\mathbf{x}) \leq 0 \tag{A.49}$$

$$q(\mathbf{x}) = 0 \tag{A.50}$$

where $p : \mathbb{R}^n \to \mathbb{R}^m$ and $q : \mathbb{R}^n \to \mathbb{R}^l$, and state when they are necessary and sufficient, for more details see, for example [51, 49] and for a more general treatment of constrained problems in Hilbert space see [15].

First denote by by $\nabla p(\mathbf{x})$ the $m \times n$ matrix whose ith row is $\nabla p_i(\mathbf{x})^T$ and by $\nabla q(\mathbf{x})$ the $l \times n$ matrix whose ith row is $\nabla q_i(\mathbf{x})^T$. The feasible or constraint set is the set $\{\mathbf{x} \in \mathbb{R}^n | p(\mathbf{x}) \leq 0, q(\mathbf{x}) = 0\}$, if $\mathbf{x}$ is a feasible point, then the index $1 \leq i \leq m$ is said to be active if $p_i(\mathbf{x}) = 0$, otherwise it is said to be inactive. The active set of the feasible point $\mathbf{x}$ is denoted $A(\mathbf{x})$ and the inactive set $I(\mathbf{x})$.

Theorem A.5.1 (KKT First Order Necessary Conditions)
Let $\bar{\mathbf{x}}$ be a feasible point and let the vectors $\nabla q_i(\bar{\mathbf{x}})$ for $i = 1, .., l$ and $\nabla g_i(\bar{\mathbf{x}})$ for $i \in A(\bar{\mathbf{x}})$ be linearly independent, then if $\bar{\mathbf{x}}$ is a local minimum, there exists vectors λ and σ, such that

$$\begin{aligned} \nabla f(\bar{\mathbf{x}}) + \nabla p(\bar{\mathbf{x}})^T\lambda + \nabla q(\bar{\mathbf{x}})^T\sigma = 0 \\ \lambda \geq 0 \\ \lambda_i p_i(\bar{\mathbf{x}}) = 0, \quad i = 1, .., m \end{aligned} \tag{A.51}$$

The vectors λ and σ are often known as lagrange multipliers the components λ_i of λ can only be strictly positive if the index i is in the active set. Note that the first equation in (A.51) can be written equivalently as

$$\nabla f(\bar{\mathbf{x}}) + \sum_{i=1}^{m} \lambda_i \nabla p_i(\bar{\mathbf{x}}) + \sum_{i=1}^{l} \sigma_i \nabla q_i(\bar{\mathbf{x}}). \tag{A.52}$$

Before giving the conditions for the KKT conditions to be sufficient, first I need to define quasiconvexity and pseudoconveity.

Definition A.5.1 (Quasiconvex Function)
A function $f(\mathbf{x})$ which maps S into $\mathbb{R}$, where S is a convex set, is quasiconvex if for all $\mathbf{x}, \mathbf{y} \in S$ and all $\lambda \in [0, 1]$

$$f(\lambda\mathbf{x} + (1 - \lambda)\mathbf{y}) \leq \max(f(\mathbf{x}), f(\mathbf{y})). \tag{A.53}$$

Definition A.5.2 (Pseudoconvex Function)
A differentiable function $f(\mathbf{x})$ which maps $S \subset \mathbb{R}^n$ into $\mathbb{R}$, where S is a convex set, is pseudoconvex if for all $\mathbf{x}, \mathbf{y} \in S$

$$\nabla f(\mathbf{x})^T(\mathbf{y} - \mathbf{x}) \geq 0 \Rightarrow f(\mathbf{y}) \geq f(\mathbf{x}). \tag{A.54}$$

A convex function is quasiconvex and if it is differentiable pseudoconvex, a pseudoconvex function is also quasiconvex.

Theorem A.5.2 (KKT Sufficient Conditions)
Let $\bar{\mathbf{x}}$ be a feasible point and let $\bar{\mathbf{x}}$ and vectors λ and σ satisfy (A.51), then if $f(\mathbf{x})$ is pseudoconvex, $p_i(\mathbf{x})$, $i = 1, .., m$ are quasiconvex and $q_i(\mathbf{x})$, $i = 1, .., l$ are linear, $\bar{\mathbf{x}}$ is a global minimum.

A.6 Optimization Methods: Steepest Descent and Conjugate Gradient

In this section the steepest descent and conjugate gradient optimization methods for solving quadratic and then more general optimization problems on $\mathbb{R}^n$ are introduced. These problems may arise as the equivalent formulation of a system of linear equations or from the discretization of functionals such as those seen in Chapter 3. First some notation is needed: a vector $\mathbf{d} \in \mathbb{R}^n$ is a descent direction at $\mathbf{x}$ if $f(\mathbf{x} + \epsilon\mathbf{d}) < f(\mathbf{x})$ for $\epsilon > 0$ sufficiently small. Assuming $\nabla f(\mathbf{x})$ exists the direction $\mathbf{d}$ is a descent direction if the directional derivative $\lim_{\lambda \to 0} \frac{f(\mathbf{x}+\lambda\mathbf{d}) - f(\mathbf{x})}{\lambda} = \nabla f(\mathbf{x})^T\mathbf{d}$ is negative and the direction $-\nabla f(\mathbf{x})$ which minimizes the directional derivative is known as the direction of steepest descent.

We start by considering the quadratic minimization problem

$$\min_{\mathbf{x}} f(\mathbf{x}), \; f(\mathbf{x}) = 1/2\mathbf{x}^T A\mathbf{x} - \mathbf{x}^T\mathbf{b}. \tag{A.55}$$

where $\mathbf{x}, \mathbf{b} \in \mathbb{R}^n$ and A is an $n \times n$ matrix. Given that $\mathbf{x}^T A\mathbf{x} = (\mathbf{x}^T A\mathbf{x})^T = \mathbf{x}^T A^T\mathbf{x}$, the matrix A can be assumed without loss of generality to be symmetric. The gradient of f at any $\mathbf{x}$ is $A\mathbf{x} - \mathbf{b}$ and the Hessian is A. In addition we assume that A is a positive definite matrix. The quadratic has unique solution $\mathbf{x}^*$ satisfying $\nabla f(\mathbf{x}^*) = A\mathbf{x}^* - \mathbf{b} = 0$. The approach used here follows that used by Shewchuck in [87], see also [37, 53, 81] for more on conjugate gradient methods.

A.6.1 Steepest Descent

In the method of steepest descent (A.55) is minimized by selecting some initial guess $\mathbf{x}_0$ and then on step $k + 1$ of the method updating the current approximation $\mathbf{x}_k$ by taking a step

in the direction of steepest descent $-\nabla f(\mathbf{x}_k) = \mathbf{b} - A\mathbf{x}_k = \mathbf{r}_k$ where $\mathbf{r}_k$ denotes the current residual.

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{r}_k. \tag{A.56}$$

The value of α_k is chosen to minimize f along the search direction $\mathbf{r}_k$ i.e α_k solves

$$\frac{\partial}{\partial \alpha_k} f(\mathbf{x}_k + \alpha_k \mathbf{r}_k) = \nabla f(\mathbf{x}_k + \alpha_k \mathbf{r}_k)^T \mathbf{r}_k = 0. \tag{A.57}$$

Replacing $\nabla f(\mathbf{x}_k + \alpha_k \mathbf{r}_k)$ by $A(\mathbf{x}_k + \alpha_k \mathbf{r}_k) - \mathbf{b}$ and rearranging we get

$$\alpha_k = \frac{\mathbf{r}_k^T \mathbf{r}_k}{\mathbf{r}_k^T A \mathbf{r}_k}. \tag{A.58}$$

The method of steepest descent for the quadratic problem is given in Algorithm 23.

Algorithm 23 Steepest Descent for Quadratic Problems

Choose $\mathbf{x}_0$, set $\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0$, $k = 0$.

While $\|\mathbf{r}_k\|_2 > tol$

$\mathbf{q}_k = A\mathbf{r}_k$

$\alpha_k = \frac{\mathbf{r}_k^T \mathbf{r}_k}{\mathbf{r}_k^T \mathbf{q}_k}$

$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{r}_k$

$\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k \mathbf{q}_k$

$k \leftarrow k + 1$

end

Note that the fact that $\mathbf{r}_{k+1} = \mathbf{b} - A(\mathbf{x}_k + \alpha_k \mathbf{r}_k) = \mathbf{r}_k - \alpha_k A\mathbf{r}_k$ is used, to avoid an extra matrix vector product, to prevent the accumulation of roundoff error $\mathbf{r}_{k+1}$ should periodically be evaluated as $\mathbf{b} - A\mathbf{x}_{k+1}$. The method is stopped when the residual is small enough.

Convergence

The following upper bound on the energy norm of the error after k steps of the steepest descent method for quadratic problems is given in, for example, [87]

$$\|\mathbf{e}_k\|_A \leq \left(\frac{\kappa - 1}{\kappa + 1}\right)^k \|\mathbf{e}_0\|_A, \tag{A.59}$$

where κ is the condition number of the matrix A which is equal to the largest eigenvalue of A divided by the smallest eigenvalue of A. Clearly the larger the condition number is (the more ill-conditioned A is) the slower the convergence of the steepest descent method.

To see the connection between the decrease in the energy norm of the error and the decrease in the Euclidean norm of the residual the following lemma can be used

Lemma A.6.1 *If A is spd with eigenvalues $\lambda_1 \geq \lambda_2 \geq, .., \geq \lambda_n$, then for all $z \in \mathbb{R}^n$*

$$\lambda_n^{1/2} \|\mathbf{z}\|_A \leq \|A\mathbf{z}\|_2 \leq \lambda_1^{1/2} \|\mathbf{z}\|_A$$

Proof

Let $\mathbf{v}_i$ be the unit eigenvector associated with λ_i, then we can write $A = V\Lambda V^T$ where V is a matrix whose columns are the $\mathbf{v}_i$ and Λ is a diagonal matrix of the eigenvalues. We have

$$A\mathbf{z} = \sum_{i=1}^{n} \lambda_i (\mathbf{v}_i^T \mathbf{z}) \mathbf{v}_i. \tag{A.60}$$

Hence

$$\begin{aligned} \lambda_n \|\mathbf{z}\|_A^2 = \lambda_n \mathbf{z}^T A\mathbf{z} = \lambda_n \textstyle\sum_{i=1}^n \mathbf{z}^T \lambda_i (\mathbf{v}_i^T \mathbf{z}) \mathbf{v}_i = \lambda_n \sum_{i=1}^n \lambda_i (\mathbf{v}_i^T \mathbf{z})^2 \\ \leq \textstyle\sum_{i=1}^n \lambda_i^2 (\mathbf{v}_i^T \mathbf{z})^2 = \|A\mathbf{z}\|_2^2 \\ \leq \lambda_1 \textstyle\sum_{i=1}^n \lambda_i (\mathbf{v}_i^T \mathbf{z})^2 = \lambda_1 \mathbf{z}^T A\mathbf{z} = \lambda_1 \|\mathbf{z}\|_A^2 \end{aligned} . \tag{A.61}$$

Taking square roots gives the desired result. ∎

With this result we have that $\|\mathbf{r}_k\|_2 = \|A\mathbf{e}_k\|_2 \leq \lambda_1^{1/2} \|\mathbf{e}_k\|_A$ and $\|\mathbf{r}_0\|_2 = \|A\mathbf{e}_0\|_2 \geq \lambda_n^{1/2} \|\mathbf{e}_0\|_A$, which gives us

$$\frac{\|\mathbf{r}_k\|_2}{\|\mathbf{r}_0\|_2} \leq \sqrt{\kappa} \frac{\|\mathbf{e}_k\|_A}{\|\mathbf{e}_0\|_A} \leq \sqrt{\kappa} \left(\frac{\kappa - 1}{\kappa + 1} \right)^k. \tag{A.62}$$

A.6.2 The Method of Conjugate Directions

The problem with the steepest descent method is that it often ends up taking steps in the same direction as earlier steps. A method which takes exactly one step in each search direction is the method of conjugate directions.

In the method of conjugate directions n search directions $\mathbf{d}_0, ..., \mathbf{d}_{n-1}$ are selected such that they are A-orthogonal i.e for $k \neq j$ $\mathbf{d}_k^T A \mathbf{d}_j = 0$, then on step k of the method a step is taken in the direction of $\mathbf{d}_k$. The length of the step α_k is again chosen to minimize $f(\mathbf{x}_{k+1})$

$$\begin{aligned} (A(\mathbf{x}_k + \alpha_k \mathbf{d}_k) - \mathbf{b})^T \mathbf{d}_k = 0 \\ -\mathbf{r}_k^T \mathbf{d}_k + \alpha_k \mathbf{d}_k^T A \mathbf{d}_k = 0 \\ \alpha_k = \frac{\mathbf{d}_k^T \mathbf{r}_k}{\mathbf{d}_k^T A \mathbf{d}_k} \end{aligned} . \tag{A.63}$$

Theorem A.6.1 *If the vectors $\mathbf{d}_0, .., \mathbf{d}_{n-1}$ are A-orthogonal, then they are linearly independent.*

Proof

Assume that they are not linearly independent, then there exists $\lambda_0, .., \lambda_{n-1}$, such that

$$\sum_{i=0}^{n-1} \lambda_i \mathbf{d}_i = 0 \tag{A.64}$$

and at least one of the λ_i is non zero. For any i multiplying (A.64) by $\mathbf{d}_i^T A$ we get from the A-orthogonality of the $\mathbf{d}$ that $\lambda_i = 0$, which is a contradiction. ∎

Theorem A.6.2 *The method of conjugate directions will find the exact minimum* $\mathbf{x}^*$ *in at most* n *steps.*

Proof

Let us express the initial error as a linear combination of the search directions.

$$\mathbf{e}_0 = \sum_{j=0}^{n-1} \delta_j \mathbf{d}j. \tag{A.65}$$

Multiplying by $\mathbf{d}_k^T A$ we get

$$\mathbf{d}_k^T A \mathbf{e}_0 = \sum_{j=0}^{n-1} \delta_j \mathbf{d}_k^T A \mathbf{d}_j = \delta_k \mathbf{d}_k^T A \mathbf{d}_k. \tag{A.66}$$

This gives us

$$\delta_k = \frac{\mathbf{d}_k^T A \mathbf{e}_0}{\mathbf{d}_k^T A \mathbf{d}_k}. \tag{A.67}$$

using the A-orthogonality we can rewrite this as

$$\delta_k = \frac{\mathbf{d}_k^T A(\mathbf{e}_0 - \sum_{j=0}^{k-1} \alpha_j \mathbf{d}_j)}{\mathbf{d}_k^T A \mathbf{d}_k} = \frac{\mathbf{d}_k^T A \mathbf{e}_k}{\mathbf{d}_k^T A \mathbf{d}_k} = \frac{\mathbf{d}_k^T \mathbf{r}_k}{\mathbf{d}_k^T A \mathbf{d}_k} = \alpha_k. \tag{A.68}$$

Given that $\mathbf{e}_{k+1} = \mathbf{e}_k - \alpha_k \mathbf{d}_k$ we have that the step along $\mathbf{d}_k$ eliminates the $\mathbf{d}_k$ component of the error and hence the error is zero after at most n steps. ∎

Theorem A.6.3 *The error* $\mathbf{e}_k$ *on step* k *of the method of conjugate directions minimizes the energy norm in the space* $\mathbf{e}_0 + D_k$ *where* $D_k = span\{\mathbf{d}_0, .., \mathbf{d}_{k-1}\}$.

Proof

It is clear that $\mathbf{e}_k$ is chosen from $\mathbf{e}_0 + D_k$. Any vector $\mathbf{v} \in \mathbf{e}_0 + D_k$ can be written as $\mathbf{v} = \sum_{j=0}^{k-1} \theta_j \mathbf{d}_j + \sum_{j=k}^{n-1} \delta_j \mathbf{d}_j$ where the δ_j are as above. We have

$$\|\mathbf{v}\|_A^2 = \sum_{l=0}^{k-1}\sum_{j=0}^{k-1} \theta_l \theta_j \mathbf{d}_l^T A \mathbf{d}_j + \sum_{l=0}^{k-1}\sum_{j=k}^{n-1} \theta_l \delta_j \mathbf{d}_l^T A \mathbf{d}_j + \sum_{l=k}^{n-1}\sum_{j=0}^{k-1} \delta_l \theta_j \mathbf{d}_l^T A \mathbf{d}_j + \sum_{l=k}^{n-1}\sum_{j=k}^{n-1} \delta_l \delta_j \mathbf{d}_l^T A \mathbf{d}_j. \tag{A.69}$$

By the A-orthogonality of the search directions this is

$$\|\mathbf{v}\|_A^2 = \sum_{j=0}^{k-1} \theta_j^2 \mathbf{d}_j^T A \mathbf{d}_j + \sum_{j=k}^{n-1} \delta_j^2 \mathbf{d}_j^T A \mathbf{d}_j. \tag{A.70}$$

Equation (A.70) is minimized by taking $\theta_j = 0$ for $0 \leq h \leq k-1$ i.e taking $\mathbf{v} = \mathbf{e}_k$. ∎

Gram-Schmidt Conjugation

So far we have assumed that we have a set of A-orthogonal search directions. To find such a set we must use the conjugate Gram-Scmidt process outlined below

1. Take n linearly independent vectors $\mathbf{v}_0, .., \mathbf{v}_{n-1}$.
2. Set $\mathbf{d}_0 = \mathbf{v}_0$
3. For $k = 1, .., n-1$ set
$$\mathbf{d}_k = \mathbf{v}_k + \sum_{j=0}^{k-1} \beta_{kj} \mathbf{d}_j. \tag{A.71}$$
where the β_{kj}, $j < k$ are constructed so that all components of $\mathbf{v}_k$ not A-orthogonal to the previous search directions are removed.

The term β_{kp} can be found by multiplying $\mathbf{d}_p$ by $\mathbf{d}_k^T A$.

$$\mathbf{d}_k^T A \mathbf{d}_p = \mathbf{v}_k^T A \mathbf{d}_p + \sum_{j=0}^{k-1} \beta_{kj} \mathbf{d}_j^T A \mathbf{d}_p = \mathbf{v}_k^T A \mathbf{d}_p + \beta_{kp} \mathbf{d}_p^T A \mathbf{d}_p, \tag{A.72}$$

which implies that

$$\beta_{kp} = -\frac{\mathbf{v}_k^T A \mathbf{d}_p}{\mathbf{d}_p^T A \mathbf{d}_p}. \tag{A.73}$$

We see that to construct the search direction $\mathbf{d}_k$ we need to find $k-1$ β_{kp} values each of which requires a matrix vector multiplication, making the overall cost of performing the method of conjugate directions $O(n^3)$ in addition all previous search directions must be stored. The beauty of the conjugate gradient method introduced next is that most of the cost can be eliminated and there is no need to store old search directions. Before introducing the conjugate gradient method I introduce a few useful results. Firstly from the expansion of $\mathbf{e}_j = \sum_{l=j}^{n-1} \delta_l \mathbf{d}_l$ we have

$$\mathbf{d}_k^T A \mathbf{e}_j = \mathbf{d}_K^T \mathbf{r}_j = \sum_{l=j}^{n-1} \delta_l \mathbf{d}_k^T A \mathbf{d}_l = 0 \text{ if } k < j. \tag{A.74}$$

Also from (A.71)

$$\mathbf{d}_k^T \mathbf{r}_j = \mathbf{v}_k^T \mathbf{r}_j + \sum_{l=0}^{k-1} \beta_{kl} \mathbf{d}_l^T \mathbf{r}_j, \tag{A.75}$$

which if $k < j$ gives us

$$\mathbf{v}_k^T \mathbf{r}_j = 0. \tag{A.76}$$

and if $j = k$ gives us

$$\mathbf{d}_k^T \mathbf{r}_k = \mathbf{v}_k^T \mathbf{r}_k. \tag{A.77}$$

A.6.3 The Conjugate Gradient Method

The conjugate gradient method is just the method of conjugate directions with the $\mathbf{v}_k$ set to be the residuals $\mathbf{r}_k$, from (A.74) we see that the residual is orthogonal to all previous search directions, which guarantees a new linearly independent search direction. With this choice of $\mathbf{v}_k$ (A.76) becomes

$$\mathbf{r}_k^T \mathbf{r}_j \text{ if } k \neq j. \tag{A.78}$$

which is important in the following discussion.

Recall that the error $\mathbf{e}_k$ on step k of the method of conjugate directions is chosen from $\mathbf{e}_0 + D_k$ and minimizes the energy norm among all vectors in this space. Given that the search directions are constructed from the $\mathbf{v}_k$ then $D_k = span\{\mathbf{d}_0, .., \mathbf{d}_{k-1}\} = span\{\mathbf{v}_0, .., \mathbf{v}_{k-1}\}$, which in the case of the conjugate gradient method is $span\{\mathbf{r}_0, .., \mathbf{r}_{k-1}\}$. Recalling that

$$\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k A\mathbf{d}_k \tag{A.79}$$

we see that in the special case of the conjugate gradient method

$$D_k = span\{\mathbf{r}_0, A\mathbf{r}_0, A^2\mathbf{r}_0, .., A^{k-1}\mathbf{r}_0\}. \tag{A.80}$$

This subspace is an example of a Krylov Subspace and the conjugate gradient method is part of a class of methods for solving $A\mathbf{x}^* = \mathbf{b}$ known as Krylov subspace methods, which also includes the generalized minimum residual (GMRES) method, which can be used for general A, see for example [81].

What makes the conjugate gradient method desirable is the following: from (A.74) $\mathbf{r}_k$ is orthogonal to D_k, however we see from above that D_k contains AD_{k-1} making $\mathbf{r}_k$ A-orthogonal to D_{k-1}. The β_{kp} terms in the Gram-Schmidt process are

$$\beta_{kp} = \frac{-\mathbf{r}_k^T A\mathbf{d}_p}{\mathbf{d}_p^T A\mathbf{d}_p} \tag{A.81}$$

for $p < k$, which using the A-orthogonality of $\mathbf{r}_k$ and D_{k-1} are zero, except when $p = k-1$. The need to store $\mathbf{d}_0, .., \mathbf{d}_{k-1}$ and carry out $k-1$ matrix vector products when evaluating $\mathbf{d}_k$ has reduced to doing one matrix vector product involving the most recent search direction. The calculation of $\beta_{k,k-1}$ which we can now simply denote β_k can be simplified further by using (A.79) to replace $A\mathbf{d}_{k-1}$ by $(\mathbf{r}_{k-1} - \mathbf{r}_k)/\alpha_k$ we then have

$$\mathbf{r}_k^T A\mathbf{d}_{k-1} = \frac{\mathbf{r}_k^T \mathbf{r}_{k-1} - \mathbf{r}_k^T \mathbf{r}_k}{\alpha_{k-1}}. \tag{A.82}$$

From (A.78) and the fact that $\alpha_{k-1} = \frac{\mathbf{d}_{k-1}^T \mathbf{r}_{k-1}}{\mathbf{d}_{k-1}^T A\mathbf{d}_{k-1}}$ we have

$$\beta_k = \frac{\mathbf{r}_k^T \mathbf{r}_k}{\mathbf{d}_{k-1}^T \mathbf{r}_{k-1}}. \tag{A.83}$$

From (A.77) this is equivalent to

$$\beta_k = \frac{\mathbf{r}_k^T \mathbf{r}_k}{\mathbf{r}_{k-1}^T \mathbf{r}_{k-1}}. \tag{A.84}$$

Putting everything together we have the conjugate gradient method (Algorithm 24).

Algorithm 24 The Conjugate Gradient Method

Choose $\mathbf{x}_0$, set $\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0$, $\mathbf{d}_0 = \mathbf{r}_0$ $\gamma_0 = \mathbf{r}_0^T \mathbf{r}_0$

For k=0,..,n-1

$\mathbf{q}_k = A\mathbf{d}_k$

$\alpha_k = \frac{\gamma_k}{\mathbf{d}_k^T \mathbf{q}_k}$

$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k$

$\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k \mathbf{q}_k$

$\gamma_{k+1} = \mathbf{r}_{k+1}^T \mathbf{r}_{k+1}$

$\beta_{k+1} = \frac{\gamma_{k+1}}{\gamma_k}$

$\mathbf{d}_{k+1} = \mathbf{r}_{k+1} + \beta_{k+1} \mathbf{d}_k$

end

Although the conjugate gradient method finds the exact minimum in n steps, for large problems it is not feasible to carry out n steps and the conjugate gradient method is usually terminated once the norm of the residual has been reduced by some specified amount.

Convergence

In for example [87] the following upper bound on the energy norm of the error after k steps of the conjugate gradient method is given:

$$\|\mathbf{e}_k\|_A \leq 2 \left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} \right)^k \|\mathbf{e}_0\|_A. \tag{A.85}$$

Comparing this to (A.59) we see that the conjugate gradient method converges significantly faster than the steepest descent method when κ is large.

Preconditioning

We have seen that the larger the condition number of the matrix A the worse the convergence of the conjugate gradient method is. One way to improve the performance of the conjugate gradient method might be to replace the original system $A\mathbf{x} = b$ by the new system

$$M^{-1} A\mathbf{x} = M^{-1}\mathbf{b}, \tag{A.86}$$

where M is a symmetric positive definite matrix and $M^{-1}A$ has a smaller condition number than A. The problem is that $M^{-1}A$ is not necessarily SPD, so instead the system

$$E^{-1} A E^{-T} \hat{\mathbf{x}} = E^{-1}\mathbf{b} \tag{A.87}$$

is solved, where $\hat{\mathbf{x}} = E^T\mathbf{x}$ and the matrix E satisfies $EE^T = M$ (any SPD matrix can be written in this form). $E^{-1}AE^{-T}$ is clearly SPD, furthermore it has the same eigenvalues and hence the same condition number as $M^{-1}A$ to see this let $M^{-1}A\mathbf{v} = \lambda\mathbf{v}$ then

$$E^{-T}E^{-1}A\mathbf{v} = E^{-T}E^{-1}A(E^{-T}E^T)\mathbf{v} = \lambda(E^{-T}E^T)\mathbf{v} \tag{A.88}$$

$$E^{-1}AE^{-T}(E^T\mathbf{v}) = \lambda(E^T\mathbf{v}) \tag{A.89}$$

Applying the conjugate gradient method to (A.87) we get Algorithm 25.

Algorithm 25 The Conjugate Gradient Method on $E^{-1}AE^{-T}\hat{\mathbf{x}} = E^{-1}\mathbf{b}$

Choose $\hat{\mathbf{x}}_0$, set $\hat{\mathbf{r}}_0 = E^{-1}\mathbf{b} - E^{-1}Ae^{-T}\hat{\mathbf{x}}_0$, $\hat{\mathbf{d}}_0 = \hat{\mathbf{r}}_0$ $\gamma_0 = \hat{\mathbf{r}}_0^T\hat{\mathbf{r}}_0$

For k=0,..,n-1

$\hat{\mathbf{q}}_k = E^{-1}AE^{-T}\hat{\mathbf{d}}_k$

$\alpha_k = \frac{\gamma_k}{\hat{\mathbf{d}}_k^T\hat{\mathbf{q}}_k}$

$\hat{\mathbf{x}}_{k+1} = \hat{\mathbf{x}}_k + \alpha_k\hat{\mathbf{d}}_k$

$\hat{\mathbf{r}}_{k+1} = \hat{\mathbf{r}}_k - \alpha_k\hat{\mathbf{q}}_k$

$\gamma_{k+1} = \hat{\mathbf{r}}_{k+1}^T\hat{\mathbf{r}}_{k+1}$

$\beta_{k+1} = \frac{\gamma_{k+1}}{\gamma_k}$

$\hat{\mathbf{d}}_{k+1} = \hat{\mathbf{r}}_{k+1} + \beta_{k+1}\hat{\mathbf{d}}_k$

end

In reality there is no need to find E instead we can set $\hat{\mathbf{r}}_k = E^{-1}\mathbf{r}_k$, $\hat{\mathbf{d}}_k = E^T\mathbf{d}_k$ and $\hat{\mathbf{q}}_k = E^{-1}\mathbf{q}_k$ and use the fact that $\hat{\mathbf{x}} = E^T\mathbf{x}$ to rewrite the algorithm in the form given in Algorithm 26.

Algorithm 26 The Preconditioned Conjugate Gradient Method

Choose $\mathbf{x}_0$, set $\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0$, $\mathbf{z}_0 = M^{-1}\mathbf{r}_0$, $\mathbf{d}_0 = \mathbf{z}_0$, $\gamma_0 = \mathbf{r}_0^T\mathbf{z}_0$

For k=0,..,N-1

$\mathbf{q}_k = A\mathbf{d}_k$

$\alpha_k = \frac{\gamma_k}{\mathbf{d}_k^T\mathbf{q}_k}$

$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k\mathbf{d}_k$

$\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k\mathbf{q}_k$

$\mathbf{z}_{k+1} = M^{-1}\mathbf{r}_{k+1}$

$\gamma_{k+1} = \mathbf{r}_{k+1}^T\mathbf{z}_{k+1}$

$\beta_{k+1} = \frac{\gamma_{k+1}}{\gamma_k}$

$\mathbf{d}_{k+1} = \mathbf{r}_{k+1} + \beta_{k+1}\mathbf{d}_k$

end

Choice of Preconditioner

There are several requirements that a preconditioner should meet if an efficient algorithm is to be achieved, the first is that it should be a good approximation to the original matrix A, on this criteria alone $M = A$ would be the best choice, only one step of the preconditioned conjugate gradient method would be required, but it would involve the solution of the system $Az = \mathbf{r}$, which is of course as difficult a problem as the original problem. The second requirement therefore is that the preconditioner should be able to be inverted relatively cheaply. Finally the preconditioner should not be too expensive to construct. The topic of preconditioning is large and growing, some preconditioners are quite general, others are more problem specific. Below I review only a few basic approaches to preconditioning that are relevant to my own topic.

Diagonal Preconditioning

The simplest preconditioner is a diagonal preconditioner i.e M is the diagonal matrix with the same diagonal as A. Diagonal preconditioners are trivial to construct and invert, but generally do not speed up the convergence of the conjugate gradient method significantly.

Incomplete Cholesky Preconditioner

The Cholesky factorization of a SPD matrix A into LL^T is a special case of an LU factorization. Performing a full LU factorization and then solving $A\mathbf{x} = \mathbf{b}$ by inverting in turn the lower triangular matrix L and the upper triangular matrix U is a direct method, the algorithm for constructing the Cholesky matrix L for an $N \times N$ symmetric matrix A is given in Algorithm 27.

Algorithm 27 Cholesky Factorization

Set $l_{11} = \sqrt{a_{11}}$

For $j = 2, .., N$

$\quad l_{j1} = a_{j1}/l_{11}$

end

For $i = 2, .., N-1$

$\quad l_{ii} = \left(a_{ii} - \sum_{k=1}^{i-1} l_{ik}^2\right)^{1/2}$

$\quad$ For $j = i+1, .., N$

$\quad\quad l_{ji} = \frac{1}{l_{ii}}\left(a_{ji} - \sum_{k=1}^{i-1} l_{jk}l_{ik}\right)$

$\quad$ end

end

$l_{nn} = \left(a_{nn} - \sum_{k=1}^{n-1} l_{nk}^2\right)^{1/2}$

When an incomplete Cholesky factorisation is performed only some of the entries of L

are computed, this can be done for example by only calculating the entries of L which fit the sparsity pattern of the original matrix A. In my experiments I make use of MATLAB's CHOLINC function with the drop tolerance option to construct incomplete Cholesky factorisations. After each column l_i of L has been computed all entries (with the exception of the diagonal entry l_{ii}) in that column less than $droptol\|l_i\|_2$ are dropped (set to zero) before the algorithm proceeds. The matrix $M = LL^T$, where LL^T comes from an incomplete Cholesky factorisation, can be used as a preconditioner for the conjugate gradient method. To invert M the lower triangular matrix L is first inverted followed by the upper triangular matrix L^T. The more entries kept the more accurate the incomplete Cholesky factorisation is and the better the convergence of the PCG method, however the more entries that are kept, the more expensive to construct and invert the preconditioner is. How much information to retain will depend on the difficulty and size of the problem to be solved.

Multigrid Preconditioning

The step $\mathbf{z} = M^{-1}\mathbf{r}$ in the PCG method can be seen as an approximate solve of the problem $A\mathbf{z} = \mathbf{r}$, since M is an approximation of A. One way to approximately solve $A\mathbf{z} = \mathbf{r}$ would be to apply one multigrid cycle to the system with zero initial guess, hence a multigrid method can be used as a preconditioner for the PCG method. If an efficient multigrid method for solving $A\mathbf{x} = \mathbf{b}$ exists, there will be no advantage in using it as a preconditioner for conjugate gradient, in fact the overall cost of solving the problem is likely to increase, however if, the convergence of the multigrid method is not satisfactory it can be accelerated by using it as a preconditioner within the conjugate gradient method.

Remark A.6.1 *If we wish to use multigrid as a preconditioner, then the multigrid method should preserve symmetry.*

A.6.4 General Steepest Descent and Global Optimization Methods

For a general minimization problem $\min_{\mathbf{x}} f(\mathbf{x})$ the steepest descent method involves taking on step $k+1$ a step in the direction of steepest descent $\mathbf{d}_k = -\nabla f(\mathbf{x}_k)$. In the case where f was a quadratic the step length α_k which minimized f along $\mathbf{d}_k$ could be found analytically, in the more general case α_k is determined using a line search procedure. The simplest line search is a backtracking algorithm in which the value of α_k is continually reduced by $\theta \in (0,1)$ until the following sufficient decrease condition is satisfied

$$f(\mathbf{x}_{k+1} - \alpha_k \nabla f(\mathbf{x}_k)) - f(\mathbf{x}_k) < -\lambda \|\nabla f(\mathbf{x}_k)\|_2^2. \tag{A.90}$$

where λ is typically 10^{-4}. This sufficient decrease condition basically says that the decrease in the function value is some fraction of the reduction predicted by approximating f around $\mathbf{x}_k$ by the linear model

$$f_k(\mathbf{x}) = f(\mathbf{x}_k) + \nabla f(\mathbf{x}_k)(\mathbf{x} - \mathbf{x}_k). \tag{A.91}$$

The general steepest descent algorithm with a backtracking line search is given in Algorithm 28.

Algorithm 28 Steepest Descent

Choose $\mathbf{x}_0$, set $k = 0$.
Choose $\theta \in (0,1)$, λ
While $\|\nabla f(\mathbf{x}_0)_k\|_2 > tol$
    $\mathbf{d}_k = -\nabla f(\mathbf{x}_k)$
    $\alpha_k = 1$
    While $f(\mathbf{x}_{k+1} + \alpha_k \mathbf{d}_k) - f(\mathbf{x}_k) > -\lambda \|\mathbf{d}_k\|_2^2$
        $\alpha_k \leftarrow \theta \alpha_k$
    end
    $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k$
    $k \leftarrow k + 1$
end

Remark A.6.2 *If the function f does not have a unique minimum, then the minimum found by the steepest descent method will depend on the initial guess $\mathbf{x}_0$*

The steepest descent method is part of a larger class of global minimization methods, based on quadratic models of f around $\mathbf{x}_k$

$$q_k(\mathbf{x}) = f(\mathbf{x}_k) + \nabla f(\mathbf{x}_k)^T(\mathbf{x} - \mathbf{x}_k) + 1/2(\mathbf{x} - \mathbf{x}_k)^T H_k(\mathbf{x} - \mathbf{x}_k), \tag{A.92}$$

where H_k is SPD. The search direction $\mathbf{d}_k$ is taken to be the value of $\mathbf{x} - \mathbf{x}_k$ which minimizes $q_k(\mathbf{x})$ i.e $\mathbf{d}_k$ satisfies

$$\nabla q_k(\mathbf{x}) = \nabla f(\mathbf{x}_k) + H_k \mathbf{d}_k = 0. \tag{A.93}$$

The search direction is therefore $\mathbf{d}_k = -H_k^{-1} \nabla f(\mathbf{x}_k)$. Given that H_k and hence H_k^{-1} is positive definite it is clear that $\nabla f(\mathbf{x}_k)^T \mathbf{d}_k < 0$ and hence $\mathbf{d}_k$ is a descent direction. In Newton's Method $H_k = H(\mathbf{x}_k)$, where $H(\mathbf{x}_k)$ is the Hessian of f at $\mathbf{x}_k$, in general $H(\mathbf{x}_k)$ is not guaranteed to be positive definite, unless $\mathbf{x}_k$ is sufficiently close to the minimum of f, see [65]. The general sufficient decrease condition for such methods is

$$f(\mathbf{x}_k + \alpha_k \mathbf{d}_k) - f(\mathbf{x}_k) < \lambda \alpha_k \nabla f(\mathbf{x}_k)^T \mathbf{d}_k. \tag{A.94}$$

Often the sufficient decrease condition is accompanied by another condition on α_k known as the curvature condition, which prevents unacceptably short steps along $\mathbf{d}_k$ from being taken

$$\nabla f(\mathbf{x}_k + \alpha_k \mathbf{d}_k)^T \mathbf{d}_k \geq \sigma \nabla f(\mathbf{x}_k)^T \mathbf{d}_k, \tag{A.95}$$

where the value of σ should be between λ and 1. A stronger version of this condition is

$$|\nabla f(\mathbf{x}_k + \alpha_k \mathbf{d}_k)^T \mathbf{d}_k| \leq \sigma |\nabla f(\mathbf{x}_k)^T \mathbf{d}_k|. \tag{A.96}$$

(A.94) with (A.95) are known as the Wolfe conditions and (A.94) with (A.96) the strong Wolfe conditions. A line search procedure to find a step length which satisfies the (strong) Wolfe conditions, involves two phases, a bracketing phase in which an interval guaranteed to contain points satisfying the Wolfe conditions is found and a second phase in which cubic or quadratic interpolation on $\phi(\alpha) = f(\mathbf{x}_k + \alpha \mathbf{d}_k)$ is applied continually until an acceptable point is found, for more details see, for example [49].

A.6.5 Nonlinear Conjugate Gradient

A version of the conjugate gradient method also exists for general nonlinear minimization problems, although it does not have the nice convergence properties of the conjugate gradient method for SPD linear systems. There are several changes which must be made to Algorithm 24, firstly the residual $\mathbf{r}_k$ is replaced by its nonlinear equivalent $\mathbf{r}_k = -\nabla f(\mathbf{x}_k)$ also $\mathbf{r}_{k+1}$ cannot be defined recursively from $\mathbf{r}_k$ and must simply be evaluated. Secondly the value of α_k cannot be determined analytically and is instead determined by a line search procedure, usually required to satisfy the strong Wolfe conditions. Finally, there are two different options for β_h (equivalent in the linear case since the residuals are orthogonal), either the Fletcher Reeves choice

$$\beta_{k+1}^{FR} = \frac{\nabla f(\mathbf{x}_{k+1})^T \nabla f(\mathbf{x}_{k+1})}{\nabla f(\mathbf{x}_k)^T \nabla f(\mathbf{x}_k)} \tag{A.97}$$

or the Polak Ribiere choice

$$\beta_{k+1}^{PR} = \frac{\nabla f(\mathbf{x}_{k+1})^T (\nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k))}{\nabla f(\mathbf{x}_k)^T \nabla f(\mathbf{x}_k)}. \tag{A.98}$$

The Fletcher-Reeves formula has a neater convergence theory, but the Polak-Ribiere formula generally performs better. In order to guarantee convergence of the Polak-Ribiere method β_{k+1}^{PR} can be replaced by $\max(\beta_{k+1}^{PR}, 0)$. For more details on nonlinear conjugate gradient, see [65, 87] and the references therein.

A.7 Multilevel Optimization

In this section I briefly review two optimization methods which employ multilevel techniques.

A.7.1 Multigrid Optimization based on FAS

In [73] Nash presents a Multigrid optimization algorithm based on the full approximation scheme for solving an unconstrained optimization problem

$$\min_{u_h} f_h(u_h). \tag{A.99}$$

on Ω^h. To clarify what this notation means we may have for example the problem

$$\min_u f(u) \text{ where } f(u) = \int_\Omega L(x, y, u(x,y), u_x, u_y) dx dy, \tag{A.100}$$

where u is some continuous variable on the bounded and open domain $\Omega \subset \mathbb{R}^2$. If Ω is discretized and the resulting grid Ω^h has grid spacing h in both directions and grid points at (ih, jh) then the discrete optimization problem is

$$\min_{u_h} f_h(u_h) \text{ where } f(u_h) = \sum_{(i,j)} L(ih, jh, u_{ij}, (u_x)_{ij}, (u_y)_{ij}), \tag{A.101}$$

where $(u_x)_{ij}$ and $(u_y)_{ij}$ are finite difference approximations of u_x and u_y at the grid point (i, j).

One step of Nash's multigrid method is denoted $u_h^{k+1} = MGOP(u_h^k, \min_{u_h} f_h(u_h))$ where $MGOP$ is defined recursively in Algorithm 29.

Algorithm 29 MGOP

$$u_h^{k+1} = MGOP(u_h^k, \min_{u_h} f_h(u_h))$$

1. If Ω^h is the coarsest grid, solve

   $$\min_{u_h} f_h(u_h)$$

   and return.

   Else: apply ν_1 steps of an optimization algorithm with initial guess u_h^k

   $$\bar{u}_h^k = OP^{\nu_1}(u_h^k, \min_{u_h} f_h(u_h))$$

2. Evaluate

   $$\bar{u}_H^k = I_H^h \bar{u}_h^k$$

   $$v = \nabla f_H(\bar{u}_H) - I_h^H(\nabla f_h(\bar{u}_h))$$

3. Apply multigrid method to solve: $\min_{u_H} f_H(u_H) - v^T u_H$ with initial guess $\bar{u}_H^k$

   $$\bar{u}_H^{k+1} = MGOP(\bar{u}_H^k, \min_{u_H} f_H(u_H) - v^T u_H)$$

4. Evaluate

   $$e_h = I_H^h(\bar{u}_H^{k+1} - \bar{u}_H^k)$$

5. Perform Line Search

   $$\bar{u}_h^{k+1} = \bar{u}_h^k + \lambda e_h$$

6. Apply ν_2 steps of an optimization algorithm with initial guess $\bar{u}_h^{k+1}$

   $$u_h^{k+1} = OP^{\nu_2}(\bar{u}_h^{k+1}, \min_{u_h} f_h(u_h))$$

In Algorithm 29 Ω^H is the next coarsest grid e.g Ω^{2h} and I_H^h and I_h^H are transfer operators for transferring grid functions between Ω^h and Ω^H. f_H denotes the original optimization problem discretized on the grid Ω^H.

Note that the coarse grid optimization problem is not simply

$$\min_{u_H} f_H(u_H) \tag{A.102}$$

but

$$\min_{u_H} f_H(u_H) - v^T u_H. \tag{A.103}$$

where $v = \nabla f_H(\bar{u}_H) - I_h^H(\nabla f_h(\bar{u}_h))$. If $\nabla f_h(u_h) = N_h(u_h) - g_h$ where $N_h(u_h)$ is some nonlinear operator, then

$$\begin{aligned}\nabla[f_H(u_H) - v^T u_h] &= N_H(u_H) - g_H - (N_h(\bar{u}_H) - g_H) + I_h^H(N_h(\bar{u}_h) - g_h) \\ &= N_H(u_H) - [N_H(\bar{u}_H) + I_h^H(g_h - N_h(\bar{u}_h))]\end{aligned} \tag{A.104}$$

which is the equivalent coarse grid problem in the original FAS method presented in Algorithm 5.

The following theorem establishes that under certain assumptions the coarse grid correction is a descent direction.

Theorem A.7.1 *If the following conditions hold e_h is a descent direction for f_h at $\bar{u}_h^k$.*

1. *The optimization problem is convex.*
2. $I_H^h = C(I_h^H)^T$
3. *The coarse grid problem is solved accurately enough. i.e*

$$\nabla(f_H(u_H) - v^T u_H)(\bar{u}_H^{k+1}) = \nabla f_H(\bar{u}_H^{k+1}) - (\nabla f_H(\bar{u}_H^k) - I_h^H \nabla f_h(\bar{u}_h^k)) = \epsilon$$

where ϵ is small enough.

Proof

By definition

$$\nabla f_h(\bar{u}_h^k)^T e_h = \nabla f_h(\bar{u}_h^k)^T (I_H^h e_H). \tag{A.105}$$

By condition (2) this is equivalent to

$$\nabla f_h(\bar{u}_h^k)^T e_h = C(I_h^H \nabla f_h(\bar{u}_h^k))^T e_H. \tag{A.106}$$

By condition (3) this is equivalent to

$$\nabla f_h(\bar{u}_h^k)^T e_h = C(\nabla f_H(\bar{u}_H^k) - \nabla f_H(\bar{u}_H^{k+1}) + \epsilon)^T e_H. \tag{A.107}$$

Using the mean value theorem we have

$$\nabla f_H(\bar{u}_H^{k+1}) - \nabla f_H(\bar{u}_H^k) = \nabla^2 f_H(\zeta)(\bar{u}_H^{k+1} - \bar{u}_H^k) = \nabla^2 f_H(\zeta) e_H. \tag{A.108}$$

where $u^k < \zeta < u^{k+1}$. Substituting into (A.107) we get

$$\nabla f_h(\bar{u}_h^k)^T e_h = C(-e_H^T \nabla^2 f_H(\zeta) e_H + \epsilon^T e_H). \tag{A.109}$$

Given that the optimization problem is convex the first term in the right hand side is negative and given that ϵ is small the second term is negligible. Hence e_h is a decent direction. ∎

If e_h is a descent direction then the combination of the multigrid method *MGOP* with an optimization method which is globally convergent will result in a globally convergent method. In [73] Nash shows that e_h can be expected to be a descent direction even if the coarse grid problem is not solved to high accuracy, he also extends the method to non-convex problems.

A.7.2 Chan and Chen's Multilevel Optimization

In [25] Chan and Chen propose a multilevel optimization method which uses local optimization and correction via coarse grids. The method is applied to the discrete Total Variation denoising optimization problem in 1 and 2 dimensions. A brief outline of the method is given below.

Given a discrete optimization problem

$$\min_{u_h} f(u_h) \tag{A.110}$$

where u_h is a grid function on the finest (cell-centered) grid Ω^h, several steps of a local optimization method are applied to the problem to generate an approximation $\tilde{u}_h$. Each step of the local optimization method involves cycling through the grid points in a Gauss-Seidel fashion, solving at each grid point the local optimization problem $\min_{u_{i,j}} f_{i,j}(u_{i,j})$ which results from freezing all non (i,j) components of u_h at their current value. This problem is either solved analytically (if possible) or via iterations.

Once $\tilde{u}_h$ has been found a multilevel algorithm, utilising standard coarsened, coarse grids $\Omega^{2h}, \Omega^{4h}, .., \Omega^{ph}$ with $p = 2^L$, is employed. The algorithm is given in Algorithm 30.

Algorithm 30 Multilevel Optimization

Find $\tilde{u}_h$ by applying a local minimization method to $\min_{u_h} f(u_h)$

For k=1:L

$l = 2^k$

$\hat{c}_{lh} = \text{argmin}_{c_{lh}} f(\tilde{u}_h + P_{lh}^h c_{lh})$

$\tilde{u}_h = \tilde{u}_h + P_{lh}^h \hat{c}_{lh}$

end

In Algorithm 30 the interpolation operator P_{lh}^{h} maps grid functions on Ω^{lh} to grid functions on Ω^{h}. Assuming that the grids are cell-centered P is defined as the operator which assigns the value at a coarse grid cell, to all fine grid cells contained within it e.g for $\Omega \in \mathbb{R}^2$, P_{2h}^{h} is the cell-centered interpolation operator seen in equation (2.141). The solution of the minimization problem $\min_{c_{lh}} f(\tilde{u}_h + P_{lh}^{h} c_{lh})$ on Ω^{lh} involves $1/(l^d)$ as many unknowns as the fine grid problem where d is the dimension. $\hat{c}_{lh}$ is found using several steps of a local optimization method, as on the finest grid. The whole procedure is iterated until some stopping criteria is satisfied.

In addition the same approach has also been employed using a nonstandard coarse grid Ω^{H}, which is defined based on the properties of the fine grid approximation $\tilde{u}$. The principle is exactly as above with an interpolation operator P_{H}^{h} mapping grid functions on Ω^{H} to grid functions on Ω^{h} being required.

Appendix B

Cost Estimate For the AMG Method

In this appendix we present an analysis of the costs associated with implementing the AMG method in MATLAB with the standard C/F splitting algorithm and direct interpolation as detailed in §2.7. Due to the automatic nature of AMG the analyis is non trivial and certain assumptions will have to be made. We will use the following notation in the cost analysis that follows

$$\begin{aligned}
n^k &= \text{ number of points on level } k \\
n^{F^k} &= \textit{number of fine points on level } k \\
n^{C^k} = n^{k+1} &= \text{ number of coarse points on level } k \\
\gamma_{A^k} &= \text{ max number of entries in a row of } A^k \\
\gamma_{R^k} &= \text{ max number of entries in a row of } R^k = I_k^{k+1}(\text{ restriction } k \to k+1) \\
\gamma_{P^k} &= \text{ max number of entries in a row of } P^k = I_{k+1}^{k}(\text{ prolongation } k+1 \to k)
\end{aligned} \qquad \text{(B.1)}$$

Although the flop cost in MATLAB of finding all the nonzero entries in a set or finding the maximum value in set is zero, below we assume that the costs are approximately equal to twice the size of the set. To justify the assumptions made we present below the cost in cputime of finding the nonzero entries in a random vector of size 10^7 using MATLAB's find function, finding the maximum value in a random vector of size 10^7 using MATLAB's max function and multiplying component wise two vectors of size 10^7 i.e an operation which costs 10^7 flops.

operation	cpu
find	0.76
max	0.92
multiply	.50

We see that our assumptions appear reasonable as an upper bound.

B.1 The AMG setup phase

B.1.1 Cost of Finding Neighbours and Strong Connections

To find N_i we must search row i of the matrix for non-zero, non-diagonal entries. In reality the position of all non-zero entries in a sparse matrix can be found very cheaply in MATLAB so we ignore this cost here. Once we have the set of neighbours we have to find the maximum value of $-a_{ij}$ over $j \in N_i$ and then find $S_i = \{j \in N_i | -a_{i,j} \geq \theta \max_{k \neq i}(-a_{ik})\}$. We have to search through the set $\{-a_{ij} | j \in N_i\}$ to find the maximum, and then search the set again to find the entries greater than θ times the maximum, since an upper bound on the size of N_i is $\gamma_{A^k} - 1$ an upper bound for this cost is $4(\gamma_{A^k} - 1)$. An upper bound on the cost of finding all N_i and S_i on level k is therefore

$$n^k \left[4(\gamma_{A^k} - 1)\right]. \tag{B.2}$$

B.1.2 The C/F splitting algorithm

We assume that the size of $C^k = 1/4n^k$ i.e we assume standard coarsening. We therefore go around the C/F splitting loop $1/4n$ times and on average each coarse point defines 3 fine points i.e the size of $S_i^T \cap U$ is 3. Although in our experience based on cpu analysis similar to above a reasonable assumption is that the cost of finding $S \cap U$ using MATLAB's intersect function (in fact a slight modification of it) where S is a small set, is around 3 times the size of U we can in our current implementation find $S_i^T \cap U$ simply by finding the nonzero entries of a set the size of S_i^T, which we assume below has size 4. Based on these assumptions we have the following estimate for the cost of performing the C/F split on level k.

$$\sum_{m=0}^{1/4n^k - 1} \left[2(n^k - 4m) + 8\right] + 3\,(8 + 4) + 8 + 4. \tag{B.3}$$

The fisrt term comes from finding $i \in U$ with maximal λ_i and then finding $j \in S_i^T \cap U$, the second term comes from finding $l \in S_j \cap U$ setiing $\lambda_l = \lambda_l + 1$ for each $j \in S_i^T \cap U$ (we assume that the size of S_j and $S_j \cap U$ is 4) the third term comes from finding $j \in S_i \cap U$ and the last term setting $\lambda_j = \lambda_j - 1$ (again we assume S_i and $S_i \cap U$ have 4 members). An approximate cost estimate is therefore

$$(1/4n^k)(2n^k + 56) - 8 \sum_{m=0}^{1/4n^k - 1} m \approx 1/2(n^k)^2 + 14n^k - 8(1/8n^k)(1/4n^k) = 1/4(n^k)^2 + 14n^k. \tag{B.4}$$

B.1.3 The Direct Interpolation

On level k direct interpolation involves finding for each $i \in F^k$

$$\alpha_i = \frac{\sum_{j \in N_i} a_{ij}^-}{\sum_{k \in P_i} a_{ik}^-} \text{ and } \beta_i = \frac{\sum_{j \in N_i} a_{ij}^+}{\sum_{k \in P_i} a_{ik}^+}. \tag{B.5}$$

or if P_i^+ is empty finding α_i and setting $a_{ii} = a_{ii} + \sum_{j \in N_i} a_{ij}^+$. Then defining the interpolation weights as

$$w_{ik} = \begin{cases} -\alpha_i a_{ik}/a_{ii} & k \in P_i^- \\ -\beta_i a_{ik}/a_{ii} & k \in P_i^+ \end{cases}. \tag{B.6}$$

where $P_i = C_i$.

Denote by n^{P^k} the max size of P_i over all $i \in F^k$ and by n^{N^k} the max size of N_i over all $i \in F^k$ then an upper bound for the cost of the direct interpolation on level k is

$$n^{F^k}\left[n^{N^k} + 3n^{P^k}\right]. \tag{B.7}$$

B.1.4 Cost of Forming the Galerkin Matrix RAP

Given that we expect that A and P will be very sparse most row_A, $column_P$ multiplications will produce zeros. A cost of $\gamma_{A^k}^2 n^k$ is a reasonable assumption for the cost of evaluating $A^k P^k$ similarly $\gamma_{R^k}^2 n^{C^k}$ is a reasonable assumption for the cost of multiplying $A^k P^k$ by R^k.

B.1.5 Overall Cost

Putting all of this together and assuming standard coarsening and also taking γ_A, γ_R, γ_P as the maximum value over all levels of γ_{A^k}, γ_{R^k}, γ_{P^k} noting that $n^{N^k} = \gamma_{A^k} - 1$ and assuming that $n^{P^k} = 4$ we have as an estimate for the cost of an AMG setup phase the following:

$$\begin{aligned}
&\sum_k 4(\gamma_{A^k} - 1)n^k + 1/4(n^k)^2 + 14n^k + n^{F^k}(\gamma_A + 11) + \gamma_A^2 n^k + \gamma_R^2 n^{C^k} \\
&= \sum_k 1/4(n^k)^2 + 4(\gamma_{A^k} - 1)n^k + 14n^k + 3/4n^k(\gamma_A + 11) + \gamma_A^2 n^k + \gamma_R^2 1/4n^k \\
&\approx \sum_{l=0}^{\infty} 1/4(1/16)^l n^2 + 3/4(1/4)^l(17/3\gamma_A + 4/3\gamma_A^2 + 1/3\gamma_R^2 + 8)n \\
&= 17/64n^2 + (17\gamma_A + 4\gamma_A^2 + \gamma_R^2 + 24)n/3.
\end{aligned} \tag{B.8}$$

B.2 The V-Cycle

Once the setup phase has been performed the costs associated with the V-cycle on level k are

1. The cost of 4 Gauss-Seidel steps each of which involves solving $Mv^{new} = (Nv^{old} + f)$ where M is the lower triangular part of A^k including the diagonal and N is the upper triangular part of A^k. Assuming that γ_A^k is small and that the entries of A^k are split evenly about the diagonal and using the fact that the cost of inverting or multiplying by a $n^k \times n^k$ triangular matrix with a small number, say σ, of entries on each row is approximately $(2\sigma - 1)n^k$ we have $4\left[2(\gamma_A^k - 1)n^k + n^k\right]$ as an estimate for the cost of the Gauss-Seidel steps.

2. The cost of the residual calculation $r^k = f^k - A^k v^k$, an upper bound for which is $n^k + 2\gamma_{A^k} n^k$.

3. The cost of restricting the residual, an upper bound for which is $2\gamma_{R^k} n^{C^k}$.

4. The cost of interpolating the error back from the coarse grid and correcting, an upper bound for which is $2\gamma_{P^k} n^{F^k} + n^{C^k} + n^k$.

Putting all of this together we have as an upper bound for the cost of a V-cycle:

$$\sum_{l=0}^{\infty}(1/4)^l \left[10\gamma_A + 1/2\gamma_R + 3/2\gamma_P - 7/4\right] n$$
$$= (40\gamma_A + 2\gamma_R + 6\gamma_P - 7)\, n/3. \tag{B.9}$$

B.3 Cost Comparison

From observations we take $\gamma_A = 20$, $\gamma_R = 15$ and $\gamma_P = 10$ we then have an approximate cost for the V-cycle of $300n$ and an approximate cost for the setup phase of $17/64n^2 + 730n$. If we take $n = 256^2$ we get that the cost of a V-cycle is approximately 2×10^7 and the cost of a setup is approximately 1.19×10^9, around 60 times the cost of the V-cycle. In §5.3 we present the idea of recycling setup data, essentially on some fixed point steps we replace a whole setup with a simple evaluation of RAP using stored interpolation and restriction operators. The cost of the recycling will be $(4/3\gamma_A^2 + 1/3\gamma_R^2)n$ which is approximately $610n$, making the cost of a V-cycle + recycling around $910n$ which for $n = 256^2$ is 6.0×10^7. In reality the dominant cost is the cost on the finest level, on this level we know $\gamma_A = 5$ and we can assume $\gamma_R = 5$ and $\gamma_P = 4$. With these parameters we get that a V-cycle costs approximately $75n$ and a setup costs approximately $17/64n^2 + 80n$, while a recycle costs around $40n$, for $n = 256^2$ this is 5.0×10^6 flops for a V-cycle, 2.8×10^6 for a recycle and 1.15×10^9 for a setup. When we actually measure the flops associated with a V-cycle and a recycle we get that a V-cycle costs around 1.2×10^7 flops and a recycle costs 6.1×10^6, which is somewhere in between the two estimates made above.

This is of course only an estimate of the sort of costs involved based on some assumptions about the flops cost of certain operations which register no flop cost in MATLAB.

Bibliography

[1] R. Acar and C. R. Vogel, *Analysis of Bounded Variation Penalty Methods for Ill-Posed Problems*, Inverse Problems **10** (1994), pp. 1217-1229.

[2] R. A. Adams and J. J. F. Fournier, *Sobolev Spaces, Second Edition*, Elsevier Science Ltd, Oxford, 2003.

[3] K. G. Binmore, *The Foundations of Analysis: A Straightforward Introduction, Book1: Logic Sets and Numbers*, Cambridge University Press, Cambridge, 1980.

[4] K. G. Binmore, *The Foundations of Analysis: A Straightforward Introduction, Book2: Topological Ideas*, Cambridge University Press, Cambridge, 1981.

[5] P. Blomgren, *Total Variation Methods for Restoration of Vector Valued Images*, PhD Thesis, UCLA, 1998.

[6] P. Blomgren, T. F. Chan and P. Mulet, *Extensions to Total Variation Denoising*, Proc. SPIE 97, San Diego.

[7] P. Blomgren, T. F. Chan, P. Mulet, L. Vese and W. L. Wan, *Variational PDE models and methods for image processing*, in: *Research Notes in Mathematics*, 420: 43-67. Chapman & Hall/CRC, 2000.

[8] J. M. Borwein and A. S. Lewis, *Convex Analysis and Nonlinear Optimization*, Springer-Verlag, New York, 2000.

[9] A. Brandt, S. F. McCormick and J. Ruge, *Algebraic Multigrid (AMG) for Automatic Multigrid Solution with Application to Geodetic Computations*, Institute for Computational Studies, Fort Collins, Colarado, 1982.

[10] U. Brechtken-Manderscheid, *Introduction to the Calculus of Variations*, Chapman and Hall, London, 1991.

[11] M. Brezina, R. Falgout, S. Maclachlan, T. Manteuffel, S. Mccormick and J. Ruge, *Adaptive Smoothed Aggregation*, SIAM J. Sci. Comput., **25** (2004), No. 6, pp.1896-1920.

[12] W. Briggs, *A Multigrid Tutorial*, SIAM, Philadelphia, 1987.

[13] A. Bruhn, J. Weickert, T. Kholberger and T. Schnorr, *A Multigrid Platform for Real-Time Motion Computation with Discontinuity Preserving Variational Methods*, Technical Report No. 136, Department of Mathematics, Saarland University, Saarbrucken, Germany, May 2005.

[14] R. Burden and D. Faires, *Numerical Analysis, sixth ed,* ITP publishing, London, 1997.

[15] M. Burger, *Infinite-dimensional Optimization and Optimal Design*, Lecture Notes, 285J, Department of Mathematics, UCLA, 2003.

[16] M. Burger, S. Osher, J. Xu and G. Gilboa, *Nonlinear Inverse Scale Space Methods for Image Restoration*, UCLA CAM report, 2005.

[17] J. L. Carter, *Dual Methods for Total Variation Based Image Restoration*, PhD Thesis, UCLA, 2001.

[18] A. Chambolle, *An algorithm for Total Variation Minimization and Applications*, Journal of Mathematical Imaging and Vision., **20** (2004) pp. 89-97.

[19] A. Chambolle and P-L. Lions, *Image Recovery via Total Variation Minimization and Related Problems*, Numer. Math., **76** (1997), pp. 167-188.

[20] R. H. Chan, T. F. Chan and W. L. Wan, *Multigrid for Differential-Convolution Problems Arising from Image Processing*, Proceedings of the Workshop on Scientific Computing 97, Springer-Verlag, 1997.

[21] R. Chan, C. Hu and M. Nikolova, *An Iterative Procedure for Removing Random-Valued Noise*, IEEE Signal Proc. Letters **11** (2004), pp.921-942.

[22] T.Chan, *An optimal Circulant Preconditioner for Toeplitz Systems*, SIAM J. Sci. Stat. Comput. **9** (1988), pp. 766-771.

[23] T. F. Chan and W. L. Wan, *Robust multigrid methods for nonsmooth coefficient elliptic linear systems*, Journal of Computational and Applied Mathematics, **123** (2000), pp.323-352.

[24] R. H. Chan, T. F. Chan and C-K. Wong, *Cosine Transform Based Preconditioners for Total Variation Deblurring*,UCLA CAM report 97-44, 1997.

[25] T. Chan and K. Chen, *On a Nonlinear Multigrid Algorithm with Primal Relaxation for the Image Total Variation Minimization*, to appear in SIAM J. Multiscale Modeling and Simulation (MMS), 2006.

[26] T. Chan and S.Esedoglu, *Aspects of Total Variation Regularized L^1 Function Approximation*, UCLA CAM report, 2004.

[27] T. Chan, S. Esedoglu, F.Park and A. Yip, *Recent Developments in Total Variation Image Restoration*, Mathematical Models in Computer Vision: The Handbook, 2004.

[28] T. F. Chan, S. Esedoglu and F.E. Park, *Image Decomposition Combining Staircase Reduction and Texture Extraction*, UCLA CAM Report, 2005.

[29] T. F. Chan, G. H. Golub and P. Mulet, *A Nonlinear Primal-Dual Method for Total Variation-Based Image Restoration*, SIAM J. Sci. Comput., **20** (1999), pp. 1964-1977.

[30] T. F. Chan, A. Marquina and P. Mulet, *Second Order Differential Functionals in Total Variation-Based Image Restoration*, UCLA CAM Report 98-35, 1998.

[31] T. F. Chan and L. Vese, *Active Contours Without Edges*, UCLA CAM report 98-53, 1998.

[32] T. F. Chan, R. H. Chan and H. M. Zhou, *Continuation Method for Total Variation Denoising Problems*, UCLA CAM Report, 1995.

[33] T. Chan and P. Mulet, *Iterative Methods for Total Variation Image Restoration*, UCLA CAM Report 96-38, 1996.

[34] Q. Chang and I. Chern, *Acceleration Methods for Total Variation Based Image Denoising*, SIAM J. Sci. Comput., **25** (2003), pp. 982-994.

[35] Q. S. Chang, Y. S. Wong and H. Fu, *On the algebraic multigrid method*, J. Comput. Phys., **125** (1996), pp.279-292.

[36] Q. Chang, W. Wang and J. Xu, *A Method for Total Variation-Based Reconstruction of Noisy and Blurred Image*, Submitted to the proceedings of the International Conference on PDE-Based Image Processing and Related Inverse Problems, Oslo, August 2005.

[37] K. Chen, *Matrix Preconditioning Techniques and Applications*, Series: Cambridge Monographs on Applied and Computational Mathematics (No. 19), Cambridge University Press, UK, 2005.

[38] K. Chen and J. Savage, *On Two New and Nonlinear Iterative Methods for Total-Variation Based Image Deblurring*, in Proceedings of the Fifth UK Conference on Boundary Integral Methods, K. Chen, ed., 2005.

[39] K. Chen and X-C. Tai, *A Nonlinear Multigrid Method for Curvature Equations related to Total Variation Minimization*, UCLA CAM report, 2005.

[40] Y. Chen, S. Levine and M.Rao, *Functionals with p(x)-Growth in Image Restoration*, Duquesne Univ. Dept. of Math and Comp. Sci. Tech. Report, 2004.

[41] Y. Chen, S. Levine and J. Stanich, *Image Restoration via Nonstandard Diffusion*, Duquesne Univ. Dept. of Math and Comp. Sci. Tech. Report, 2004.

[42] S. Cheng, *Multi-Linear Maps and Taylor's Theorem in Higher Dimensions*, February 2006.

[43] J. E. Dendy, *Black Box Multigrid*, Journal of Computational Physics, **48** (1982), pp.366-386.

[44] P. M. De Zeeuw, *Matrix-Dependent Prolongations and Restrictions in a Blackbox Multigrid Solver*, Journal of Computational and Applied Mathematics **33** (1990), pp.1-27.

[45] B. Diskin, J. L. Thomas and R. E. Mineck, *On Quantitative Analysis Methods for Multigrid Solutions*, SIAM J. Sci. Comput., **27** (2005), No. 1, pp.108-129.

[46] D. C. Dobson and C. R. Vogel, *Convergence of an Iterative Method for Total Variation Denoising*, SIAM J. Numer. Anal., **34** (1997), No. 5, pp.1779-1791.

[47] H. Engl, M. Hanke and A. Neubauer, Regularization of Inverse Problems, Kluwer, Dordrecht, 1996.

[48] C. Frohn-Schauf, S. Henn and K. Witsch, *Nonlinear Multigrid Methods for Total Variation Denoising*, Comput. Vision. Sci., **7** (2004), pp.199-206.

[49] R. Fletcher, *Practical Methods of Optimization*, Wiley, Chichester, 1987.

[50] R. M. Freund, *Introduction to Optimization and Optimality Conditions for Unconstrained Problems*, Massachusettes Institute of Technology, 2004.

[51] R. M. Freund, *Optimality Conditions for Constrained Optimization Problems* , Massachusettes Institute of Technology, 2004.

[52] E. Giusti, *Minimal Surfaces and Fuctions of Bounded Variation*, Birkhauser, Boston, 1984.

[53] G. Golub and C. van Loan, *Matrix Computations, third ed.*, The John Hopkins University Press, London, 1996.

[54] D. H. Griffel, *Applied Functional Analysis*, Ellis Horwood Limited, Chichester, 1981.

[55] M. Hanke and J. G. Nagy, *Restoration of Atmospherically Blurred Images by Symmetric Indefinite Conjugate Gradient Techniques*, Inverse Problems **12** (1996), pp. 157-173.

[56] P. C. Hansen, *Rank-Deficient and Discrete Ill-Posed Problems*, SIAM Philadelphia, 1998.

[57] V. E. Henson, *Multigrid Methods for Nonlinear Problems: An Overview*, Center for Applied Scientific Computing Lawrence Livermore Laboratory.

[58] K. Ito and K. Kunisch, *An Active-Set Strategy Based on the Augmented Lagrangian Formulation for Image Restoration*, M2AN Mathematical Modelling and Numerical Analysis, **33**, No.1 (1999), pp.1-21.

[59] K. Ito and K. Kunisch, *BV-Type Regularization Methods for Convoluted Objects with Edge Flat and Grey Scales*, Inverse Problems **16** (2000), pp.909-928.

[60] K. Joo and S. Kim, *PDE-Based Image Restoration, I: Anti-Staircasing and Anti-Diffusion*, University of Kentucky Technical Report, 2003.

[61] T. Karkkainen and K. Majava, *Nonmonotone and Monotone Active-Set Methods for Image Restoration*, Journal of Optimization Theory and Applications, **106**, No. 1 (2000), pp.61-105.

[62] T. Karkkainen, K. Majava and M. M. Makela, *Comparisons of Formulations and Solution Methods for Image Restoration Problems*, Tech. Rep. B 14/2000, Department of Mathematical Information Technology University of Jyvaskyla, 2000.

[63] T. Karkkainen and K. Majava, *Semi-Adaptive Optimization Methodology for Image Denoising* , IEE Proc. Vis. Image Signal Process., **152**, No. 5, October 2005, pp.553-560.

[64] C. T. Kelley, *Iterative Methods for Linear and Nonlinear Equations*, SIAM, Philadelphia, 1995.

[65] C. T. Kelley, *Iterative Methods for Optimization*, SIAM, Philadelphia, 1999.

[66] M. Khalil and P. Wesseling, *Vertex-Centered and Cell-Centered Multigrid for Interface Problems*, Journal of Computational Physics **98** (1992), pp.1-10.

[67] S-H. Lee and J. K. Seo, *Noise Removal with Gauss Curvature Driven Diffusion*, IEEE transactions on Image Processing, 2005.

[68] S. E. Levine, *An Adaptive Variational Model for Image Decomposition*, Duquesne Univ. Dept. of Math and Comp. Sci. Tech. Report, 2004.

[69] M. Lysaker and X-C. Tai, *Interactive Image Restoration Combining Total Variation Minimization and a Second Order Functional*, Int. J. Comp Vision, to appear.

[70] M. Lysaker, A. Lundervold and X-C. Tai, *Noise Removal Using Fourth-Order Partial Differential Equation with Applications to Medical Magnetic Resonance Images in Space and Time*, IEEE Trans. Image Processing, **12** (2003).

[71] A. Marquina and S. Osher, *Explicit Algorithms for a New Time Dependant Model Based on Level Set Motion for Nonlinear Deblurring and Noise Removal*, SIAM J. Sci. Comput., **22** (2000), pp. 387-405.

[72] A. R. Mitchell and D. F. Griffiths, *The Finite Difference Method in Partial Differential Equations*, John Wiley and Sons, Chichester, 1980.

[73] S. G. Nash, *A Multigrid Approach to Discretized Optimization Problems*, Journal of Optimaization Methods and Software, **14** (2000), pp. 99-116.

[74] Markus von Oehsen, *Multiscale Methods for Variational Image Denoising*, Logos-Verl, Berlin, 2002.

[75] S. Osher, M. Burger, D. Goldfarb, J. Xu and W. Yin, *An Iterative Regularization Method for Total Variation Based Image Restoration*, Multiscale Model. and Simul., **4** (2005) pp. 460-489.

[76] S. Osher, A. Sole and L. Vese, *Image Decomposition and Restoration Using Total Variation Minimization and the H^{-1} Norm*, Multiscale Model Simul., **1** (2003), pp.349-370.

[77] P. Perona and J. Malik, *Scale-Space and Edge Detection Using Anisotropic Difussion*, IEEE Transactions on Pattern Analysis and Machine Intelligence, **12**, No. 7, July 1990, pp.629-639.

[78] L. I. Rudin, S. Osher and E. Fatemi, *Nonlinear Total Variation Based Noise Removal Algorithms*, Physica D, **60** (1992), pp. 259-268.

[79] K. L. Riley, *Two-Level Preconditioners for Regularized Ill-Posed Problems*, PhD Thesis, Montana State University-Bozeman, 1999.

[80] J. W. Ruge and K. Stuben, *Algebraic Multigrid*, In S. F. McCormick (Ed.), Multigrid Methods, SIAM, Philadelphia, 1987.

[81] Y. Saad, *Iterative Methods for Sparse Linear Systems, 2nd ed.*, SIAM, Philadelphia, 2003.

[82] H. Sagan, *Introduction to the Calculus of Variations*, Dover Publications, New York, 1992.

[83] J. Savage and K. Chen, *An improved and accelerated Nonlinear Multigrid Method for Total-Variation Denoising*, International Journal of Computer Mathematics., **82**, No 8, August 2005, pp.1001-1015.

[84] J. Savage and K. Chen, *On Multigrids for Solving a Class of Improved Total Variation Based Staircasing Reduction Models*, To appear in the proceedings of the International Conference on PDE-Based Image Processing and Related Inverse Problems, CMA Oslo, Springer-Verlag, 2006.

[85] J. Savage and K. Chen, *An Accelerated Algebraic Multigrid Algorithm for Total-Variation Denoising*, Submitted to BIT, 2006.

[86] O. Scherzer and J. Weickert, *Relations Between Regularization and Diffusion Filtering*, Journal of Mathematical Imaging and Vision **12** (2000), pp.43-63.

[87] J. R. Shewchuk, *An Introduction to the Conjugate Gradient Method Without the Agonizing Pain*, Carnegie Mellon University, Pittsburgh, 1994.

[88] G. D. Smith, *Numerical Solution of Partial Differential Equations: Finite Difference Mathods*, Clarendon Press, Oxford, 1985.

[89] D. M. Strong, J-F. Aujol and T. F. Chan, *Scale Recognition, Regularization Parameter Selection and Meyer's G Norm in Total Variation Regularization*, UCLA CAM Report, 2004.

[90] D. M. Strong and T. F. Chan, *Exact Solutions to Total Variation Regularization Problems*, UCLA CAM Report, 1996.

[91] D. M. Strong and T. F. Chan, *Edge Preserving and Scale Dependant Properties of Total Variation Regularization*, Inverse Problems **19** (2003) pp. 165-187.

[92] U. Trottenberg, C. Oosterlee and A. Schuller, *Multigrid*, Academic Press, London, 2001.

[93] C. R. Vogel and M. E. Oman, *Iterative Methods for Total Variation Denoising*, SIAM J. Sci. Comput., **17** (1996), pp. 227-238.

[94] C. R. Vogel and M. E. Oman, *Fast Total Variation-Based Image Reconstruction* , In Proceedings of the ASME Symposium on Inverse Problems, 1995.

[95] C. R. Vogel and M. E. Oman, *Fast, Robust Total Variation-Based Reconstruction of Noisy Blurred Images*, IEEE Transactions on Image Processing, **7**, 1998, pp.813-824.

[96] C. R. Vogel, *A Multigrid Method for Total Variation Based Image Denoising*, Computaion and Control IV, Bikhauser, 1995.

[97] C. R. Vogel, *Computational Methods for Inverse Problems*, SIAM, Philadelphia, 2002.

[98] L. Vese and S. Osher, *Modelling Textures with Total Variation Minimization and Oscillating Patterns in Image Processing*, UCLA CAM Report 02-19, 2002.

[99] C. Wagner, *Introduction to Algebraic Multigrid*, Course Notes of an Algebraic Multigrid Course at the University of Heidelberg in the Winter semester 1998/1999.

[100] T. Washio and C. Oosterlee, *Krylov Subspace Acceleration for Nonlinear Multigrid Schemes*, Electronic Transactions on Numerical Analysis., **6** (1997), pp. 271-290.

[101] J. Weickert, *Recursive Seperable Schemes for Nonlinear Diffusion Filters*, B. ter Haar Romeny, L. Florack, J. Koenderink, M. Viergever (Eds.), Scale-Space Theory in Computer Vision, Lecture Notes in Computer Science, Vol. 1252, Springer, Berlin, pp. 260-271, 1997.

[102] P. Wesseling, *An Introduction to Multigrid Methods*, Wiley, Chichester, 1992.

[103] I. Yavneh and G. Dardyk, *A Multilevel Nonlinear Method*, SIAM J. Sci. Comput., **28**, No. 1, pp.24-46.

[104] W. Yin, D. Goldfarb and S. Osher, *Image Cartoon-Texture Decomposition and Feature Selection using the Total Variation Regularized L^1 Functional*, UCLA CAM report, 2005.

[105] N. Young, *An Introduction to Hilbert Space*, Cambridge University Press, Cambridge, 1988.