Numerical Solution of Weather and Climate Systems

submitted by

Sean Buckeridge

for the degree of Doctor of Philosophy

of the

University of Bath

Department of Mathematical Sciences

November 2010

Summary

The subject of this thesis is an optimal and scalable parallel geometric multigrid solver for elliptic problems on the sphere, crucial to the forecasting and data assimilation tools used at the UK Met Office. The optimality of multilevel techniques for elliptic problems makes them a suitable choice for these applications. The Met Office uses spherical polar grids which, although structured, have the drawback of creating strong anisotropies near the poles. Moreover, a higher resolution in the radial direction introduces further anisotropies, and so modifications to the standard multigrid relaxation and coarsening procedures are necessary to retain optimal efficiency.

Since the strength of anisotropy varies, we propose a non-uniform strategy, coarsening the grid only in regions that are sufficiently isotropic. This is combined with line relaxation in the radial direction. The inspiration of a non-uniform coarsening strategy comes from algebraic multigrid (AMG) methods, which have already demonstrated the success of this technique. The large setup cost required by AMG, however, means that geometric multigrid methods are typically more efficient. Since all the problems dealt with in this thesis have the convenient feature of a grid-aligned anisotropy, we can exploit this and retain the efficiency of a geometric approach to produce an optimal method that surpasses the costs of AMG.

We demonstrate both theoretically and experimentally that the non-uniform geometric multigrid method is robust with respect to grid refinement, and therefore an optimal method for solving elliptic problems on the sphere. The theory, which exploits the grid-aligned anisotropy in these problems, is based on a separation of coordinate directions using a tensor product approach, as done in [14], and using existing theory from Hackbusch et al [44].

The advantages of the method are shown experimentally on model problems, both sequentially and in parallel, and show robustness and optimal efficiency of the method with constant convergence factors of less than 0.1. It substantially outperforms Krylov subspace methods with one-level preconditioners and the `BoomerAMG` implementation of AMG on typical grid resolutions used at the Met Office. The parallel implementation scales almost optimally on up to 256 processors, so that a global solve of 3D problems with a maximum horizontal resolution of about 10km and 3×10^9 unknowns takes about 60 seconds.

The non-uniform multigrid method is also applicable to certain elliptic problems arising in a new development within data assimilation, called the "potential vorticity (PV)-based control variable transform". These problems are so ill-conditioned that the solvers currently used at the Met Office fail to converge to a solution. However, using the non-uniform multigrid method as a preconditioner to a Krylov subspace method, it has been possible not only to solve these problems, but to solve them almost optimally.

Acknowledgements

I would like to begin by thanking my main PhD supervisor, Robert Scheichl, who tirelessly supported me with precise, efficient and detailed guidance throughout the project. His enthusiasm, commitment and expertise are second to none, and I thank him for the sheer amount of time and effort he has dedicated to ensure the success of this project. Thank you to Chris Budd for his useful advise and support during my PhD and to John Thuburn for his input from the University of Exeter. Thank you also to my external examiner Andy Wathen for his comments and interest on my work (and, of course, for passing me).

I would also like to thank the team at the Met Office, in particular Mike Cullen and Marek Wlasak for their supervision. Mike's comprehensive knowledge of Data Assimilation has been instrumental to my understanding of the field, and I was grateful for his willingness to always answer the many questions I had for him in great detail. Marek's excellent knowledge of the VAR system made life easier for me during my placements at the Met Office, and his patience and aptness were very much appreciated.

Thank you to the staff of the Department of Mathematical Sciences for their help with technical and administrative issues, and for funding several visits to national and international conferences. Thank you also to the Numerical Analysis group for many interesting and useful seminars, and for showing a genuine interest in my work. Many of them helped me and gave me fruitful comments, these include Ivan Graham, Alastair Spence, Jan Van Lent, Melina Freitag and Emily Walsh.

Special thanks to Great Western Research and the Met Office who financed this project and supported the collaboration with the University of Bath. I hope the collaboration will continue for many years and lead to several successful projects in the future.

I would like to pay a particular respect to my family who got me here with continued support and belief, without whom none of this would have been possible.

Finally I would like to thank my friends who have made my PhD experience such an enjoyable one. I thank Phil, my housemate for the past three years, for his unique brand of highly intellectual humour which is still lost on me, and for his impeccable decorum at all times. Thanks to Fynn and Jane for making away days useful and fun, neither of which were perceived to be possible before! Finally, thank you to Maddy for her understanding and encouragement from the start of my PhD, and for giving me a push in the right direction when I needed it. Her proof reading skills have also been invaluable several times.

Contents

1 Introduction **1**

1.1 The Subject of the Thesis 1
1.2 The Aims of the Thesis 7
1.3 The Achievements of the Thesis 7
1.4 The Structure of the Thesis 8

2 Elliptic Problems in Numerical Weather Prediction **10**

2.1 The Helmholtz Problem in NWP 11
2.2 Data Assimilation in NWP 17
2.2.1 3D-VAR 18
2.2.2 4D-VAR 19
2.2.3 Control Variable Transforms in 3D-VAR 20
2.2.4 Choice of Control Variable Transform 22
2.3 PV-Based Control Variable Transformations 26
2.4 Grid Structure for the UM 28
2.5 Summary of the Key Elliptic Equations in NWP 30

3 Model Problem and Discretisation **32**

3.1 Model Problem 32
3.2 Finite Volume Discretisation 34
3.2.1 Discretisation of the General Problem 34
3.2.2 Discretisation of the Terms on the Boundary 38
3.2.3 Special Case – Spherical Polar Coordinates 40
3.2.4 Two Dimensional Problems on the Surface of the Sphere 45
3.3 Finite elements and a Link to Finite Volumes Using Quadrature 47
3.3.1 Piecewise Bilinear Finite Elements in Two Dimensions 47
3.3.2 Link to the Finite Volume Scheme 49
3.3.3 Extension to Three Dimensions 53

4 Numerical Solution of Large Sparse Systems of Linear Equations 58
4.1 Introduction and Model Problems . . . 58
4.2 Basic Iterative Methods . . . 62
4.2.1 Point-Wise Relaxation Schemes . . . 63
4.2.2 Block Relaxation Schemes . . . 65
4.2.3 Error Analysis . . . 66
4.3 Preconditioned Krylov Subspace Methods . . . 69
4.4 Multigrid Methods . . . 70
4.4.1 Heuristics . . . 70
4.4.2 Grid Transfer Operators . . . 72
4.4.3 The Two Grid Method (TGM) . . . 76
4.4.4 The Multigrid Method (MGM) . . . 77
4.4.5 Basic Theory . . . 79
4.5 Anisotropic Problems . . . 80
4.5.1 Semi Coarsening with Point Relaxation . . . 81
4.5.2 Line Relaxation with Full Coarsening . . . 82
4.5.3 Semi Coarsening with Line Relaxation . . . 83
4.6 Convergence Analysis of the Multigrid Method . . . 84
4.6.1 Analysis of the Two-Grid Method . . . 84
4.6.2 Analysis of the Multigrid V-Cycle . . . 91
4.7 Algebraic Multigrid (AMG) . . . 93
4.7.1 Algebraic Smoothness . . . 95
4.7.2 Strong Coupling . . . 95
4.7.3 Selecting the Coarse Grid . . . 96
4.7.4 The Interpolation Operator . . . 97

5 Non-Uniform Multigrid for Spherical Polar Grids 100
5.1 Motivation by Studying Anisotropic Problems on the Unit Square . . . 102
5.1.1 Standard Geometric Multigrid Approaches . . . 103
5.1.2 Algebraic Multigrid (AMG) for Anisotropic Problems . . . 107
5.2 New Approach – Conditional Semi-Coarsening . . . 109
5.2.1 Heuristic Explanation of the Effectiveness . . . 113
5.2.2 Numerical Experiments . . . 114
5.3 Extensions to Three Dimensions . . . 116
5.3.1 Dealing with the Additional Anisotropy in the z-Direction . . . 117
5.3.2 Reducing the Anisotropy in the z-Direction . . . 119
5.3.3 Line Relaxation and No Coarsening in z . . . 120

5.3.4 Comparison with Full Coarsening and Semi Coarsening 121
5.4 Convergence Theory Using a Tensor Product Approach 122
5.4.1 Convergence Theory for NUMG in Two Dimensions 123
5.4.2 Convergence Theory for NUMG in Three Dimensions 132
5.5 Non-Uniform Multigrid in Spherical Geometries 138
5.5.1 Poisson-Type Equations on the Unit Sphere 139
5.5.2 Dealing with the Singularity of the Problem 141
5.5.3 Numerical Results 143
5.5.4 The Galerkin Approach 145
5.5.5 Comparison with Algebraic Multigrid (AMG) 147
5.6 Extension to 3D Elliptic Problems in NWP 148
5.6.1 The Galerkin Approach 150
5.6.2 Reducing the Anisotropy in the Radial Direction 151
5.6.3 Comparison with Algebraic Multigrid (AMG) 153
5.6.4 The Quasi-Geostrophic Omega Equation 153
5.6.5 The Helmholtz Problem 155

6 Parallelization 158
6.1 Introduction to Parallelism 158
6.1.1 Message Passing 159
6.1.2 Performance of Parallel Algorithms 161
6.2 Multigrid in Parallel 162
6.2.1 Grid Partitioning 162
6.2.2 Communication 163
6.2.3 Parallel Components of Multigrid 165
6.2.4 Parallelization Strategy of NUMG 169
6.2.5 Parallel Multigrid Algorithm 172
6.3 Parallel Numerical Results – Speedup and Scaling 173
6.3.1 2D results 174
6.3.2 3D results 177

7 Application of Multigrid in the Potential Vorticity Based Control Variable Transform 181
7.1 PV-Based Control Variable Transform 182
7.2 Strategy for Solving the Balanced PV-Equation 188
7.3 Results for the Simplified Model Problem 191
7.4 Results for the Full Problem 193
7.5 Summary 196

8 Future Work and Extensions **198**

A Convergence Theory Using Fourier Analysis **199**

B The Multigrid Iteration Matrix **203**

C Proof of the Approximation Property **205**

Bibliography **208**

List of Figures

2-1 4D-VAR 20
2-2 Arakawa C-grid used in the horizontal 29
2-3 Charney Phillips grid used in the vertical 29
2-4 The Met Office grid staggering 30

3-1 (a) The computational grid in the λ-ϕ plane, and (b) the graded mesh in the r-direction. 41
3-2 The polar region, with the pole at the centre of n_λ half cells 43
3-3 A spy plot showing the sparsity pattern of matrix A for the 3D problem. 45
3-4 A spy plot showing the sparsity pattern of the matrix for the 2D problem. 46
3-5 Finite element mesh 50
3-6 Element $\tau_{i+\frac{1}{2},j+\frac{1}{2}}$ 51

4-1 The two-dimensional grid on the unit square 60
4-2 The error of a random initial guess after zero iterations (top left), two iterations (top right), four iterations (bottom left) and six iterations (bottom right) of the Gauss–Seidel method. The oscillatory components are removed after very few iterations, but the smooth components are damped very slowly. 68
4-3 Full coarsening on an 8×8 uniform grid 72
4-4 Linear interpolation in two-dimensions for (a) a cell centred grid and (b) a vertex centred grid. Circles denote the coarse grid nodes and crosses denote fine grid nodes 75
4-5 The multigrid (a) V-cycle and (b) W-cycle on four grids 78
4-6 Semi coarsening on an 8×8 uniform grid 82

5-1 Conditional semi-coarsening on a 16×8 grid 111
5-2 Conditional semi-coarsening on a 400×200 grid (after four refinements) 111

5-3 The weighting of values from adjacent nodes in (a) a uniform mesh, and (b) a non-uniform mesh. 113
5-4 Aspect ratio $\frac{h_y}{h_x}$ for (top left) zero, (top right) one, (bottom left) four and (bottom right) seven refinements 115
5-5 A_F has a 5-point stencil and A_ℓ, for $\ell = 1, \ldots, F-1$ have a 21-point stencil, the extra fill-in created as a result of the Galerkin Product . . . 146
5-6 Snapshot of the vertical velocity, w, after one iteration of VAR at θ-level 20 (roughly 8.8km above ground level) 156

6-1 The `MPI_BCAST` operation 160
6-2 The `MPI_ALLREDUCE` operation, with an arithmetic/ logical operation $\odot$. 161
6-3 Partitioning Ω into four subdomains 162
6-4 The communication needed for matrix-vector multiplications 165
6-5 (a) Linear interpolation: some entries of the matrix depend on data from processors associated with adjacent subdomains only, and (b) bi-linear interpolation: some entries of the matrix depend on data also from processors associated with subdomains that share a corner with Ω_i. Dotted lines denote the fine grid, the thin lines denote the coarse grid and the thick lines denote the subdomain boundaries 166
6-6 (a) Full weighting restriction: some entries of the matrix depend on data from processors associated with adjacent subdomains, and (b) Four point average: all the entries of the matrix depend on the data from the local processor. Dotted lines denote the fine grid, the thin lines denote the coarse grid and the thick lines denote the subdomain boundaries 167
6-7 Parallel non-uniform coarsening keeping the location of the processor interface fixed. Grids for one refinement (top) and three refinements (bottom) with a uniform 4×4 partitioning onto 16 processors. 171
6-8 Communication for processor subdomains on the periodic boundary . . 172
6-9 Speedup (strong scalability) test of the 2D NUMG method on `wolf` for two different problem sizes. 175
6-10 Scaled efficiency (weak scalability) test of 2D NUMG on `wolf`: Problem size 1920×960 on each processor. 176
6-11 Scaled efficiency (weak scalability) test of 2D NUMG on `aquila`: Problem size 512×256 on each processor. 176
6-12 Speedup (strong scalability) test on `wolf` for 3D NUMG (solid blue line) and the r-line preconditioned CG method (dotted black line). Global problem size: $360 \times 180 \times 100$ 177

6-13 Scaled efficiency (weak scalability) test on `wolf` for 3D NUMG: Problem size $200 \times 100 \times 50$ on each processor. 178
6-14 Scaled efficiency (weak scalability) test on `aquila` for 3D NUMG: Problem size $192 \times 120 \times 50$ on each processor. 178
6-15 Scaled efficiency (weak scalability) test on `wolf` for the r-line preconditioned CG method: Problem size $200 \times 100 \times 50$ on each processor. . . . 179
6-16 Snapshot of the vertical velocity, w, after one iteration of VAR at θ-level 20 (roughly 8.8km above ground level), using eight processors 180

7-1 (a) The finite volume grid with (a) p'-points as cell centres and (b) ψ'-points as cell centres 184
7-2 Snapshot of the balanced streamfunction, ψ_b, at ρ-level 40 (roughly 29.8km above ground level) 197

List of Tables

5.1 Number of iterations, N_{its}, required to solve (5.1.2). 106
5.2 Total CPU time taken (including setup time) to solve (5.1.2). 107
5.3 Average convergence factor, μ_{avg}, when solving (5.1.2). 107
5.4 The `BoomerAMG` method applied to model problem (5.1.2) as a stand alone solver and a preconditioner for the CG method. CPU times in seconds. 108
5.5 Model problem (5.1.2) solved using NUMG as a stand alone solver. CPU time in seconds. 116
5.6 Model problem (5.1.2) solved using NUMG as a preconditioner to the CG method. CPU time in seconds. 116
5.7 Three dimensional model problem (5.3.1) with $L_z = 10^4$ solved using NUMG. CPU time in seconds. 118
5.8 Three dimensional model problem (5.3.1) with $L_z = 10^4$ solved using z-line relaxation. CPU time in seconds. 119
5.9 NUMG used to solve (5.3.1) for varying strengths of the vertical anisotropy. CPU time in seconds. 119
5.10 NUMG with conditional semi-coarsening on the $x-y$ plane, z-line relaxation and no vertical coarsening applied to (5.3.1). CPU time in seconds. The method is optimal for all values of $L_z > 0$. 121
5.11 Solving (5.3.1) using NUMG with z-line relaxation, no vertical coarsening and *full* coarsening on the x–y plane. CPU time in seconds. 121
5.12 Solving (5.3.1) using NUMG with z-line relaxation, no vertical coarsening and coarsening only in the x-direction. CPU time in seconds. 122
5.13 Two dimensional Poisson's equation on the unit sphere solved using NUMG (with a projection onto the range of A_F in each iteration). CPU time in seconds. 143
5.14 Two dimensional Poisson's equation on the unit sphere with varying anisotropy solved using NUMG (CPU time in seconds). 145

5.15 Two dimensional Poisson's equation on the unit sphere solved using NUMG, the Galerkin product and full-weighting restriction 146
5.16 Two dimensional Poisson's equation on the unit sphere solved using NUMG, the Galerkin product and four-point averaging restriction . . . 147
5.17 `BoomerAMG` used to solve model problem (5.5.1). CPU times in seconds. 147
5.18 Three dimensional Poisson's equation on the unit sphere solved using NUMG. CPU time in seconds. 150
5.19 Three dimensional Poisson's equation on the unit sphere solved using (column 1) NUMG with coarse grid operators created using the Galerkin product, (column 2) r-line Gauss-Seidel and (column 3) CG preconditioned with the r-line Gauss–Seidel smoother. CPU time in seconds. . . 150
5.20 Preconditioned CG method (with r-line Jacobi preconditioner) used to solve (5.6.1) with $L_\lambda = L_\phi = 1$ and $L_r = 10^{-4}$. 152
5.21 NUMG with conditional semi-coarsening on the $\phi - \lambda$ plane, r-line relaxation and no vertical coarsening applied to (5.6.1), with $L_\lambda = L_\phi = 1$ and various values of L_r. CPU time in seconds. 152
5.22 Three Dimensional Poisson-type Equation (5.6.1) with $L_\lambda = L_\phi = 1$ and $L_r = 10^{-4}$ solved using `BoomerAMG` as a preconditioner to CG. 153
5.23 Three dimensional Helmholtz problem (2.1.24), as used in NWP, solved on the unit sphere using NUMG. CPU time in seconds. 157

7.1 The magnitude of errors in the PV-based CVT. Column 5 denotes the number of PCG iterations required for solving (7.3.2) and (7.3.3), respectively. Similarly column 6 denotes the CPU time (in seconds) required for solving (7.3.2) and (7.3.3), respectively. 192
7.2 Details of each component when solving the 3D balanced equation (7.3.2). CPU time in seconds. Identical results are found when solving the 3D unbalanced equation (7.3.3). 193
7.3 The magnitude of errors in the PV-based CVT. Column 5 denotes the number of PCG iterations required for solving (7.1.4) and (7.1.6), respectively. Similarly column 6 denotes the CPU time (in seconds) required for solving (7.1.4) and (7.1.6), respectively. 196
7.4 Details of each component when solving the 3D balanced equation (7.1.4). The same results are found when solving the 3D unbalanced equation (7.1.6). CPU time in seconds. 196

List of Algorithms

4.1 The Gauss–Seidel Method: `gs(`$A, \mathtt{u}, \mathtt{b}$`)` 65
4.2 The Block Gauss–Seidel Method: `blockgs(`$A, \mathtt{u}, \mathtt{b}$`)` 66
4.3 Preconditioned conjugate gradient method: `pcg(`$A, \mathtt{u}, \mathtt{b}$`)` 70
4.4 The Two Grid Method (TGM) 76
4.5 The Multigrid method (MGM) 78
4.6 The Multigrid V-cycle: `Vcycle(`$A_\ell, \mathtt{u}_\ell, \mathtt{b}_\ell, \ell$`)` 79
5.1 Conditional semi-coarsening for model problem (5.1.2): `Conditional`$(h_x^{(\ell)}, h_y^{(\ell)}, n_x^{(\ell)}, n_y^{(\ell)}, \ell, h_x^{(\ell-1)}, h_x^{(\ell-1)}, n_x^{(\ell-1)}, n_y^{(\ell-1)})$ 112
5.2 Conditional semi-coarsening for the Poisson-type equation on the sphere: `Cond_spherical`$(h_\lambda^{(\ell)}, h_\phi^{(\ell)}, n_\lambda^{(\ell)}, n_\phi^{(\ell)}, \ell, h_\lambda^{(\ell-1)}, h_\lambda^{(\ell-1)}, n_\lambda^{(\ell-1)}, n_\phi^{(\ell-1)})$ 140
6.1 The hybrid Jacobi/Gauss–Seidel Method: `jac_gs(`$A_i, \mathtt{u}_i, \mathtt{b}_i$`)` 169
6.2 The parallel V-cycle on processor i: `PVcycle`$((A_\ell)_i, (\mathbf{u}_\ell)_i, (\mathbf{b}_\ell)_i, \ell)$ 173
6.3 Parallel conditional semi-coarsening for the Poisson-type equation on the sphere: `Cond_parallel`$(h_\theta^{(\ell)}, h_\phi^{(\ell)}, n_\theta^{(\ell)}, n_\phi^{(\ell)}, \ell, h_\theta^{(\ell-1)}, h_\theta^{(\ell-1)}, n_\theta^{(\ell-1)}, n_\phi^{(\ell-1)})$ 174
7.1 `apply_operator(`$\mathcal{A}, \mathtt{p}, \mathtt{q}$`)` 190
7.2 Preconditioned Conjugate Gradient method: `pcg(`$A, \mathtt{u}, \mathtt{b}$`)` 191
7.3 Bi-CGSTAB method with left preconditioning: `pcg(`$A, \mathtt{u}, \mathtt{b}$`)` 195

Chapter 1

Introduction

1.1 The Subject of the Thesis

Two and three dimensional elliptic partial differential equations (PDEs) play a major role in the weather and climate numerical model at the UK Met Office, and in its variational data assimilation code (VAR).

The fully compressible and non-hydrostatic Euler equations form the basis of the dynamical core of the Met Office's weather and climate prediction unified model (UM). These equations form a hyperbolic system, but through the application of a semi-implicit time discretisation, an elliptic Helmholtz problem is naturally derived for the increment of the pressure field, denoted Π', between two adjacent time steps. The equation, given in spherical polar coordinates, has the form

$$-\nabla_r^2\Pi' - \frac{1}{r^2}\frac{\partial}{\partial r}\left(a(r)\frac{\partial\Pi'}{\partial r}\right) + b(r)\frac{\partial\Pi'}{\partial r} + c(r)\Pi' = \Phi\,, \tag{1.1.1}$$

where $a(r),\ b(r),\ c(r) > 0\ \forall r$, ∇_r^2 is the two-dimensional (2D) Laplacian in spherical polar coordinates at a constant height r, and the right-hand-side contains variables computed at the previous time step. The numerical solution of (1.1.1) plays a vital role in the UM, as it must be solved at each time step. The full equation, including its derivation, is found in [28, 31]. The UM is in fact currently undergoing a significant overhaul, and its new models are described in [57, 73, 89], but once again the key problem within these models is the solution of the Helmholtz equation which retains the form of (1.1.1).

In variational data assimilation, the main task is to produce the best estimate for the current state of the atmosphere, which is then used as an initial condition for a future forecast. Every day the Met Office receives around half a million observations

recording the atmospheric conditions around the world. However, even with this many observations there is not enough information on the behaviour of the atmosphere at all points on and above the Earth's surface. There are large areas of ocean, inaccessible regions on land and remote levels in the atmosphere where very few, or no, observations are recorded. To fill in the 'gaps', the available observations must be combined with forecasts of the expected conditions in the atmosphere, known as the background state. The Met Office uses a set of model variables, such as wind velocity, pressure and temperature, to describe the state of the atmosphere. Thus, given the background state and a set of observations in a certain time window, variational data assimilation is the task of adjusting the model variables in view of gaining the best estimate of the current state of the atmosphere.

This is done via the minimization of a cost function, which unfortunately involves the inversion of a large and dense matrix related to error covariances of the model variables, which is operationally unfeasible. The matrix, known as the background error covariance matrix, is dense because the errors of the model variables in the background state are highly correlated. A key task in simplifying the minimization process is the "control variable transform (CVT)", which is used to transform between the model variables and a set of variables known as the *control variables*. The purpose of the CVT is to perform a transformation to a set of variables that are assumed less correlated so that the background error covariance matrix of these variables has a more sparse representation. The atmospheric state can be partitioned into components that are known to be *balanced* or *unbalanced*, which evolve independently and are therefore assumed to be uncorrelated. Hence the choice of control variables is made according to whether they are balanced or unbalanced.

The main bottleneck in the CVT currently operational at the Met Office is in the numerical solution of the *Quasi-Geostrophic Omega* equation, which has the form

$$-N^2(r)\,\nabla_r^2 w' \;-\; f_0^2\,\frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial w'}{\partial r}\right) \;=\; g\,. \tag{1.1.2}$$

The equation is solved to find the vertical velocity increment (i.e. the deviation from the vertical velocity, w, given in the background state) in scales important for weather forecasting in the atmosphere. f_0 is the Coriolis parameter, which in the quasi-geostrophic regime is assumed constant. Note however that changing this to a (more realistic) variable parameter $f(\phi)$ poses no additional difficulties for the methods we propose in this thesis. $N^2(r)$ is related to the frequency of vertical buoyancy oscillations, which depends on the temperature gradient and varies smoothly with r. The right hand side term g encompasses all the sources of quasi-geostrophic forcing for vertical motion, such

as temperature gradients, quasi-geostrophic wind, quasi-geostrophic vorticity, and latent heat release. Details of the equation, including its derivation and the asymptotic regimes in which it is valid, can be found in [12, 26]. Operationally at the Met Office, the solution w' is currently simply set to zero because solving the equation takes too long using the existing solvers at the Met Office.

The current CVT is limited to certain regimes of the atmosphere. However, several studies in [5, 26, 25, 53, 81] have demonstrated that it is possible to eliminate the shortcomings of the current CVT by choosing a set of new potential vorticity (PV) based control variables. PV is a completely balanced variable so a control variable related to PV will be suited to describe the balanced part of the atmospheric flow better. Likewise, anti-PV is completely unbalanced, and so control variables related to anti-PV will accurately describe the unbalanced components of flow. Using these control variables, a new PV-based CVT has been proposed to satisfy the assumption of the non-correlation between the variables at *all* regimes of the atmosphere, thus overcoming the limitations of the existing CVT. Unfortunately, the PV-based CVT poses further difficulties, where more highly ill-conditioned three dimensional (3D) elliptic problems must be solved in addition to (1.1.2). The equations, written abstractly, have the form

$$-\alpha_0 \nabla_r^2 u \; - \varepsilon_0 \frac{\partial^2}{\partial r^2} \left(\nabla_r^{-2} \nabla_r \cdot f \nabla_r u \right) \; = \; f \, , \tag{1.1.3}$$

meaning a two dimensional (2D) solve is embedded within the 3D problems, and so they cannot be discretised directly. As with (1.1.2), solving (1.1.3) is so far unfeasible using the solvers at the Met Office, and so the PV-based CVT cannot be made operational until a better solver is developed for these problems.

For the spatial discretisation of the above problems, many of the standard meteorological codes, in particular at the Met Office, use spherical polar grids, which lead to strong grid anisotropies near the poles where the grid lines converge. Alternative grids that avoid the "pole problem" such as Yin−Yang or icosahedral grids, are becoming increasingly popular in numerical weather prediction, as discussed in [9, 79]. Nevertheless, for all the negative things the spherical polar grids might entail, these grids are very structured. This greatly simplifies the discretisation and coding for problems such as (1.1.1) and (1.1.2), which is why they are still widely used. We will show in this thesis that from a solver point of view the bad reputation of spherical polar grids (e.g. in [79, §3.2b]) is unjustified, provided the solvers are suitably adapted. Before we expand a bit further on this let us note that the grid spacings in the radial direction are in general much smaller than in the horizontal ones, since the thickness of the atmosphere is two orders of magnitude smaller than the circumference of the Earth.

This creates a further source of anisotropy. Additionally, the grid is usually strongly graded in the radial direction with smaller grid spacings near the surface of the Earth to obtain a better resolution in the regions of most interest.

A standard finite volume discretisation of (1.1.1) or (1.1.2) on this anisotropic mesh leads to a system of equations

$$A\mathbf{u} = \mathbf{b}, \tag{1.1.4}$$

where A is a large, sparse, symmetric positive definite (SPD) matrix. The discretisation which we use is basically identical to that given in [7] for Poisson's equation on a spherical polar grid. The matrix A contains a 7-point stencil for each node on the grid, with non-zero entries only for the node itself and for its immediate neighbours. Typical grid resolutions currently used in data assimilation at the Met Office are 216, 163 and 70 nodes in the latitudinal, longitudinal and radial directions, respectively. This leads to a large problem size of over a million degrees of freedom and an ill-conditioned system matrix A, making (1.1.4) very difficult to solve efficiently. The solver currently used at the Met Office, i.e. a Krylov subspace method preconditioned with simple r-line relaxation or ADI-type methods, performes increasingly poorly as the problem size is increased, and so restricts the grid resolutions that are currently feasible for global simulations, as highlighted in [79, §3.2b]. [1]

It is the subject of this thesis to analyze and solve elliptic problems arising in numerical weather prediction using an optimal and scalable parallel iterative solver. We focus on solving (1.1.1), (1.1.2) and (1.1.3), all of which are given in spherical polar coordinates. By *optimal* we mean that the time for solving the discretised problem is proportional to the (discrete) problem size. Similarly, we say that an algorithm has *optimal parallel scalability*, if the solution time remains constant when the problem size and the number of processors are increased proportionally. The aforementioned solver used at the Met Office is not optimal because the increase in solve times with respect to the problem size is not linear. Note that fast *direct* solvers for elliptic problems in spherical geometries based on fast Fourier transforms (FFT) have also been investigated in [54], but these solvers are not quite optimal either.

It is well known that it is necessary to resort to multilevel techniques to obtain optimality of iterative methods for large elliptic problems. Many variants of these iterative methods exist, but as outlined in [20, 75] they all rely on reducing high frequency errors of an initial approximation using a relaxation method (the smoother), and approximating the remaining low frequency errors on a succession of coarser grids (the coarse grid correction). For isotropic problems with smoothly varying coefficients,

[1] Note that the mean radius of the earth is about 6370km, and so the horizontal grid size in the latitudinal direction for 216 nodes is about 185km near the equator.

standard geometric multigrid with simple point-wise smoothing and uniform coarsening is the most efficient method, as demonstrated experimentally in several articles such as [27, 41]. For anisotropic problems, this standard approach unfortunately does not lead to an optimal method. However, if the anisotropies are aligned with the grid then simple modifications achieve optimality even with strong anisotropies. These modifications are line/plane smoothing and/or semi-coarsening. Line smoothing involves collectively relaxing all unknowns on an entire grid line by solving a tridiagonal system corresponding to the unknowns on that line. Semi-coarsening uses a family of coarse grids that are only coarsened in the direction of the larger coefficient. In (1.1.4) there are two sources of anisotropy: one due to the large aspect ratio between the radial and horizontal grid spacings; the second due to the converging grid lines at the poles. In this thesis we propose a robust geometric multigrid method that is able to deal with both these problems by applying a simple non-uniform partial coarsening combined with an r-line smoother.

The robustness of the non-uniform coarsening strategy is first demonstrated on a two-dimensional model problem: Poisson's equation on the unit sphere. The idea is simple. The spherical polar grid introduces anisotropy near the poles but not near the equator, so the grid is semi-coarsened near the poles but fully coarsened near the equator. We compare the off-diagonal matrix entries in the latitudinal and longitudinal direction at each line of latitude, and the grid line is fully coarsened only if the coefficients in both directions are of similar magnitude. This will be true near the equator where we coarsen in both directions, but not near the poles where we only coarsen in the longitudinal direction, thus leading to coarse grids that are better and better adapted to the anisotropy. This coarsening strategy is the key heuristic for the popular algebraic multigrid (AMG) methods (see e.g. [20, 71]), and the robustness of these methods even on highly anisotropic problems demonstrate the success of the strategy.

In 3D we deal with the strong anisotropy in the radial direction by using r-line relaxation and no r-coarsening. This is then combined with the nonuniform coarsening strategy in the longitudinal and latitudinal directions. Although this partial coarsening only leads to a coarsening factor of about 3 from one grid to the next (instead of 8 for uniform coarsening), it guarantees that the method is fully robust to the anisotropies induced by the geometry and the grid and leads to an optimal method with an average V-cycle convergence factor of less than 0.1, as our numerical tests show.

Geometric multigrid methods with line and plane smoothers, but with "uniform" semi-coarsening, have already been studied in [7]. PDEs of the type (1.1.2) from meteorological applications have already been solved with geometric multigrid methods, but only on cube-like domains with doubly-periodic boundary conditions and not on

the entire globe (cf. [3, 39, 62, 85, 83, 84]). The most closely related paper is [84], where r-line relaxation and partial coarsening (i.e. uniform coarsening in the horizontal directions and no coarsening in the radial direction) was already studied extensively for the quasi-geostrophic equations in Cartesian coordinates. However, since the domain was not the entire atmosphere the additional complication of the anisotropy at the poles played no role. Multigrid algorithms have also already been proposed for alternative grids on the sphere, such as the icosahedral or Yin−Yang grids, in [9, 51]

The idea of "conditional" semi-coarsening in the longitudinal direction proposed here has only been explored for edge and corner singularities so far (cf. [40, 55, 90]) but not for spherical polar grids (even in two dimensions). To the best of our knowledge, it seems to be a novel approach. It is clearly inspired by AMG ideas. AMG methods are fully automatic and only based on algebraic information in the matrix A. Coarse grid unknowns are chosen based on the relative size of the off-diagonal entries in the matrix which in the application here will lead to very similar coarse grids. However, AMG methods are known to require a large setup cost to design these coarse grids and the operator-dependent interpolation and restriction operators, especially in three dimensions. Our geometric method on the other hand, requires almost no setup cost to obtain the same robustness, which is why it easily outperforms established AMG implementations. Numerical tests (cf. Chapter 5) for a variety of problem sizes confirm this. In that chapter, we also give a comparison to preconditioned Krylov solvers as currently used by the MET Office [28] and their collaborators [22]. As expected, Krylov methods are only optimal when preconditioned with a robust multigrid method, such as AMG or the non-uniform geometric method proposed in this thesis. With standard preconditioners used at the MET Office, such as r-line relaxation or ADI-type preconditioners (on a single grid), the number of iterations grows with the problem size. We can make theoretical justifications for the optimality of the non-uniform geometric multigrid method based on the heuristics that discretisations on coarser grids produced by conditional semi-coarsening yield more isotropic problems. Note that no such theory exists for AMG.

All sequential computations will be carried out using the Fortran95 compiler `ifort` on a single processor of a Dual dual-core 64bit AMD Opteron 2210 processor with clock speed of 1.8GHz, cache size 1.0MB and 2GB memory. The initial guess for each iterative scheme is always taken to be zero. The stopping criterion is a relative residual reduction of 10^{-8}. Parallel computations are carried out using two different clusters, a 64-bit AMD Opteron 2210 cluster (`wolf`) with a total of 24 processors (each the same as above) and a 64-bit Intel Xeon E5462 cluster (`aquila`) of over 800 processors, each with 2GB memory and 3MB Cache. Both clusters use an Infinipath network.

1.2 The Aims of the Thesis

The main aim of this thesis is to provide a fast, efficient and robust iterative solver for the numerical solution of elliptic problems in spherical geometries arising in numerical weather prediction at the Met Office via a novel non-uniform multigrid method. Since the Met Office runs its codes using massively parallel computers, we also aim for the solver to have optimal parallel scalability. We aim to show the optimality of the method both experimentally and theoretically.

A second major aim of the thesis is to theoretically prove the robustness of the non-uniform multigrid method with respect to problem size, when applied to model problems of type (1.1.1) and (1.1.2). The classical result for the uniform convergence of the multigrid V-cycle [44, 16] is only applicable for isotropic problems. For some anisotropic problems, theoretical results for multigrid using a variety of techniques can be found in, for example, [10, 17, 58, 87]. In particular, for planar polar coordinates, multigrid theory with line smoothers and (uniform) semi-coarsening can be found in [14] (see also [13]). In this thesis a tensor product analysis is used in order to achieve a separation of coordinate directions for an anisotropic 2D problem. The analysis relies on the fact that the anisotropy is grid aligned, and reduces the 2D problem to a family of 1D problems to which the standard theory of Hackbusch can be applied. We aim to prove the robustness of the non-uniform multigrid method applied to anisotropic 3D elliptic problems of the form (1.1.1) and (1.1.2) using a similar approach.

A final aim of the thesis is to provide an efficient solver for the 3D problems in the new PV-based CVT. Since (1.1.3) cannot be discretised, a discrete operator does not exist. Thus we must apply the operator by means of 2D Poisson solves, the solutions to which are applied to the remaining components of the operator. Then we can use a preconditioned Krylov subspace method to solve (1.1.3). The preconditioning step involves using multigrid to solve a simplified form of the (1.1.3) that resembles the Quasi-Geostrophic omega equation, which can be done optimally.

1.3 The Achievements of the Thesis

The following are the main achievements of the thesis:

1. A robust multigrid method for the 2D Poisson equation on the unit sphere was successfully developed using the non-uniform coarsening strategy. The uniform convergence of the method was shown experimentally, and some heuristic arguments were also given to back-up the experimental results.

2. The non-uniform multigrid method was developed further and extended to 3D to

optimally solve the Helmholtz problem (1.1.1) and the Quasi-Geostrophic omega equation (1.1.2). Note that the direction of strongest coupling is not always fixed, so an r-line smoother is used to handle the strong coupling in the radial direction, whilst conditional semi-coarsening deals with the anisotropy on the 2D plane. The same multigrid method solves both problems optimally.

3. We use heuristics for the convergence of the V-cycle for the 2D Poisson problem and theory from [14] to show that 3D elliptic problems with grid-aligned anisotropy converge uniformly when solved using non-uniform multigrid, with a contraction factor that is independent of mesh refinement and the varying coefficients. The proof is based on a tensor product analysis that leads to a separation of coordinate directions to reduce the analysis of the 3D problem to that of a family of simpler 2D problems which can be solved optimally with the non-uniform multigrid method.

4. A highly scalable parallel implementation of the non-uniform multigrid method was developed. The method was shown to be almost optimally scalable up to 256 processors, and was observed to perform significantly faster than several existing methods that were tested.

5. The non-uniform multigrid method was also used successfully as a preconditioner to Krylov subspace methods, accelerating these methods to an extent that they perform optimally. Using multigrid as a preconditioner enabled a unique method for solving the elliptic problems in the PV-based CVT, which was not possible when using the solvers currently employed at the Met Office.

1.4 The Structure of the Thesis

The majority of the subsequent chapters begin with a preamble to motivate the work of the chapter in addition to a literature review related to the work. The contents of each chapter in the remainder of this thesis is as follows:

In Chapter 2 we present the main elliptic problems that arise in numerical weather prediction, namely from the Met Office's Unified Model and its variational data assimilation scheme. The derivation of each problem is described, as well as the structure of the computational grids that are used at the Met Office.

In Chapter 3, we describe the finite volume discretisation of the 2D and 3D elliptic problems, using the grid structure of the Met Office described in Chapter 2. We present the discretisation in such a way that it applies to all the elliptic problems of interest in this thesis. We also present an alternative discretisation method using finite elements,

which will be necessary for the theoretical analysis. However, we show that the two discretisation schemes can be linked using a quadrature rule that is at least second order accurate.

Chapter 4 gives an overview of existing iterative solvers for elliptic problems, starting with relaxation schemes and Krylov subspace methods. We then describe the idea of multigrid methods for isotropic elliptic problems solved on a uniform grid, and how the method needs to be adapted for anisotropic problems. This chapter also includes an overview of the convergence proofs of multigrid methods, following the work of Hackbusch [44], and it is concluded with a description of AMG methods.

Chapter 5 is the main chapter of the thesis. We describe the novel idea of a non-uniform geometric multigrid method that is adapted to the particular anisotropies induced by the geometry and by the grid, and highlight some similarities with the AMG methods described in Chapter 4. Numerical results are given for the sequential solver applied to 2D and 3D model problems on the unit square/cube with degenerate coefficients, as well as comparisons with AMG and preconditioned Krylov methods. The heart of this chapter is a new convergence theory that follows the techniques used by Börm and Hiptmair [14] and confirms the robustness of the method. However, the theory relies on a finite element discretisation of the elliptic problem, so the quadrature rule from Chapter 3 is used to show that the theory also carries over to a finite volume setting. The section is concluded with the application of the method to elliptic problems on the sphere, in particular the Helmholtz problem and the Quasi-Geostrophic Omega equation.

In Chapter 6, we outline how we parallelized our method and demonstrate its parallel scalability for up to 256 processors, as well as comparing the scalability to that of parallel versions of the other solvers.

Finally in Chapter 7, we outline another novel method for solving the 3D problems present in the PV-based CVT. We describe how the operator is applied without the use of a matrix, and how Krylov methods can be accelerated using a multigrid preconditioner that solves a simplified problem. Numerical results are given for the performance of the method and for the accuracy of the complete cycle of the PV-based CVT when using this novel approach.

Chapter 2

Elliptic Problems in Numerical Weather Prediction

In this chapter we describe the main elliptic problems that arise in numerical weather prediction (NWP). The process of NWP involves the assimilation of estimates to the initial conditions for a numerical weather forecast model, and once these conditions are calculated, the changes in weather are predicted by advancing the model in time. The flagship numerical model developed and used at the Met Office is called the *Unified Model* (UM) and the assimilation of the initial conditions is accomplished by *variational data assimilation* (VAR).

For two decades, the UM has been used at the Met Office for both low resolution climate modelling and high resolution operational NWP. It is versatile and capable of modelling a wide range of time and space scales and is run in many different configurations at the Met Office. It has been in continual development since the early 1990s, taking advantage of increasing supercomputer power, improved understanding of atmospheric physics and an increasing range of observational data sources. The most current version of the UM uses governing equations with a non-hydrostatic, fully compressible and deep atmosphere formulation (see [24, 28, 31, 78]), which describe the rates of change of the wind components, the potential temperature, density, humidity and pressure variables. The equations are formulated in spherical polar coordinates and discretised using a staggered grid in each coordinate direction. The discretisation in time is done using a two-time level predictor-corrector implementation of a semi-implicit scheme, as described in [28, 31]. The discretisation in this form leads to increased stability, thus allowing larger time steps to be used, but is complicated by the need to solve a three-dimensional Helmholtz equation at each time step which takes up a significant fraction of the cost of the UM. This Helmholtz problem is the first

of the elliptic problems we are interested in. To solve it, a preconditioned generalized conjugate residual (GCR) method, as described in [34], is currently used operationally in the UM.

The UM also relies on accurate initial conditions of the current state of the atmosphere, which is provided via data assimilation. By combining observational data, statistical data, knowledge of atmospheric dynamics and previous forecasts, the best estimate to the initial conditions is found. Due to the sheer problem size involved there are inherent problems in defining the required matrices, and a process known as the *control variable transform* (CVT) is used to simplify the problem, as described in [30, 52, 53, 80]. A key equation in the CVT is the *Quasi-Geostrophic omega* (QG-Ω) equation [38], which is the second of the elliptic problems of interest.

The CVT is limited to certain regimes of atmospheric flow. Some of these limitations are thought to be overcome by a newly proposed CVT based on potential vorticity (PV). Many theoretical studies of this new scheme exist (see [52, 53, 6, 5, 25, 78, 81]), but it is not yet operational as the current solvers are not capable of solving the problems involved. Some of the problems that appear in the PV-transform are so ill-conditioned that the current solvers at the Met Office do not converge to a sufficiently accurate solution even after several hundred iterations. Thus the third of the elliptic problems of interest appears within the PV-based CVT.

Each of the elliptic problems described above have different characteristics, but all of them are highly important in improving the efficiency and accuracy of NWP at the Met Office.

The chapter is organized as follows. Section 2.1 describes the governing equations used in the UM and how the Helmholtz problem is derived from the semi implicit discretisation of the equations. In Section 2.2 we focus our attention on data assimilation. We describe the CVT that is currently operational at the Met Office and that is central to simplifying the VAR process, and the QG-Ω equation that needs to be solved as a result of the transformations. In Section 2.3 we discuss the new PV-based CVT as a possible improvement to the currently operational CVT and the equations involved. Finally in Section 2.4 the staggered grids used at the Met Office are defined, before summarizing the key elliptic problems in NWP in Section 2.5.

2.1 The Helmholtz Problem in NWP

The fully compressible, non-hydrostatic and deep atmosphere equations form the basis of the Met Office weather and climate prediction unified model (UM). A fully compressible system is one where the density of fluid in the system changes with respect to

pressure, which is the case with the Earth's atmosphere. Many climate models assume hydrostatic balance, where the two main vertical forces, gravity and the pressure gradient, balance each other out. Hydrostatic balance explains why the Earth's atmosphere does not collapse to a thin layer on the ground, and is an accurate approximation on large horizontal scales. However, the UM does not assume complete hydrostatic balance which allows it to take into account vertical wind acceleration. For accuracy, a deep atmosphere formulation is used instead of the shallow atmosphere approximations (see [78]). The prognostic model variables used in atmospheric modelling are the three-dimensional wind $\mathbf{u}$ with components (u, v, w), potential temperature θ, pressure p, density ρ and the specific humidity q. In the UM the Exner function is used as the pressure variable, defined as

$$\Pi = \left(\frac{p}{p_{ref}}\right)^{\kappa}, \quad \kappa = \frac{R}{C_p}. \tag{2.1.1}$$

$R = 287.05$ is the gas constant, $C_p = 1005$ the specific heat at constant temperature and $p_{ref} = 10^5$ Pa a constant reference value of the pressure. Potential temperature is defined in terms of temperature, T, as $\theta = T/\Pi$. The governing equations are written generically as

$$\frac{D\mathbf{X}}{Dt} = \mathbf{L}(\mathbf{x}, t, \mathbf{X}) + \mathbf{N}(\mathbf{x}, t, \mathbf{X}), \tag{2.1.2}$$

where

$$\frac{D}{Dt} = \frac{\partial}{\partial t} + \mathbf{u} \cdot \nabla_z$$

is the Lagrangian derivative [8] which describes the time rate of change of the prognostic variables while moving with a velocity field. $\mathbf{X}$ is a vector of the prognostic variables and $\mathbf{x}$ and t denote position and time. $\mathbf{L}$ and $\mathbf{N}$ represent terms that are linear and nonlinear in $\mathbf{X}$, respectively. ∇_z is the horizontal gradient operator.

The equations are discretised in the UM using a predictor-corrector implementation of a two-time level (2TL), semi-implicit (SI) time discretisation scheme, as outlined in [28], though further developments are being made to the current methods in a scheme called ENDGame [57, 73, 89]. A SI discretisation of (2.1.2) is given by

$$\frac{\mathbf{X}^{n+1} - \mathbf{X}_d^n}{\Delta t} = (1-\alpha)(\mathbf{L}+\mathbf{N})_d^n + \alpha(\mathbf{L}+\mathbf{N})^{n+1},$$

which advances the equation from time level n to time level $n+1$, where $\alpha \in [0, 1]$. The subscript d denotes the evaluation at a departure point $\mathbf{x}_d$, which is the point in space that $\mathbf{X}$ was measured at time level n.

The predictor-corrector approach is then described as follows

$$\begin{aligned}\mathbf{X}^{(1)} &= \mathbf{X}_d^n + (1-\alpha)\Delta t(\mathbf{L}+\mathbf{N})_d^n + \alpha\Delta t(\mathbf{L}+\mathbf{N})^n\,,\\ \mathbf{X}^{(2)} &= \mathbf{X}^{(1)} + \alpha\Delta t\mathbf{L}^{(2)} + \alpha\Delta t(\mathbf{N}^{(1)} - \mathbf{N}^n - \mathbf{L}^n)\,,\end{aligned}$$

where $\mathbf{X}^{(1)}$ is the predicted value at time level $n+1$, and $\mathbf{X}^{(2)}$ is the corrected estimate, such that $\mathbf{X}^{n+1} = \mathbf{X}^{(2)}$. Also, let primed quantities denote the increment from time level n to $n+1$, i.e.

$$\mathbf{X}' = \mathbf{X}^{n+1} - \mathbf{X}^n\,,$$

for each of the prognostic variables.

Equations (2.1.3) – (2.1.7), below, are the governing equations used in the UM, as taken from [28], but with some simplifications made for purposes of clarity. The simplifications are that we write the equations in Cartesian coordinates and assume a dry atmosphere with no orography and a uniform rotation rate of $f = -2\Omega\cos\ y$, where Ω is the Earth's angular velocity.

Momentum Equations

$$\frac{Du}{Dt} = fv - C_p\theta\frac{\partial\Pi}{\partial x} + F_u\,, \tag{2.1.3}$$

$$\frac{Dv}{Dt} = -fu - C_p\theta\frac{\partial\Pi}{\partial y} + F_v\,, \tag{2.1.4}$$

$$\frac{Dw}{Dt} = -g - C_p\theta\frac{\partial\Pi}{\partial z} + F_w\,. \tag{2.1.5}$$

Potential Temperature Equation

$$\frac{D\theta}{Dt} = S\,.$$

Continuity Equation (Conservation of Mass)

$$\frac{\partial\rho}{\partial t} + \nabla_z\cdot(\rho\mathbf{u}) = 0\,. \tag{2.1.6}$$

Equation of State

$$\kappa\Pi\theta\rho = \frac{p}{C_p}\,. \tag{2.1.7}$$

g is the Earth's gravitational force, and F_u, F_v, F_w and S are the tendencies from the physics parameterizations. Let us now discuss how these equations are discretised and which further approximations are used. The equations for θ and $\mathbf{u}$ and are discretised

using the predictor-corrector approach, and after the correction step, they are:

$$\theta^{(2)} = \theta^{(1)} - \alpha\Delta t \frac{\partial \theta^{(1)}}{\partial z} w' \,, \tag{2.1.8}$$

$$u^{(2)} = u^{(1)} + \alpha\Delta t \left[f(v^{(2)} - v^n) - C_p \left\{ (\theta^{(1)} - \theta^n) \tfrac{\partial \Pi^n}{\partial x} + \theta^{(1)} \tfrac{\partial}{\partial x}(\Pi') \right\} \right] , \tag{2.1.9}$$

$$v^{(2)} = v^{(1)} - \alpha\Delta t \left[f(u^{(2)} - u^n) + C_p \left\{ (\theta^{(1)} - \theta^n) \tfrac{\partial \Pi^n}{\partial y} + \theta^{(1)} \tfrac{\partial}{\partial y}(\Pi') \right\} \right] , \tag{2.1.10}$$

$$w^{(2)} = w^{(1)} + \alpha\Delta t C_p \left\{ (\theta^n - \theta^{(2)}) \tfrac{\partial \Pi^n}{\partial z} - \theta^{(1)} \tfrac{\partial}{\partial z}(\Pi') \right\} . \tag{2.1.11}$$

Substituting $\theta^{(2)}$ from (2.1.8) into (2.1.11) and setting $G = 1 - \alpha^2 \Delta t^2 C_p \frac{\partial \theta^{(1)}}{\partial z} \frac{\partial \Pi^n}{\partial z}$,

$$G w^{(2)} = w^{(1)} - (1 - G) w^n - \alpha\Delta t C_p (\theta^{(1)} - \theta^n) \frac{\partial \Pi^n}{\partial z} - \alpha\Delta t C_p \theta^{(1)} \frac{\partial}{\partial z}(\Pi') \,. \tag{2.1.12}$$

The evolution of the equation for density is handled in an Eulerian flux-form (see [28]):

$$\rho^{(2)} = \rho^n - \Delta t \left[\frac{\partial}{\partial x} \left(\rho^n (u^n + \alpha u') \right) + \frac{\partial}{\partial y} \left(\rho^n (v^n + \alpha v') \right) + \frac{\partial}{\partial z} \left(\rho^n (w^n + \alpha w') \right) \right] .$$

Finally, there is no time derivative of Π in the equation of state, and so the equation is assumed to hold for the latest estimates of each variable, denoted by superscript (2):

$$\kappa \Pi^{(2)} \theta^{(2)} \rho^{(2)} = \frac{p^{(2)}}{C_p} \,. \tag{2.1.13}$$

Since these equations are coupled, we must solve them simultaneously to obtain $\mathbf{X}^{(2)}$. The aim of the SI discretisation is to advance the model variables from $\mathbf{X}^n$ to $\mathbf{X}^{n+1}$, and this can be done by finding $\mathbf{X}'$ instead. (2.1.13) is rewritten in terms of the increments as

$$\kappa (\Pi^n + \Pi')(\theta^n + \theta')(\rho^n + \rho') = \frac{p^n + p'}{C_p} \,. \tag{2.1.14}$$

Now using the definition of Π from (2.1.1) and a linear approximation,

$$\Pi^n + \Pi' = \left(\frac{p^n + p'}{p_0} \right)^\kappa = \left(\frac{p^n}{p_0} \right)^\kappa \left(1 + \frac{p'}{p^n} \right)^\kappa \approx \Pi^n \left(1 + \frac{\kappa p'}{p^n} \right) ,$$

and we obtain the following approximation to the pressure increment

$$p' \approx \frac{p^n}{\kappa} \frac{\Pi'}{\Pi^n} \,. \tag{2.1.15}$$

Now by expanding (2.1.14) and neglecting products of two or more primed quantities (e.g. $\Pi' \theta' \rho' \approx \Pi' \theta' \rho^n \approx 0$) we have

$$\kappa\Pi'\theta^n\rho^n + \kappa\Pi^n\theta'\rho^n + \kappa\Pi^n\theta^n\rho' + \kappa\Pi^n\theta^n\rho^n = \frac{p^n + p'}{C_p}. \tag{2.1.16}$$

Then using (2.1.15) to eliminate p' from (2.1.16) and rearranging gives the following linearized form of the equation of state:

$$-\left(\frac{p^n}{\kappa C_p \Pi^n} - \kappa\theta^n\rho^n\right)\Pi' + \kappa\Pi^n\theta^n\rho' + \kappa\Pi^n\rho^n\theta' = \frac{p^n}{C_p} - \kappa\Pi^n\theta^n\rho^n. \tag{2.1.17}$$

The unknown quantities in (2.1.17) are Π', θ' and ρ'. We find that

$$\begin{aligned}
\theta' &= \left(\theta^{(1)} - \theta^n\right) - \alpha\Delta t w' \frac{\partial\theta^{(1)}}{\partial z} \;=\; -\alpha\Delta t w' \frac{\partial\theta^{(1)}}{\partial z} + X_\theta\,, \\
\rho' &= -\Delta t\left[\frac{\partial}{\partial x}\left(\rho^n(u+\alpha u')\right) + \frac{\partial}{\partial y}\left(\rho^n(v+\alpha v')\right) + \frac{\partial}{\partial z}\left(\rho^n(w+\alpha w')\right)\right],
\end{aligned}$$

where

$$\begin{aligned}
u' &= (u^{(1)} - u^n) + \alpha\Delta t\left[f(v^{(2)} - v^n) - C_p\left\{(\theta^{(1)} - \theta^n)\tfrac{\partial\Pi^n}{\partial x} + \theta^{(1)}\tfrac{\partial}{\partial x}(\Pi')\right\}\right] \\
&= -A\alpha\Delta t C_p\theta^{(1)}\tfrac{\partial}{\partial x}(\Pi') - F\alpha\Delta t C_p\theta^{(1)}\tfrac{\partial}{\partial y}(\Pi') + X_u\,, \qquad (2.1.18) \\
v' &= (v^{(1)} - v^n) - \alpha\Delta t\left[f(u^{(2)} - u^n) + C_p\left\{(\theta^{(1)} - \theta^n)\tfrac{\partial\Pi^n}{\partial y} + \theta^{(1)}\tfrac{\partial}{\partial y}(\Pi')\right\}\right] \\
&= -A\alpha\Delta t C_p\theta^{(1)}\tfrac{\partial}{\partial y}(\Pi') + F\alpha\Delta t C_p\theta^{(1)}\tfrac{\partial}{\partial x}(\Pi') + X_v\,, \qquad (2.1.19) \\
w' &= G^{-1}\left\{(w^{(1)} - w^n) - \alpha\Delta t C_p(\theta^{(1)} - \theta^n)\frac{\partial\Pi^n}{\partial z}\right\} - G^{-1}\alpha\Delta t C_p\theta^{(1)}\frac{\partial}{\partial z}(\Pi') \\
&= -G^{-1}\alpha\Delta t C_p\theta^{(1)}\tfrac{\partial}{\partial z}(\Pi') + X_w\,, \qquad (2.1.20)
\end{aligned}$$

where $A = 1/(1+\alpha^2\Delta t^2 f^2)$, $F = \alpha\Delta t f A$ and X_θ, X_u, X_v and X_w contains explicit terms that are already known. We substitute the equations for u', v' and w' into the equation for ρ', and substitute w' into θ'. Then we substitute ρ' and θ' into (2.1.17) to yield the following Helmholtz equation for the remaining unknown Π'

$$\begin{aligned}
-\kappa\Pi^n\theta^n\Delta t^2\alpha^2 C_p A\Bigg\{&\frac{\partial}{\partial x}\left[\rho^n\theta^{(1)}\frac{\partial\Pi'}{\partial x} + F\rho^n\theta^{(1)}\frac{\partial\Pi'}{\partial y}\right] \\
&+\frac{\partial}{\partial y}\left[\rho^n\theta^{(1)}\frac{\partial\Pi'}{\partial y} + F\rho^n\theta^{(1)}\frac{\partial\Pi'}{\partial x}\right] \\
&+\frac{\partial}{\partial z}\left[G^{-1}A^{-1}\rho^n\theta^{(1)}\frac{\partial\Pi'}{\partial z}\right]\Bigg\} - \left(\kappa\Pi^n\Delta t^2\alpha^2 C_p\rho^n\theta^{(1)}G^{-1}\frac{\partial\theta^{(1)}}{\partial z}\right)\frac{\partial\Pi'}{\partial z} \\
&+\left(\frac{p^n}{\kappa C_p\Pi^n} - \kappa\theta^n\rho^n\right)\Pi' = \kappa\Pi^n\theta^n\rho^n - \frac{p^n}{C_p} + \Phi^n\,, \qquad (2.1.21)
\end{aligned}$$

where Φ^n contains physical parameters and terms evaluated at the previous time step.

This equation contains cross derivative terms that introduce several unwanted difficulties. It is therefore more practical to introduce further simplifications to the model instead, and one way of simplifying the model is to assume a constant Coriolis term, i.e. set $f = f_0$. This results in A and F being constant. In addition we can enforce an averaging for ρ^n and $\theta^{(1)}$, meaning they will not vary with latitude or longitude. This means the cross derivative terms cancel, and the $\rho^n\theta^{(1)}$ terms can be pulled out of the derivatives. With these simplifications imposed, we obtain the following simplified model equation that is more amenable to practical implementation:

$$\begin{aligned}&-\kappa\Pi^n\theta^n\Delta t^2\alpha^2 C_p\rho^n\theta^{(1)}A\left\{\frac{\partial^2\Pi'}{\partial x^2}+\frac{\partial^2\Pi'}{\partial y^2}+\frac{\partial}{\partial z}\left(\frac{1}{GA}\frac{\partial\Pi'}{\partial z}\right)\right\}\\&\quad-\left(\kappa\Pi^n\Delta t^2\alpha^2 C_p\rho^n\theta^{(1)}G^{-1}\frac{\partial\theta^{(1)}}{\partial z}\right)\frac{\partial\Pi'}{\partial z}+\left(\frac{p^n}{\kappa C_p\Pi^n}-\kappa\theta^n\rho^n\right)\Pi'=\kappa\Pi^n\theta^n\rho^n-\frac{p^n}{C_p}+\Phi^n\end{aligned} \tag{2.1.22}$$

We assume that the Earth's surface and the top of the atmosphere are rigid boundaries, thus impose rigid boundary conditions $\mathbf{u}'\cdot\mathbf{n}=0$ at these boundaries, where $\mathbf{n}$ is a unit vector pointing outward from the boundary. This implies the Neumann boundary condition

$$\frac{\partial\Pi'}{\partial n}=-\mathbf{X}\cdot\mathbf{n}\,, \tag{2.1.23}$$

using equations (2.1.18) – (2.1.20). Once the equation is solved for Π', we use this to update the Exner pressure at time level $n+1$. This is used to find u^{n+1}, v^{n+1} and w^{n+1} which are then used to find θ^{n+1} and ρ^{n+1}. The papers [28, 31] also describe the derivation of the Helmholtz equation in the UM but without the assumptions made in this section. In [57, §3.2], the Helmholtz equation used in ENDgame is described, which is a simplified equation comparable to (2.1.22), obtained by making suitable assumptions as in this section to simplify the model.

The solution to the Helmholtz equation is therefore central to advancing the model at each time step, and so it must be solved efficiently to reduce the costs of the UM. The equation is solved operationally using a conjugate residual solver [34] on a grid with resolution $432\times325\times70$. This method alone is not adequate for solving the problem efficiently, thus it is preconditioned using a two dimensional alternative direction implicit (ADI) method [11]. The ADI method uses a tridiagonal solver in the x- and $z-$directions, and the affect of this as a preconditioner is investigated in [22, 69]. Although the preconditioner accelerates the method, it is not optimal and the time taken to solve the equation is still not adequate. So further improvements are sought especially for the huge problem sizes used in the UM.

We conclude this section by defining the Helmholtz equation in spherical polar coordinates, since we are interested in solving this equation on a spherical geometry. We have $r \in [a, a+d]$ (the radius), $\phi \in [0, \pi]$ (the polar angle) and $\lambda \in [0, 2\pi]$ (the azimuthal angle), where $a \approx 6371$km is the radius of the Earth and $d \approx 63$km is the depth of the atmosphere. The governing equations in spherical polar coordinates, given in [28], are:

$$
\begin{aligned}
&\frac{Du}{Dt} = fv - \frac{C_p\theta}{r\sin\phi}\frac{\partial \Pi}{\partial \lambda} + F_u\,,\\
&\frac{Dv}{Dt} = -fu - \frac{C_p\theta}{r}\frac{\partial \Pi}{\partial \phi} + F_v\,,\\
&\frac{Dw}{Dt} = -g - C_p\theta\frac{\partial \Pi}{\partial r} + F_w\,,\\
&\frac{D\theta}{Dt} = S\,,\\
&\frac{\partial \rho}{\partial t} + \nabla_z\cdot(\rho\mathbf{u}) = \frac{\partial \rho}{\partial t} + \frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\rho w\right) + \frac{1}{r\sin\phi}\frac{\partial}{\partial \phi}\left(\sin\phi\ \rho v\right) + \frac{1}{r\sin\phi}\frac{\partial}{\partial \lambda}(\rho u) = 0\,,\\
&\kappa\Pi\theta\rho = \frac{p}{C_p}\,,
\end{aligned}
$$

where $f = -2\Omega\cos\phi$. Then the Helmholtz equation, with the same simplifications made for (2.1.22), is

$$
\boxed{
\begin{aligned}
&-\kappa\Pi^n\theta^n\Delta t^2\alpha^2 C_p\rho^n\theta^{(1)}A\left\{\frac{1}{r^2\sin^2\phi}\frac{\partial^2\Pi'}{\partial\lambda^2} + \frac{1}{r^2\sin\phi}\frac{\partial}{\partial\phi}\left(\sin\phi\frac{\partial\Pi'}{\partial\phi}\right) + \frac{1}{r^2}\frac{\partial}{\partial r}\left(\frac{r^2}{GA}\frac{\partial\Pi'}{\partial r}\right)\right\}\\
&-\left(\kappa\Pi^n\Delta t^2\alpha^2 C_p\rho^n\theta^{(1)}G^{-1}\frac{\partial\theta^{(1)}}{\partial r}\right)\frac{\partial\Pi'}{\partial r} + \left(\frac{p^n}{\kappa C_p\Pi^n} - \kappa\theta^n\rho^n\right)\Pi' = \kappa\Pi^n\theta^n\rho^n - \frac{p^n}{C_p} + \Phi^n
\end{aligned}
}
\tag{2.1.24}
$$

with Neumann boundary conditions (2.1.23) at the rigid boundaries, i.e. the Earth's surface and the top of the atmosphere.

2.2 Data Assimilation in NWP

Forecasts using NWP have improved greatly since they were introduced over 50 years ago, and one of the main reasons for this is due to the improvement in obtaining the initial conditions, $\mathbf{x}_0$, at a given time, t_0, for the model forecast. $\mathbf{x}_0$ is a state of the atmosphere at time t_0 obtained for the 'model variables'. The model variables of the UM used at the Met Office are

$$\mathbf{x} = (u, v, w, \theta, \rho, p, q),$$

where (u, v, w) are the three components of wind velocity, θ is the potential temperature, ρ is density, p is pressure and q is specific humidity. These variables are highly correlated (e.g. a change in pressure will affect the density of the air) so any errors in these variables will also be correlated. The model domain used has $432 \times 325 \times 70$ degrees of freedom, and so the state vector of the model variables will have $\mathcal{O}(10^7)$ elements for each variable. The analysis grid used for data assimilation, however, has a smaller resolution of $216 \times 163 \times 70$ degrees of freedom, and so the initial conditions obtained using data assimilation are interpolated before they are used in the forecast.

Data Assimilation is the process of finding the best estimate of the current state of a system. In NWP, this system is the atmosphere and the oceans, and data assimilation is used to estimate $\mathbf{x}_0$ by combining a previous forecast with observational data, knowledge of atmospheric dynamics and statistical data that measures the accuracy of the forecast and observations. The uncertainties are quantified with Gaussian probability density functions (PDF). Typically there are $\mathcal{O}(10^6)$ observations and $\mathcal{O}(10^7)$ model variables, thus the system is underobserved and an operator is required to map the model space to the observation space. A model state is found that is the statistically optimal estimate of the truth given the previous forecast and new observations, together with estimates of the errors in each. The errors in the observations and the previous forecast are minimized in order to produce the best estimate, or 'analysis', of the current state of the atmosphere. Due to the chaotic nature of the governing equations in the model, any errors in the initial conditions will be amplified in the forecast. Thus, despite the continuous advancements in computational power and numerical methods, these benefits cannot be fully realized in NWP without accurate data assimilation techniques.

There are several type of data assimilation methods such as optimal statistical interpolation, Newtonian relaxation, 3D-VAR and 4D-VAR (see [56] for details). Also, ensemble-based methods such as ensemble Kalman filter methods [35] are being explored as possible additions to current operational analysis techniques. However, in this section we focus on 3D and 4D VAR, as these are currently the most commonly used data assimilation methods in weather forecasting, in particular by the Met Office who use an incremental version of 4D-VAR, described in [53, 61].

2.2.1 3D-VAR

We denote the current state of the atmosphere obtained from a previous forecast as the 'background state', $\mathbf{x}_b$. The background state will have uncertainties, and these are quantified in the 'background error covariance matrix', $\mathbf{B}$, using the PDF associated with the background state. The matrix $\mathbf{B}$ is of size $O(10^7 \times 10^7)$. If $\mathbf{x}$ is the 'true state'

of the atmosphere, then the error in the background state is quantified as

$$\mathbf{x}' = \mathbf{x} - \mathbf{x}_b, \tag{2.2.1}$$

where $\mathbf{x}'$ is referred as the increment, or perturbation, to the background state. We combine the forecast error with the observation errors, which is quantified in the following cost function:

$$J(\mathbf{x}) = J_b + J_o = (\mathbf{x} - \mathbf{x}_b)\mathbf{B}^{-1}(\mathbf{x} - \mathbf{x}_b) + (\mathbf{y} - \mathcal{H}(\mathbf{x}))\mathbf{R}^{-1}(\mathbf{y} - \mathcal{H}(\mathbf{x}))\,,$$

where $\mathbf{y}$ is the set of observations, $\mathcal{H}$ is a nonlinear observation operator that maps the model space to the observation space and $\mathbf{R}$ is the observation error covariance matrix of size $O(10^6 \times 10^6)$. J_b and J_o are the cost functions for the background state and observations respectively, and the objective of 3D-VAR is to find the model state $\mathbf{x}$ that minimizes the nonlinear cost function J. However, for operational purposes, it is more practical to linearise the cost function with respect to the increments to the background state:

$$J(\mathbf{x}') = \mathbf{x}'\mathbf{B}^{-1}\mathbf{x}' + (\mathbf{y} - \mathcal{H}(\mathbf{x}_b) - H(\mathbf{x}'))\mathbf{R}^{-1}(\mathbf{y} - \mathcal{H}(\mathbf{x}_b) - H(\mathbf{x}'))\,, \tag{2.2.2}$$

where H is a linear approximation to the observation operator $\mathcal{H}$.

2.2.2 4D-VAR

In 4D-VAR, the background error is modified not only in space but also over a certain time span called the 'assimilation window'. The assimilation window typically spans a 12 hour time period around the analysis time, and the objective of 4D-VAR is to minimize the misfit between the previous forecast and the observations during this time period. We describe a variant of 4D-VAR called strong constraint 4D-VAR, in which the model state $\mathbf{x}_0$ at time $t = t_0$ is found that minimizes the following cost function

$$J(\mathbf{x}_0) = (\mathbf{x}_0 - \mathbf{x}_b)\mathbf{B}^{-1}(\mathbf{x}_0 - \mathbf{x}_b) + \sum_{i=0}^{n}(\mathbf{y}_i - \mathcal{H}_i(\mathbf{x}_i))\mathbf{R}^{-1}(\mathbf{y}_i - \mathcal{H}_i(\mathbf{x}_i))\,,$$

with the constraint

$$\mathbf{x}_i = \mathcal{M}(t_i, t_0, \mathbf{x}_0)\,.$$

where $\mathcal{M}$ is the nonlinear time dependent model from Section 2.1 and i is an index of timesteps. By minimizing this cost function, $\mathbf{x}_0$ is found at the start of the assimilation

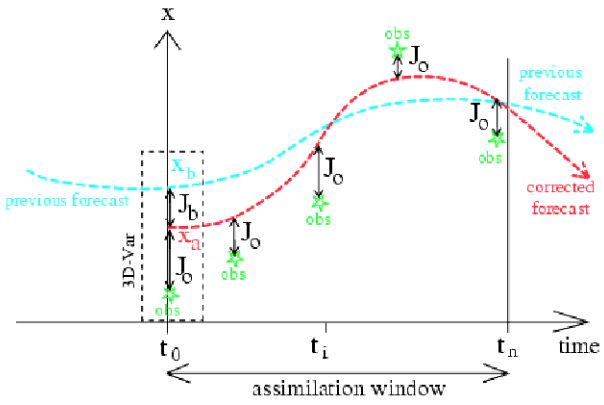

Figure 2-1: 4D-VAR

window. Then the corrected forecast across the assimilation window, produced by evolving the model $\mathcal{M}$, minimizes the errors in the forecast and the observations during this time period. Figure 2-1 demonstrates the process of 4D-VAR, where t_n is the end of the assimilation window. As with 3D-VAR, the 4D-VAR cost function is linearized with respect to the model state increments for a more practical implementation. In addition to the observation operator, a linearized approximation to the model is also used which updates the model space increments at each time step in the assimilation window.

2.2.3 Control Variable Transforms in 3D-VAR

Minimizing the cost function in 3D- or 4D-VAR is hugely costly due to the huge problem size involved and the sheer cost in inverting the **B** matrix. The matrix represents the errors in the background state which are highly correlated due to the strong correlations between the model variables, such as density and pressure. These correlations in the errors are represented in the matrix, hence **B** is dense and is in fact too large to be used or even stored explicitly. Therefore attempting to invert the matrix operationally is completely impractical. Instead, a new representation of the matrix is devised which is better conditioned and has a simpler structure, consequently allowing the minimization process to be performed more efficiently.

As with other assimilation schemes, a new representation of the **B** matrix is achieved in the Met Office VAR by a transformation from the space of model variables, e.g. pressure, potential temperature and wind velocity, to a space of variables known as the 'control variables', e.g. streamfunction, velocity potential and geostrophically un-

balanced pressure. Then it is in this control space that the minimization process is performed. The process of the control variable transform (CVT) is done operationally at the Met Office within incremental 4D-VAR, but in this section we describe the process in 3D-VAR for simplicity.

Firstly, instead of carrying out the transformations using the model variables, $\mathbf{x}$, we formulate the problem in terms of their increments from the background state, $\mathbf{x}'$ (recall (2.2.1)), because it is with respect to these increments that the cost function is minimized (see (2.2.2)). We have

$$\mathbf{x}' = (u', v', w', \theta', \rho', p', q')\,,$$

and we let, for example,

$$\begin{aligned} u &= \overline{u} + u'\,, \\ v &= \overline{v} + v'\,, \\ p &= \overline{p} + p'\,, \end{aligned}$$

where $\overline{u}$, $\overline{v}$ and $\overline{p}$ are the background states of the variables. Now, let $\mathbf{v}$ be the set of control variables. Then $\mathbf{v}'$ is the set of increments of the control variables, and without giving the details the transformation from the model variable increments to the control variable increments can be compactly formulated as

$$\mathbf{v}' = \mathbf{T}\mathbf{x}', \tag{2.2.3}$$

for some matrix $\mathbf{T}$, which we denote the $\mathbf{T}$-transform. It's inverse, i.e. the transformation from $\mathbf{v}'$ to $\mathbf{x}'$ is denoted the $\mathbf{U}$-transform

$$\mathbf{x}' = \mathbf{U}\mathbf{v}'. \tag{2.2.4}$$

The choice of the control variables is based on the fact that the background errors for each control variable are assumed uncorrelated with each other. Then the background error covariance matrix for the control variables, $\mathbf{B}_v$, will only contain spatial correlations in the errors for each control variable, and can be assumed to be block diagonal. $\mathbf{B}_v$ is obtained from $\mathbf{B}$ by applying the $\mathbf{U}$-transform (2.2.4) to the first term of the cost function (2.2.2):

$$\begin{aligned} \mathbf{x}'^T\mathbf{B}^{-1}\mathbf{x}' &= \mathbf{v}'^T\mathbf{U}^T\mathbf{B}^{-1}\mathbf{U}\mathbf{v}' \\ \Rightarrow\ \mathbf{B}_v^{-1} &= \mathbf{U}^T\mathbf{B}^{-1}\mathbf{U}\,. \end{aligned}$$

With this simpler structure of the matrix, the minimization process can be done more easily in the control space. A simpler form of (2.2.2) is now obtained:

$$J(\mathbf{v}') = \mathbf{v}'\mathbf{B}_v^{-1}\mathbf{v}' + (\mathbf{y} - \mathcal{H}(\mathbf{U}\mathbf{v}_b) - H(\mathbf{U}\mathbf{v}'))\mathbf{R}^{-1}(\mathbf{y} - \mathcal{H}(\mathbf{U}\mathbf{v}_b) - H(\mathbf{U}\mathbf{v}'))\,.$$

Once the minimization is completed, the solution $\mathbf{v}'$ is transformed back to the model space via the $\mathbf{U}$-transform to give the solution $\mathbf{x}'$ in the model space. Operationally at the Met Office, the transformations are further extended to remove the spatial correlations of each variable using vertical and horizontal transforms [49]. By doing this, the matrix can be assumed to be diagonal, hence it doesn't need to be stored explicitly. Let $\mathbf{z}'$ be the set of variable increments which have undergone a control variable and spatial transform. Then

$$\mathbf{z}' = \mathbf{T}_h\mathbf{T}_v\mathbf{T}\mathbf{x}',$$
$$\mathbf{x}' = \mathbf{U}\mathbf{U}_v\mathbf{U}_h\mathbf{z}',$$

where $\mathbf{T}_h$ and $\mathbf{T}_v$ are the horizontal and vertical transforms which remove the spacial correlations of the variables, and $\mathbf{U}_h$ $\mathbf{U}_v$ are the respective inverses. The transformed background error covariance matrix, Λ, is simply the identity matrix:

$$I = \Lambda^{-1} = \mathbf{T}_h\mathbf{T}_v\mathbf{T}\mathbf{B}^{-1}\mathbf{U}\mathbf{U}_v\mathbf{U}_h\,.$$

Therefore, the problem of storing and inverting the $\mathbf{B}$-matrix has been shifted to defining suitable control variable and spacial transforms, and so the matrix never needs to be explicitly represented.

2.2.4 Choice of Control Variable Transform

We now discuss the physics behind the the choice of the control variables. It is important that these variables are uncorrelated, or at least nearly uncorrelated, otherwise the assumption of the transformed background error covariance matrix having a diagonal structure will not be accurate.

Although the Met Office uses compressible non-hydrostatic equations, here we only analyze the hydrostatic motions. In [49, 53], it was identified that there are two modes of atmospheric motion, geostrophically balanced Rossby waves and geostrophically unbalanced inertia-gravity waves. Balanced and unbalanced flows are assumed to have little or no interaction between each other, meaning that their errors will be uncorrelated. In a linear shallow atmosphere system, one third of the modes are characterized as balanced and two thirds as unbalanced [5], and this can be extended to a 3D atmo-

sphere with the assumption that the modes will still be uncorrelated. This provides the inspiration for the choice of control variables in the CVT for the operational VAR system. Good control variables are ones which capture either the balanced or unbalanced modes, and a separation of control variables in such a way forms the basis of the idea that their errors will be uncorrelated. At the Met Office the control variables currently used operationally are:

- The streamfunction, ψ
- The velocity potential, χ
- The unbalanced pressure, p_u

The streamfunction is assumed to be a completely balanced variable, whilst the velocity potential is assumed unbalanced. In general, pressure and streamfunction have balanced and unbalanced components, and likewise for their perturbations:

$$p' = p'_u + p'_b, \tag{2.2.5}$$

$$\psi' = \psi'_u + \psi'_b. \tag{2.2.6}$$

However, for the CVT, it is assumed that $\psi'_u = 0$ because the streamfunction is assumed to be completely balanced. Since a shallow atmosphere approximation is used for the analysis of the modes, the depth of the atmosphere is assumed to have a constant value of a.

The T-transform: $(u', v', w', \theta', \rho', p', q') \rightarrow (\psi', \chi', p'_u)$

We now give the details of the **T**-transform, i.e. how to calculate each of the control variables from the model variables. Note that there are more model variables than control variables, but in order to carry out the **T**-transform we only need the three model variables (u', v', p'). The streamfunction and velocity potential increments are found from the horizontal wind components by solving the following elliptic equations on each vertical level of the sphere

$$\nabla_r^2 \psi' = \frac{1}{a \sin\phi} \frac{\partial v'}{\partial \lambda} - \frac{1}{a} \frac{\partial u'}{\partial \phi}, \tag{2.2.7}$$

$$\nabla_r^2 \chi' = \frac{1}{a \sin\phi} \frac{\partial u'}{\partial \lambda} + \frac{1}{a} \frac{\partial v'}{\partial \phi}, \tag{2.2.8}$$

where ∇_r^2 is the 2D Laplacian on the sphere:

$$\nabla_r^2 = \frac{1}{a^2 \sin\phi} \left(\frac{\partial}{\partial \phi} \left(\sin\phi \frac{\partial}{\partial \phi} \right) + \frac{\partial}{\partial \lambda} \left(\frac{1}{\sin\phi} \frac{\partial}{\partial \lambda} \right) \right).$$

Since the right hand sides of (2.2.7) and (2.2.8) have a zero mean value on the sphere, this ensures the existence of a solution in both equations which will be unique up to a constant.

Then, the linear balance equation

$$\nabla_r \cdot (f\rho_0 \nabla_r \psi_b') - \nabla_r^2 p_b' = 0, \tag{2.2.9}$$

for the increments of ψ_b' and p_b' is used to calculate the balanced pressure field where ρ_0 is a reference state density and f is the Coriolis force which is latitude dependant. Since the streamfunction is assumed to be completely balanced, we have $\psi' = \psi_b'$, and so p_b' can be calculated. The full pressure increment p' is known because it is a model variable, so the unbalanced pressure is calculated trivially from (2.2.5).

The U-transform: $(\psi', \chi', p_u') \to (u', v', w', \theta', \rho', p', q')$

We now give the details of the transformation from the control variables to the model variables, i.e. the **U**-transform. The three control variables are transformed to the model variables (u', v', p'), which are then used to obtain the remaining model variables (w', θ', ρ', q').

Firstly, a Helmholtz decomposition is used to separate velocities into rotational and divergent parts. Therefore we have

$$\begin{pmatrix} u' \\ v' \end{pmatrix} = \nabla_r \times \psi' + \nabla_r \chi' \,, \tag{2.2.10}$$

which gives

$$u' = \frac{1}{a \sin\phi} \frac{\partial \chi'}{\partial \lambda} - \frac{1}{a} \frac{\partial \psi'}{\partial \phi} \,, \tag{2.2.11}$$

$$v' = \frac{1}{a \sin\phi} \frac{\partial \psi'}{\partial \lambda} + \frac{1}{a} \frac{\partial \chi'}{\partial \phi} \,. \tag{2.2.12}$$

We then obtain the pressure by firstly calculating the balanced pressure, p_b'. This is calculated by solving (2.2.9) on each vertical layer, again with the assumption that $\psi' = \psi_b'$. We know p_u' as it is a control variable so we obtain the full pressure field using (2.2.5).

With the pressure increment obtained, we then use (2.1.1) and the following linearized hydrostatic equation (as stated in [80]) to obtain the *virtual* potential temperature increment $\theta^{v'}$:

$$\frac{\partial \Pi'}{\partial r} = -\frac{g}{c_p \theta^{v'}} \,.$$

The potential temperature, θ' and specific humidity, q', increments are then calculated by linearising the following equations

$$\theta^{\nu'} = \theta'(1 + [\epsilon^{-1} - 1]q'), \tag{2.2.13}$$

$$rh' = q' \left(\frac{p'}{\epsilon e_s(T)} \right), \tag{2.2.14}$$

and then solving them for θ' and q', where ϵ is the ratio of the molecular weight of water to the molecular weight of dry air, $e_s(T)$ is the saturated vapour pressure of water and rh' is the relative humidity increment.

The density increment is obtained by rearranging the following equation from [80]:

$$p' = \left[\frac{R\rho'\theta^{v'}}{a^2 p_{ref}} \right]^{\frac{1}{1-\kappa}}. \tag{2.2.15}$$

Finally we obtain the vertical velocity increment w'. Operationally in the UM, this is currently set as $w' = 0$, but a more accurate value can be obtained by solving a particular equation known as the *Quasi Geostrophic Omega* ($QG - \Omega$) equation [38]:

$$\boxed{-N^2(r)\nabla_r^2(\rho_{ref}w') - \theta_{ref}\rho_{ref}f_0^2\frac{\partial}{\partial r}\left(\frac{1}{\theta_{ref}\rho_{ref}}\frac{\partial}{\partial r}(\rho_{ref}w')\right) = -P_w\theta_{ref}\rho_{ref}} \tag{2.2.16}$$

with boundary conditions $w' = 0$ on the upper and lower vertical boundaries. θ_{ref} and ρ_{ref} are hydrostatically balanced reference states of the potential temperature and density. $N^2(r) = \frac{g}{\theta_0}\frac{\partial\theta'}{\partial r}$, and f_0 is the Coriolis parameter, assumed to be a constant. The form of this equation is valid only if the Boussinesq approximation is valid and the equation is derived in the Met Office VAR scientific paper no. 16 (yet to be published). Clearly this is not the case for a nonhydrostatic model, but the errors are likely to be small enough for equation (2.2.16) to be a useful approximation. The $QG-\Omega$ equation is a three dimensional elliptic problem solved for $\rho_{ref}w'$ that is currently attempted to be solved using the generalized conjugate residual (GCR) method [34, 69]. It is a highly ill-conditioned problem due to the large variation in coefficients between the poles in addition to the large problem size of $\mathcal{O}(10^6)$. Hence, the GCR solver used at the Met Office takes over 700 iterations to solve this problem, and so there is considerable scope for improvement before the solver can be used operationally.

2.3 PV-Based Control Variable Transformations

The control variables described in Section 2.2.3 are chosen on the basis that they satisfy the linear balance equation (2.2.9) exactly. However, this assumption is only valid for balanced components of flow because the unbalanced components cannot obey the linear balance equation. The separation between balanced and unbalanced modes is different in different *Burger regimes* [26]. The Burger number, Bu, is the ratio L_R/L where L is the horizontal length scale and L_R is the *Rossby radius of deformation*, $L_R = \frac{\sqrt{gH}}{f}$, where H is the depth scale. Bu is used to characterize the flow regime, where the two regimes are as follows:

- High Burger regimes ($Bu \gg 1$): This is achieved when $L \ll L_R$, i.e. when f is small (e.g. at the tropics). Here the streamfunction is dominated by balanced components, so $\psi = \psi_b$.
- Low Burger regimes ($Bu \ll 1$): This is achieved when $L \gg L_R$, where the pressure field is dominated by balanced components, so $p \approx p_b$ and in general $\psi_u \neq 0$.

This indicates that the control variables defined in Section 2.2.4 will capture high Burger regimes accurately because of the assumption that ψ is completely balanced. Solving the linear balance equation (2.2.9) for p_b' is therefore only valid if $L \ll L_R$ in the high Burger regimes. However, the assumption that $\psi = \psi_b$ is only satisfied in high Burger regimes and so the low Burger regimes are not captured accurately.

These limitations are overcome by using a different set of control variables which represent the separation of balanced and unbalanced components across all flow regimes. The balanced mode is captured using a quantity known as the *potential vorticity* (PV). According to the theory of [5], balanced flow is associated with PV but unbalanced flow has no PV and so is completely independent from balanced flow, hence they are uncorrelated. Thus we use this as motivation to choose PV-based control variables that will more accurately represent the flow in the two Burger regimes. These new variables are:

- The balanced streamfunction, ψ_b
- The velocity potential, χ
- The unbalanced pressure, p_u

Clearly the only difference is that ψ_b is used instead of ψ, thus the assumption that the streamfunction is balanced across all flow regimes is no longer used.

This new control variable transform using PV recognizes the presence of unbalanced streamfunction in low Burger regimes, but resembles the existing scheme (which is accurate) for high Burger regimes. Consequently the control variables can be partitioned into purely balanced and unbalanced parts. The PV-based control variable transform is based around two equations. In the UM, PV is called Ertel PV, Q, which is calculated from ψ and p, and an incremental form of the equation is given as follows

$$\alpha_0 \nabla_r^2 \psi' + \beta_0 p' + \gamma_0 \frac{\partial p'}{\partial r} + \varepsilon_0 \frac{\partial^2 p'}{\partial r^2} \;=\; Q' \,, \tag{2.3.1}$$

where α_0, β_0, γ_0 and ε_0 are reference state quantities of the model variables. The second is the linear balance equation (2.2.9) which relates the balanced components of ψ' and p'. Now, since there is no PV in unbalanced components of flow, (2.3.1) is written in terms of ψ'_u and p'_u as

$$\alpha_0 \nabla_r^2 \psi'_u + \beta_0 p'_u + \gamma_0 \frac{\partial p'_u}{\partial r} + \varepsilon_0 \frac{\partial^2 p'_u}{\partial r^2} \;=\; 0 \,, \tag{2.3.2}$$

Now, by substituting (2.2.5) and (2.2.6) into (2.3.1) and using (2.3.2) we get

$$\alpha_0 \nabla_r^2 \psi_b + \beta_0 p'_b + \gamma_0 \frac{\partial p'_b}{\partial r} + \varepsilon_0 \frac{\partial^2 p'_b}{\partial r^2} \;=\; Q' \,, \tag{2.3.3}$$

Now, only the balanced component of flow satisfies the linear balance equation (2.2.9), so replacing ψ'_b and p'_b with ψ'_u and p'_u respectively gives a residual, denoted anti-PV, $\overline{Q}$, which we calculate from p' and ψ'

$$\nabla_r \cdot (f\rho \nabla_r \psi'_u) - \nabla_r^2 p'_u = \overline{Q}' = \nabla_r^2 \xi' \,, \tag{2.3.4}$$

where ξ' is a measure of imbalance. Then adding the linear balance equation (2.2.9) to (2.3.4), we obtain

$$\nabla_r \cdot (f\rho \nabla_r \psi') - \nabla_r^2 p' = \nabla_r^2 \xi' \,, \tag{2.3.5}$$

by using again (2.2.5) and (2.2.6). We have now calculated PV and anti-PV from p' and ψ', and thus we now use these to calculate the control variables p'_u and ψ'_b. The balanced streamfunction is found by firstly using the linear balance equation (2.2.9) to solve for p'_b, which can be formally written using the inverse Laplacian operator ∇_r^{-2} as

$$p'_b = \nabla_r^{-2} \nabla_r \cdot (f\rho_0 \nabla_r \psi'_b) \,. \tag{2.3.6}$$

Substituting (2.3.6) into (2.3.3) we finally obtain the *balanced equation* for ψ'_b:

$$\boxed{\begin{aligned}\alpha_0\nabla_r^2\psi'_b + \beta_0\left(\nabla_r^{-2}\nabla_r\cdot f\nabla_r\psi'_b\right) + \gamma_0\frac{\partial}{\partial r}\left(\nabla_r^{-2}\nabla_r\cdot f\nabla_r\psi'_b\right) + \\ \varepsilon_0\frac{\partial^2}{\partial r^2}\left(\nabla_r^{-2}\nabla_r\cdot f\nabla_r\psi'_b\right) \;=\; Q'\end{aligned}} \tag{2.3.7}$$

A similar approach is used to find the unbalanced pressure. Rearranging (2.3.5) we get

$$p'_u = \nabla_r^{-2}\nabla_r\cdot(f\rho_0\nabla_r\psi'_u) - \xi', \tag{2.3.8}$$

and substituting (2.3.8) into (2.3.2), we obtain the *unbalanced equation* for ψ'_u:

$$\boxed{\begin{aligned}\alpha_0\nabla_r^2\psi'_u + \beta_0\left(\nabla_r^{-2}\nabla_r\cdot f\nabla_r\psi'_u\right) + \gamma_0\frac{\partial}{\partial r}\left(\nabla_r^{-2}\nabla_r\cdot f\nabla_r\psi'_u\right) + \\ \varepsilon_0\frac{\partial^2}{\partial r^2}\left(\nabla_r^{-2}\nabla_r\cdot f\nabla_r\psi'_u\right) \;=\; \beta_0\xi' + \gamma_0\frac{\partial\xi'}{\partial r} + \varepsilon_0\frac{\partial^2\xi'}{\partial r^2}\end{aligned}} \tag{2.3.9}$$

Then we use (2.3.8) to find p'_u.

Despite the convincing theory for the new PV-based control variable transformations (see [53, 6, 5, 25, 78, 81]), these are not yet operational. The main reason for this are the difficulties in solving (2.3.7) and (2.3.9) to obtain the control variables. The coefficients in the equations vary with height and latitude, and there are also two dimensional solves with ∇_r^{-2} required within the three dimensional problem. The result is a highly ill-conditioned problem and previous attempts at implementing a solver for this have not resulted in a satisfactory convergence to the solution. However, if these equations can be solved at all, let alone with great efficiency, then there is considerable scope for improving the accuracy of the initial conditions produced by VAR, particularly in the low Burger regimes.

2.4 Grid Structure for the UM

In this section we define the grids used for the discretisation of the equations in the UM and the CVT. The grid is regular in longitude λ and latitude ϕ, but graded in r with a higher resolution of grid points near the surface of the Earth. A graded vertical grid spacing is desirable since there are larger vertical gradients and fluxes of variables near the surface of the Earth. Also, a staggered grid is used in all three coordinate directions, where the Arakawa C-grid staggering [2] is used in the horizontal (i.e. the λ–ϕ plane, see Figure 2-2), while the Charney-Phillips grid staggering [1] is used in the vertical (see Figure 2-3).

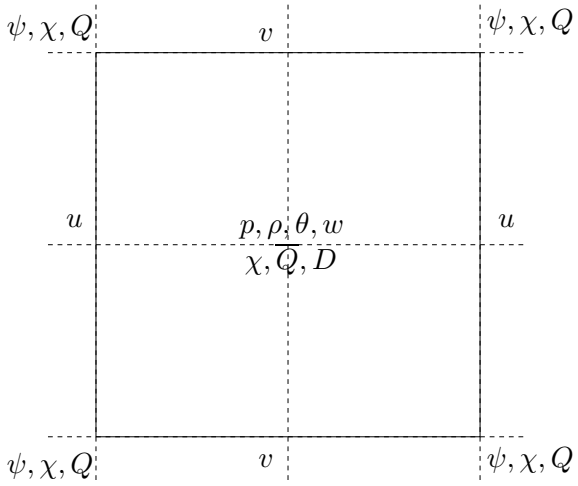

Figure 2-2: Arakawa C-grid used in the horizontal

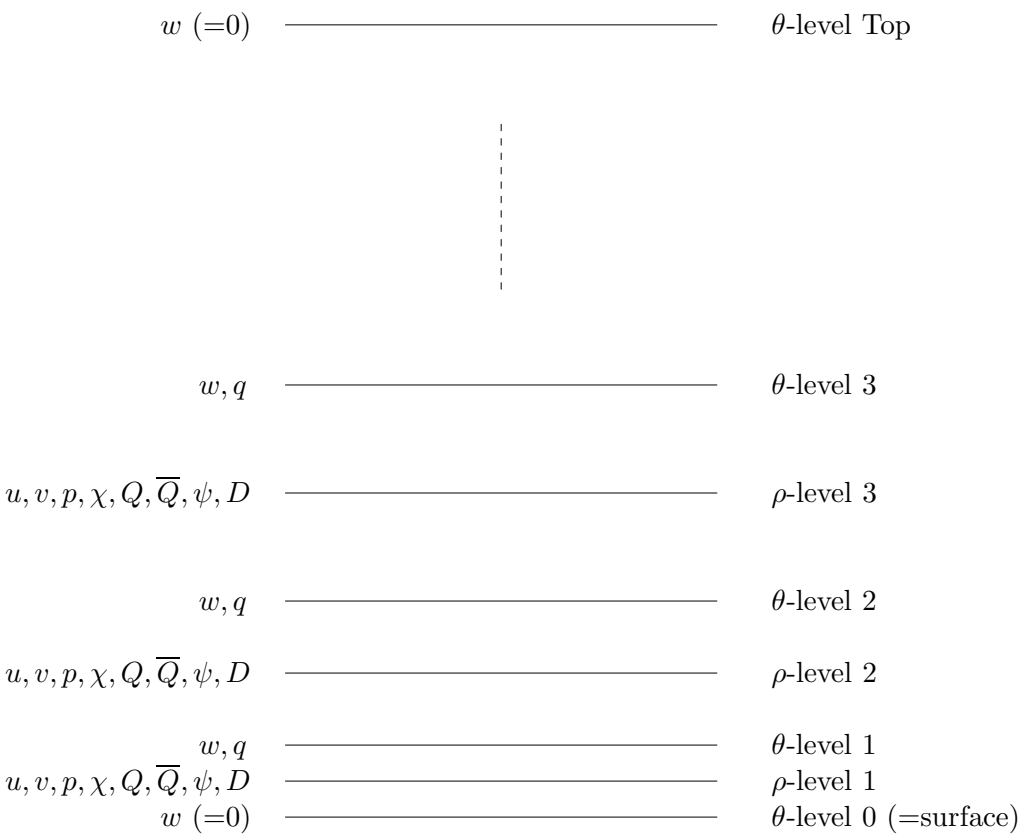

Figure 2-3: Charney Phillips grid used in the vertical

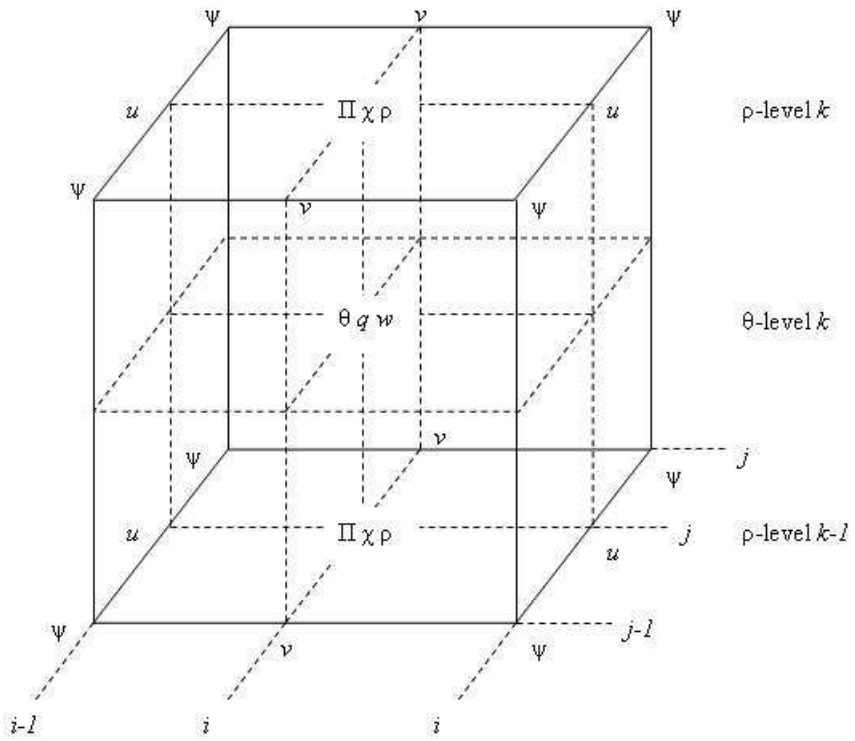

Figure 2-4: The Met Office grid staggering

In the horizontal, the variables Π, ρ, θ, w, q, χ and u are located at the poles. In the vertical, the grid is staggered into θ- and ρ-levels. There is an extra θ-level because these occupy both the upper and lower boundary, and the ρ-levels are located halfway between. The grid staggering and the position of each variable is visualized in three dimensions in Figure 2-4.

The equations in the UM and in the CVT are discretised spatially using finite differences, with a special treatment of the variables that are discretised at the poles. Details of the spacial discretisation can be found in [28], but will also be covered in detail in Chapter 3 using a *finite volume* discretisation where the discretisation at the poles can be derived more naturally.

2.5 Summary of the Key Elliptic Equations in NWP

We have seen throughout this chapter that the process of NWP and data assimilation leads to three types of three dimensional elliptic problems which are solved on a model

domain representing the atmosphere of the Earth:

- The Helmholtz problem (2.1.24), arising from the semi-implicit discretisation of the fully compressible non-hydrostatic equations used in the dynamical core of the UM.
- The Quasi Geostrophic Omega ($QG-\Omega$) equation (2.2.16) for finding the vertical velocity increment in the control variable transform, used in variational data assimilation.
- The balanced and unbalanced equations (2.3.7) and (2.3.9) for obtaining the control variables in the new PV-based control variable transform.

The three problems each have a particular role in NWP, and so it is essential to be able to solve them efficiently and accurately using sophisticated numerical methods. Although each of these equations are elliptic, they all pose different difficulties. For example, the coefficients present in each problem are different, leading to different techniques required to compensate for the variation in these coefficients. The balanced and unbalanced equations require additional two dimensional solves, so further techniques must be investigated to tackle this problem. Moreover, solving these equations on a spherical domain poses additional difficulties because the coefficients in the equations degenerate towards the poles. Therefore, it is clear that advanced numerical techniques are needed to aid the Met Office in solving these equations with greater efficiency than what they are currently capable of. This will consequently improve the forecasts from the NWP models by allowing for finer grid resolutions and by improving the quality of the initial conditions for the forecasts.

The solution of these type of problems using iterative numerical techniques will form the core of this thesis. In subsequent chapters we investigate various solvers that are known to be able to deal with the type of issues that arise from these problems, and adapt them accordingly for the three main elliptic problems of interest. Comparisons will be made with the solvers currently used at the Met Office, and if those solvers are significantly outperformed, it is likely that they will be replaced by the new solvers described in this thesis.

Chapter 3

Model Problem and Discretisation

3.1 Model Problem

All the problems described in the previous chapter (summarized in Section 2.5), when formulated in spherical coordinates and suitably scaled, take the general tensor product form

$$-\nabla \cdot (K \nabla u(\boldsymbol{\xi})) \; + \; \mathbf{a}(\boldsymbol{\xi}) \cdot \nabla u(\boldsymbol{\xi}) \; + \; c(\boldsymbol{\xi}) u(\boldsymbol{\xi}) \; = \; g(\boldsymbol{\xi}), \quad \forall \boldsymbol{\xi} = (\xi_1, \xi_2, \xi_3) \in (0,1)^3 \qquad (3.1.1)$$

with

$$K = K(\boldsymbol{\xi}) = \begin{bmatrix} \alpha_1(\boldsymbol{\xi}) & 0 & 0 \\ 0 & \alpha_2(\boldsymbol{\xi}) & 0 \\ 0 & 0 & \alpha_3(\boldsymbol{\xi}) \end{bmatrix},$$
$$\mathbf{a} = \mathbf{a}(\boldsymbol{\xi}) = [a_1(\boldsymbol{\xi}), a_2(\boldsymbol{\xi}), a_3(\boldsymbol{\xi})]^T,$$

and separable functions $\alpha_i(\boldsymbol{\xi}) = \alpha_i^1(\xi_1)\alpha_i^2(\xi_2)\alpha_i^3(\xi_3)$, for $i \in \{1,2,3\}$. The differential operators in (3.1.1) are the usual gradient and divergence operators (see [15]) in Cartesian form, i.e.

$$\left.\begin{aligned} \nabla u &= \left[\tfrac{\partial u}{\partial \xi_1}, \tfrac{\partial u}{\partial \xi_2}, \tfrac{\partial u}{\partial \xi_3}\right]^T \\ \nabla \cdot \mathbf{F} &= \tfrac{\partial F_1}{\partial \xi_1} \; + \; \tfrac{\partial F_2}{\partial \xi_2} \; + \; \tfrac{\partial F_3}{\partial \xi_3} \end{aligned}\right\}.$$

Let $\Omega = (0,1)^3$ with boundary Γ. K is assumed to be positive definite almost everywhere (a.e.) in Ω, i.e. $\alpha_i(\boldsymbol{\xi}) > 0$ for all $i \in \{1,2,3\}$ a.e., but we are particularly concerned with the highly anisotropic and degenerate cases where $\alpha_i(\boldsymbol{\xi}) \to 0$ as $\xi_i \to 0$ or $\xi_i \to 1$. These cases arise in for example spherical polar coordinates where the

functions α_i degenerate near the poles. In addition, for problems in numerical weather prediction, there will usually be a second anisotropy in that

$$\alpha_1(\boldsymbol{\xi}) \gg \alpha_2(\boldsymbol{\xi}), \alpha_3(\boldsymbol{\xi}).$$

This comes from the fact that the Earth's atmosphere is very thin compared to the dimensions of the Earth's surface. Note however, that crucially, these anisotropies are grid-aligned.

In particular, for the special case of the Poisson equation with source term f posed in the Earth's atmosphere and written in spherical polar coordinates (see [15]), we have

$$K = K(\boldsymbol{\xi}) = \begin{bmatrix} \left(\frac{a+d\xi_1}{d}\right)^2 \sin(\pi\xi_2) & 0 & 0 \\ 0 & \frac{\sin(\pi\xi_2)}{\pi^2} & 0 \\ 0 & 0 & \frac{1}{4\pi^2 \sin(\pi\xi_2)} \end{bmatrix}.$$

where ξ_1 is associated with the radial direction, ξ_2 with the polar angle and ξ_3 with the azimuthal angle. Note that we have nondimensionalized the problem by setting

$$\xi_1 := \frac{R-a}{d}, \quad \xi_2 := \frac{\Phi}{\pi}, \quad \xi_3 := \frac{\Lambda}{2\pi},$$

where $\{(R, \Phi, \Lambda) : a \leq R \leq a+d,\ 0 \leq \Phi \leq \pi,\ 0 \leq \Lambda \leq 2\pi\}$ are the usual spherical polar coordinates and $a \approx 6371$km and $d \approx 63$km are the Earth's radius and the depth of the atmosphere, respectively. Because $a \gg d$ and because $\sin(\pi\xi_2) \to 0$ as $\xi_2 \to 0$ or $\xi_2 \to 1$, we see that this problem is highly anisotropic as suggested above.

Note that the equation has to be scaled by the *volume element* of a sphere $(a+d\xi_1)^2 \sin(\pi\xi_2)$, defined in [15], so that it can be written in the general tensor product form of (3.1.1). With $\mathbf{a} = \mathbf{0}$ and $c = 0$ we have the Poisson equation:

$$-\frac{\partial}{\partial\xi_1}\left(\left(\frac{a+d\xi_1}{d}\right)^2 \sin(\pi\xi_2)\frac{\partial u}{\partial\xi_1}\right) - \frac{\partial}{\partial\xi_2}\left(\frac{\sin(\pi\xi_2)}{\pi^2}\frac{\partial u}{\partial\xi_2}\right) - \frac{\partial}{\partial\xi_3}\left(\frac{1}{4\pi^2\sin(\pi\xi_2)}\frac{\partial u}{\partial\xi_3}\right) = f(\gamma_1,\gamma_2,\gamma_3)(a+d\xi_1)^2\sin(\pi\xi_2), \tag{3.1.2}$$

where $g(\xi_1,\xi_2,\xi_3) = f(\gamma_1,\gamma_2,\gamma_3)$, and $\boldsymbol{\gamma}$ is a reparameterization that maps the unit square to the spherical shell with radius ranging from a to $a+d$ as follows:

$$\boldsymbol{\gamma}(\xi_1,\xi_2,\xi_3) = ((a+d\xi_1)\sin(\pi\xi_2)\cos(\pi\xi_3), (a+d\xi_1)\sin(\pi\xi_2)\sin(\pi\xi_3), (a+d\xi_1)\cos(\pi\xi_2))^T.$$

3.2 Finite Volume Discretisation

In this section we describe the discretisation of the general problem (3.1.1) with various different boundary conditions using the finite volume method, as described in [36, 77]. We then look at the special case of spherical polar coordinates with periodic boundary conditions at the east and west boundaries and "polar" boundaries at the north and south boundaries. This will closely follow the discretisation presented in [72] for the two dimensional (2D) Poisson equation on the surface of the sphere, and [7] which also describes the discretisation on the three dimensional (3D) spherical shell.

3.2.1 Discretisation of the General Problem

In order to discretise (3.1.1) on the unit cube, let us subdivide Ω into $n_1 \times n_2 \times n_3$ cubes with cell centres

$$\{\, (\xi_{1,i}, \xi_{2,j}, \xi_{3,k}) \;:\; i = 1, \ldots, n_1,\; j = 1, \ldots, n_2,\; k = 1, \ldots, n_3 \,\},$$

and edge lengths $h_1 = 1/n_1$, $h_2 = 1/n_2$ and $h_3 = 1/n_3$.

We discretise (3.1.1) using the cell centred finite volume method. The PDE is integrated over each mesh cell, or control volume, corresponding to a grid point $(\xi_{1,i}, \xi_{2,j}, \xi_{3,k})$, i.e.

$$\Omega_{i,j,k} \;=\; \left[\xi_{1,i-\frac{1}{2}}, \xi_{1,i+\frac{1}{2}}\right] \times \left[\xi_{2,j-\frac{1}{2}}, \xi_{2,j+\frac{1}{2}}\right] \times \left[\xi_{3,k-\frac{1}{2}}, \xi_{3,k+\frac{1}{2}}\right] ,$$

where $\xi_{1,i\pm\frac{1}{2}} = \xi_{1,i} \pm \frac{h_1}{2}$, $\xi_{2,j\pm\frac{1}{2}} = \xi_{2,j} \pm \frac{h_2}{2}$ and $\xi_{3,k\pm\frac{1}{2}} = \xi_{3,k} \pm \frac{h_3}{2}$. The boundary of each control volume consists of six faces, i.e.

$$\Gamma_{i,j,k} \;=\; \Gamma_{i-\frac{1}{2},j,k} \cup \Gamma_{i+\frac{1}{2},j,k} \cup \Gamma_{i,j-\frac{1}{2},k} \cup \Gamma_{i,j+\frac{1}{2},k} \cup \Gamma_{i,j,k-\frac{1}{2}} \cup \Gamma_{i,j,k+\frac{1}{2}} ,$$

and the cell faces are denoted by $\Gamma_{i\pm\frac{1}{2},j,k} \;=\; \{\xi_{1,i\pm\frac{1}{2}}\} \times [\xi_{2,j-\frac{1}{2}}, \xi_{2,j+\frac{1}{2}}] \times [\xi_{3,k-\frac{1}{2}}, \xi_{3,k+\frac{1}{2}}]$, with analogous definitions for $\Gamma_{i,j\pm\frac{1}{2},k}$ and $\Gamma_{i,j,k\pm\frac{1}{2}}$.

On each of the faces of Ω we impose either homogeneous Dirichlet boundary conditions, i.e.

$$u(\boldsymbol{\xi}) = 0 \qquad \forall \boldsymbol{\xi} \in \Gamma\,, \tag{3.2.1}$$

homogeneous Neumann boundary conditions, i.e.

$$\frac{\partial u}{\partial n}(\boldsymbol{\xi}) = 0 \qquad \forall \boldsymbol{\xi} \in \Gamma\,, \tag{3.2.2}$$

(where n denotes the outward normal to the boundary Γ), or periodic boundary con-

ditions, i.e.

$$u(0,\xi_2,\xi_3) = u(1,\xi_2,\xi_3)\,, \tag{3.2.3}$$

$$\frac{\partial u}{\partial \xi_1}(0,\xi_2,\xi_3) = \frac{\partial u}{\partial \xi_1}(1,\xi_2,\xi_3) \qquad \forall \xi_2 \in [0,1], \forall \xi_3 \in [0,1]. \tag{3.2.4}$$

and similarly for the boundary faces at $\xi_2 = 0, 1$ and $\xi_3 = 0, 1$.

We begin by deriving the discrete equations at points on the computational domain whose control volume does not contain a face on the boundary. The finite volume discretisation is obtained by firstly integrating (3.1.1) over each control volume $\Omega_{i,j,k}$, i.e.

$$\int_{\Omega_{i,j,k}} -\nabla\cdot(K\nabla u(\boldsymbol{\xi})) \;+\; \mathbf{a}(\boldsymbol{\xi})\cdot\nabla u(\boldsymbol{\xi}) \;+\; c(\boldsymbol{\xi})u(\boldsymbol{\xi})\; dV \;=\; \int_{\Omega_{i,j,k}} g(\boldsymbol{\xi})\; dV \quad \forall i,j,k \tag{3.2.5}$$

where dV denotes the three dimensional volume element on Ω, i.e.

$$dV \;=\; d\xi_1 d\xi_2 d\xi_3. \tag{3.2.6}$$

Discretisation of the Second-Order Term

Let us first concentrate on finding the discrete equations for the second-order term. We use the divergence theorem to simplify the integrand of equation (3.2.5) by reducing a second order term to a first order term on the boundary of the control volume, which requires the equation to be written in divergence form. Applying the divergence theorem, we obtain

$$-\int_{\Omega_{ijk}} \nabla\cdot(K\nabla u)\; dV \;=\; -\int_{\partial\Omega_{ijk}} K\nabla u\cdot\mathbf{n}\; dS\,,$$

where $\mathbf{n}$ is the outward unit normal vector to $\Omega_{i,j,k}$. Integrating over each cell face of the control volume, we have

$$\begin{aligned}
-\int_{\partial\Omega_{ijk}} K\nabla u\cdot\mathbf{n}\; dS = &-\int_{\Gamma_{i+\frac{1}{2},j,k}} \alpha_1(\boldsymbol{\xi})\frac{\partial u}{\partial\xi_1}\; dS_{i+\frac{1}{2},j,k} + \int_{\Gamma_{i-\frac{1}{2},j,k}} \alpha_1(\boldsymbol{\xi})\frac{\partial u}{\partial\xi_1}\; dS_{i-\frac{1}{2},j,k} \\
&-\int_{\Gamma_{i,j+\frac{1}{2},k}} \alpha_2(\boldsymbol{\xi})\frac{\partial u}{\partial\xi_2}\; dS_{i,j+\frac{1}{2},k} + \int_{\Gamma_{i,j-\frac{1}{2},k}} \alpha_2(\boldsymbol{\xi})\frac{\partial u}{\partial\xi_2}\; dS_{i,j-\frac{1}{2},k} \\
&-\int_{\Gamma_{i,j,k+\frac{1}{2}}} \alpha_3(\boldsymbol{\xi})\frac{\partial u}{\partial\xi_3}\; dS_{i,j,k+\frac{1}{2}} + \int_{\Gamma_{i,j,k-\frac{1}{2}}} \alpha_3(\boldsymbol{\xi})\frac{\partial u}{\partial\xi_3}\; dS_{i,j,k-\frac{1}{2}}\,,
\end{aligned} \tag{3.2.7}$$

where the individual surface elements on Γ are:

$$dS_{i\pm\frac{1}{2},j,k} = \; d\xi_2 d\xi_3, \quad dS_{i,j\pm\frac{1}{2},k} = \; d\xi_1 d\xi_3, \quad dS_{i,j,k\pm\frac{1}{2}} = d\xi_1 d\xi_2 \, .$$

We approximate the derivative across the cell faces using central differences, and then each of the line integrals are approximated using the midpoint rule which gives:

$$\begin{aligned} -\int_{\partial\Omega_{ijk}} K\nabla u \cdot \mathbf{n} \, dS \approx & -\alpha_1(\boldsymbol{\xi}_{i+\frac{1}{2},j,k})\frac{U_{i+1,j,k}-U_{i,j,k}}{h_1}h_2h_3 + \alpha_1(\boldsymbol{\xi}_{i-\frac{1}{2},j,k})\frac{U_{i,j,k}-U_{i-1,j,k}}{h_1}h_2h_3 \\ & - \alpha_2(\boldsymbol{\xi}_{i,j+\frac{1}{2},k})\frac{U_{i,j+1,k}-U_{i,j,k}}{h_2}h_3h_1 + \alpha_2(\boldsymbol{\xi}_{i,j-\frac{1}{2},k})\frac{U_{i,j,k}-U_{i,j-1,k}}{h_2}h_3h_1 \\ & - \alpha_3(\boldsymbol{\xi}_{i,j,k+\frac{1}{2}})\frac{U_{i,j,k+1}-U_{i,j,k}}{h_3}h_1h_2 + \alpha_3(\boldsymbol{\xi}_{i,j,k-\frac{1}{2}})\frac{U_{i,j,k}-U_{i,j,k-1}}{h_3}h_1h_2 \, , \end{aligned}$$

where $\boldsymbol{\xi}_{i,j,k} = (\xi_{1,i}, \xi_{2,j}, \xi_{3,k})$ and $U_{i,j,k}$ is the discrete approximation to the solution u at $\boldsymbol{\xi}_{i,j,k}$. Discrete approximations to derivatives at grid points are often written in *stencil* notation, which is a representation of the non-zero entries of a row of the matrix corresponding to a particular node on the grid. The finite volume discretisation of the second order term in (3.2.5) is a 7-point stencil (i.e. non-zero only at entries of the matrix corresponding to the node itself and for its immediate neighbours) for each node on the grid whose control cell does not contain a face on the boundary. The 7-point stencil for node (i, j, k) is:

$$-\alpha_1(\boldsymbol{\xi}_{i-\frac{1}{2},j,k})H_1 \left[\begin{array}{ccc} & -\alpha_2(\boldsymbol{\xi}_{i,j+\frac{1}{2},k})H_2 & \\ -\alpha_3(\boldsymbol{\xi}_{i,j,k-\frac{1}{2}})H_3 & -\sum & -\alpha_3(\boldsymbol{\xi}_{i,j,k+\frac{1}{2}})H_3 \\ & -\alpha_2(\boldsymbol{\xi}_{i,j-\frac{1}{2},k})H_2 & \end{array} \right] - \alpha_1(\boldsymbol{\xi}_{i+\frac{1}{2},j,k})H_1 \, , \tag{3.2.8}$$

with $\sum$ denoting the sum of all the off-diagonal entries, and where $H_1 = \frac{h_2h_3}{h_1}$, $H_2 = \frac{h_1h_3}{h_2}$ and $H_3 = \frac{h_1h_2}{h_3}$. The numbers in the square brackets give the 5-point stencil in the ξ_2–ξ_3 plane in the usual way (see, for example [20]). The numbers outside the brackets denote the entries corresponding to the upwards and downwards neighbours. Note that we have used a similar notation as in [7] to present the 7-point stencil. Note also that this stencil assumes a uniform grid is being used in each coordinate direction. The following generalizations are made to the stencil to account for non-uniform grids.

$$-\alpha_1(\boldsymbol{\xi}_{i-\frac{1}{2},j,k})H_1^- \left[\begin{array}{ccc} & -\alpha_2(\boldsymbol{\xi}_{i,j+\frac{1}{2},k})H_2^+ & \\ -\alpha_3(\boldsymbol{\xi}_{i,j,k-\frac{1}{2}})H_3^- & -\sum & -\alpha_3(\boldsymbol{\xi}_{i,j,k+\frac{1}{2}})H_3^+ \\ & -\alpha_2(\boldsymbol{\xi}_{i,j-\frac{1}{2},k})H_2^- & \end{array} \right] - \alpha_1(\boldsymbol{\xi}_{i+\frac{1}{2},j,k})H_1^+ \, . \tag{3.2.9}$$

where $H_1^{\pm} = \frac{h_{3,k}h_{2,j}}{h_{1,i}^{\pm}}$, $H_2^{\pm} = \frac{h_{3,k}h_{1,i}}{h_{2,j}^{\pm}}$ and $H_3^{\pm} = \frac{h_{2,j}h_{1,i}}{h_{3,k}^{\pm}}$, $h_{1,i}, h_{2,j}$ and $h_{3,k}$ are the edge lengths of control volume $\Omega_{i,j,k}$, and h^+ and h^- define the distances between adjacent cell centres, e.g.

$$h_{1,i}^{+} = \frac{(h_{1,i} + h_{1,i+1})}{2}\,, \quad h_{1,i}^{-} = \frac{(h_{1,i} + h_{1,i-1})}{2}\,.$$

Discretisation of the First-Order Term

We now discretise the first-order term from (3.2.5). By integrating over the control volume, we have

$$\int_{\Omega_{i,j,k}} \mathbf{a}(\boldsymbol{\xi}) \cdot \nabla u(\boldsymbol{\xi})\; dV\,,$$

where dV is defined in (3.2.6). We use the average of forward and backward differences to approximate the first order derivatives across each cell, i.e.

$$\left.\frac{\partial u}{\partial \xi_1}\right|_i \approx \frac{1}{2}\left\{\frac{U_{i+1} - U_i}{h_{1,i}^{+}} + \frac{U_i - U_{i-1}}{h_{1,i}^{-}}\right\}. \tag{3.2.10}$$

Using (3.2.10) and the midpoint rule to approximate the volume integral, we obtain

$$\begin{aligned}
&\int_{\Omega_{i,j,k}} \mathbf{a}(\boldsymbol{\xi}) \cdot \nabla u(\boldsymbol{\xi})\; dV \;=\; \int_{\Omega_{i,j,k}} a_1(\boldsymbol{\xi})\frac{\partial u}{\partial \xi_1} + a_2(\boldsymbol{\xi})\frac{\partial u}{\partial \xi_2} + a_3(\boldsymbol{\xi})\frac{\partial u}{\partial \xi_3}\; dV \\
&\approx \int_{\Omega_{i,j,k}} a_1(\boldsymbol{\xi})\frac{U_{i+1,j,k} - U_{i-1,j,k}}{2h_1} + a_2(\boldsymbol{\xi})\frac{U_{i,j+1,k} - U_{i,j-1,k}}{2h_2} + a_3(\boldsymbol{\xi})\frac{U_{i,j,k+1} - U_{i,j,k-1}}{2h_3}\; d\xi_1\; d\xi_2\; d\xi_3 \\
&= a_1(\boldsymbol{\xi}_{i,j,k})(U_{i+1,j,k} - U_{i-1,j,k})\frac{h_2h_3}{2} + a_2(\boldsymbol{\xi}_{i,j,k})(U_{i,j+1,k} - U_{i,j-1,k})\frac{h_1h_3}{2} \\
&\qquad\qquad + a_3(\boldsymbol{\xi}_{i,j,k})(U_{i,j,k+1} - U_{i,j,k-1})\frac{h_1h_2}{2}\,,
\end{aligned}$$

if a uniform mesh is used. In this case the 7-point stencil for the first-order term is

$$-a_1(\boldsymbol{\xi}_{i,j,k})\frac{h_2h_3}{2}\begin{bmatrix} & +a_2(\boldsymbol{\xi}_{i,j,k})\frac{h_3h_1}{2} & \\ -a_3(\boldsymbol{\xi}_{i,j,k})\frac{h_1h_2}{2} & 0 & +a_3(\boldsymbol{\xi}_{i,j,k})\frac{h_1h_2}{2} \\ & -a_2(\boldsymbol{\xi}_{i,j,k})\frac{h_3h_1}{2} & \end{bmatrix} + a_1(\boldsymbol{\xi}_{i,j,k})\frac{h_2h_3}{2}\,,$$

or more generally, without the assumption of a uniform mesh,

$$-a_1(\boldsymbol{\xi}_{i,j,k})\frac{V_{i,j,k}}{2h_{1,i}^{-}}\begin{bmatrix} & +a_2(\boldsymbol{\xi}_{i,j,k})\frac{V_{i,j,k}}{2h_{2,j}^{+}} & \\ -a_3(\boldsymbol{\xi}_{i,j,k})\frac{V_{i,j,k}}{2h_{3,k}^{-}} & -\sum & +a_3(\boldsymbol{\xi}_{i,j,k})\frac{V_{i,j,k}}{2h_{3,k}^{+}} \\ & -a_2(\boldsymbol{\xi}_{i,j,k})\frac{V_{i,j,k}}{2h_{2,j}^{-}} & \end{bmatrix} + a_1(\boldsymbol{\xi}_{i,j,k})\frac{V_{i,j,k}}{2h_{1,i}^{+}}\,.$$

where $V_{i,j,k} = h_{1,i}h_{2,j}h_{3,k}$. Note that the matrix resulting from the discretisation of the first order term does is non-symmetric (or skew-symmetric for a regular grid).

Discretisation of the Zeroth-Order Term and Right-Hand-Side

Lastly we discretise the zeroth order term and the right-hand-side. Approximating the integral of the zeroth order term over the control volume using again the midpoint rule we get

$$\int_{\Omega_{i,j,k}} c(\boldsymbol{\xi})u(\boldsymbol{\xi})\ dV \ \approx\ c(\boldsymbol{\xi}_{i,j,k})u(\boldsymbol{\xi}_{i,j,k})h_{1,i}h_{2,j}h_{3,k}\,.$$

Similarly the integration of the right hand side yields

$$\int_{\Omega_{i,j,k}} g(\boldsymbol{\xi})\ dV \ \approx\ g(\boldsymbol{\xi}_{i,j,k})h_{1,i}h_{2,j}h_{3,k}\,.$$

3.2.2 Discretisation of the Terms on the Boundary

Dirichlet Boundary Conditions

Suppose we have homogeneous Dirichlet boundary conditions (3.2.1) on the boundary Γ. Let us without loss of generality (w.l.o.g.) consider the nodes whose cell contains a face on the boundary $\xi_1 = 0$, e.g. nodes at which $i = 1$. Recall the following component from equation (3.2.7):

$$\int_{\Gamma_{i-\frac{1}{2},j,k}} \alpha_1(\boldsymbol{\xi})\frac{\partial u}{\partial \xi_1}\ dS_{i-\frac{1}{2},j,k}\,. \tag{3.2.11}$$

We now use one-sided differences to approximate $\frac{\partial u}{\partial \xi_1}$, rather than central differences, in order to use the value of the solution at the boundary $\xi_1 = 0$. The exact solution is known at $\xi_1 = 0$, namely $U_{\frac{1}{2},j,k} = u(0, \xi_2, \xi_3) = 0$, and the distance between the two points is $h_{1,1}/2$ instead of $h^-_{1,1}$ so (3.2.11) becomes

$$\alpha_1(\xi_{\frac{1}{2},j,k})\frac{U_{1,j,k} - \mathbf{0}}{\frac{h_{1,1}}{2}}h_2h_3 \ =\ \alpha_1(\xi_{\frac{1}{2},j,k})\frac{2U_{1,j,k}}{h_{1,1}}h_2h_3\,.$$

The stencil for the nodes at which $i = 1$, without the assumption of a uniform mesh, becomes a six-point stencil:

$$\begin{bmatrix} & -\alpha_2(\boldsymbol{\xi}_{1,j+\frac{1}{2},k})\frac{h_{3,k}h_{1,1}}{h^+_{2,j}} & \\ -\alpha_3(\boldsymbol{\xi}_{1,j,k-\frac{1}{2}})\frac{h_{2,j}h_{1,1}}{h^-_{3,k}} & -\sum +2\alpha_1(\boldsymbol{\xi}_{\frac{1}{2},j,k})\frac{h_{3,k}h_{2,j}}{h_{1,1}} & -\alpha_3(\boldsymbol{\xi}_{1,j,k+\frac{1}{2}})\frac{h_{2,j}h_{1,1}}{h^+_{3,k}} \\ & -\alpha_2(\boldsymbol{\xi}_{1,j-\frac{1}{2},k})\frac{h_{3,k}h_{1,1}}{h^-_{2,j}} & \end{bmatrix} - \alpha_1(\boldsymbol{\xi}_{\frac{3}{2},j,k})\frac{h_{3,k}h_{2,j}}{h^+_{1,1}}.$$

Analogously, we can obtain similar stencils at $i = n_1$, $j = 1$, $j = n_2$, $k = 1$ and $k = n_3$, i.e. at the edges and corners of the domain.

Neumann Boundary Conditions

Now suppose we want to impose homogeneous Neumann boundary conditions (3.2.2) on Γ, and consider the boundary face where $\xi_1 = 0$ (i.e. $i = 1$). Recall again the component (3.2.11) from (3.2.7). By boundary condition (3.2.2), $\frac{\partial u}{\partial \xi_1}|_{\xi_1=0} = 0$, and so the component (3.2.11) vanishes and we are left with the following stencil:

$$\begin{bmatrix} & -\alpha_2(\boldsymbol{\xi}_{1,j+\frac{1}{2},k})\frac{h_{3,k}h_{1,1}}{h^+_{2,j}} & \\ -\alpha_3(\boldsymbol{\xi}_{1,j,k-\frac{1}{2}})\frac{h_{2,j}h_{1,1}}{h^-_{3,k}} & -\sum & -\alpha_3(\boldsymbol{\xi}_{1,j,k+\frac{1}{2}})\frac{h_{2,j}h_{1,1}}{h^+_{3,k}} \\ & -\alpha_2(\boldsymbol{\xi}_{1,j-\frac{1}{2},k})\frac{h_{3,k}h_{1,1}}{h^-_{2,j}} & \end{bmatrix} - \alpha_1(\boldsymbol{\xi}_{\frac{3}{2},j,k})\frac{h_{3,k}h_{2,j}}{h^+_{1,1}}.$$

Similar stencils are obtained analogously at the remaining edges and corners of Ω.

Periodic Boundary Conditions

Finally, suppose periodic boundary conditions (3.2.3) – (3.2.4) are imposed on Γ, and consider the boundary faces where $\xi_1 = 0$ and $\xi_1 = 1$ (i.e. $i = 1$ and $i = n_1$). We recall again the component (3.2.11) from (3.2.7) as before, and consider the nodes whose cell contains a face on the boundary $\xi_1 = 0$. By (3.2.3), $U_{\frac{1}{2},j,k} = U_{n_1+\frac{1}{2},j,k}$ so (3.2.11) becomes

$$\alpha_1(\xi_{\frac{1}{2},j,k})\frac{U_{1,j,k} - U_{n_1,j,k}}{h^-_{1,1}}h_2h_3.$$

where $h^-_{1,1} = 0.5\ (h_{1,1} + h_{1,n_1})$, and so the seven-point stencil is

$$-\alpha_1(\boldsymbol{\xi}_{\frac{1}{2},j,k})\frac{h_{3,k}h_{2,j}}{h^-_{1,1}}\begin{bmatrix} & -\alpha_2(\boldsymbol{\xi}_{1,j+\frac{1}{2},k})\frac{h_{3,k}h_{1,1}}{h^+_{2,j}} & \\ -\alpha_3(\boldsymbol{\xi}_{1,j,k-\frac{1}{2}})\frac{h_{2,j}h_{1,1}}{h^-_{3,k}} & -\sum & -\alpha_3(\boldsymbol{\xi}_{1,j,k+\frac{1}{2}})\frac{h_{2,j}h_{1,1}}{h^+_{3,k}} \\ & -\alpha_2(\boldsymbol{\xi}_{1,j-\frac{1}{2},k})\frac{h_{3,k}h_{1,1}}{h^-_{2,j}} & \end{bmatrix} - \alpha_1(\boldsymbol{\xi}_{\frac{3}{2},j,k})\frac{h_{3,k}h_{2,j}}{h^+_{1,1}}.$$

Once again, we obtain similar stencils at each of the edges and corners of Ω.

Resulting System of Linear Equations

The discretisation results in a system of linear equations of the form

$$A\mathbf{u} = \mathbf{b},$$

where $A \in \mathbb{R}^{n\times n}$, and $n = n_\lambda \times n_\phi \times n_r$ is the dimension of the problem. Matrix A represents the discretisation of the zeroth, first and second order terms. It is a sparse and symmetric positive definite (SPD) matrix if the coefficient $\mathbf{a}$ to the first

order term in (3.1.1) is zero (otherwise it is non-symmetric) and if Dirichlet boundary conditions are imposed (otherwise it is positive semi-definite). $\mathbf{u} \in \mathbb{R}^n$ is the unknown solution vector corresponding to the values of the unknown function u at the cell centres, and $\mathbf{b} \in \mathbb{R}^n$ is the right-hand side with entries $g(\boldsymbol{\xi}_{i,j,k})h_{1,i}h_{2,j}h_{3,k}$. A lexicographical ordering of the unknowns is used.

3.2.3 Special Case – Spherical Polar Coordinates

Here we consider the discretisation of (3.1.1) for the particular case of spherical polar coordinates, with $\boldsymbol{\xi} = (r, \phi, \lambda) \in (0,1)^3$, where r, ϕ and λ are the nondimensionalized radial direction, polar angle and azimuthal angle respectively. This special case is of a particular interest in numerical weather prediction, as the elliptic problems of interest from Chapter 2 are all given in spherical polar coordinates. Recall that the problem needs to be scaled by $(a + dr)^2 \sin(\pi\phi)$ so that it can be written in the general tensor product form.

The Poisson equation on a sphere is given in (3.1.2), but in general, elliptic problems on a sphere may have additional coefficients, which is certainly the case for the problems from Chapter 2. These can be written generally as

$$\begin{aligned}
\alpha_1(r,\phi,\lambda) &= L_r(\lambda,\phi)(a+dr)^2\sin(\pi\phi)/d^2\,,\\
\alpha_2(r,\phi,\lambda) &= L_\phi(r,\lambda)\sin(\pi\phi)/\pi^2\,,\\
\alpha_3(r,\phi,\lambda) &= L_\lambda(r,\phi)/(4\pi^2\sin(\pi\phi))\,,
\end{aligned}$$

where the coefficients L_r, L_λ and L_ϕ have to again be assumed to be separable, e.g. $L_r(\phi,\lambda) = L_r^\lambda(\lambda)L_r^\phi(\phi)$. For simplicity we assume that the i^{th} coefficient L_i does not depend on the i^{th} variable, which is always the case for the problems in Chapter 2. In general we also have $c \neq 0$ and $\mathbf{a} \neq \mathbf{0}$, but for the problems in Chapter 2, these are restricted to $c \geq 0$ and $\mathbf{a}(\boldsymbol{\xi}) = \big[a_1(r)(a+dr)^2\sin(\pi\phi)/d, 0, 0\big]$.

The main difference of this setting is the boundary conditions in the ϕ-direction coming from the pole, which also affects the mesh defined on Ω at these boundaries. The mesh is defined by subdividing Ω into $n_\lambda \times n_\phi \times n_r$ cubes with cell centres

$$\{\,(r_i,\phi_j,\lambda_k)\;:\; i=1,\ldots,n_r,\; j=1,\ldots,n_\phi,\; k=1,\ldots,n_\lambda\,\},$$

and edge lengths $h_\lambda = 1/n_\lambda$, $h_\phi = 1/(n_\phi+1)$ and $h_{r,i}$, $i = 1,\ldots,n_r$. In addition there are $2 \times n_r$ cells at the poles with edge lengths 1, $h_\phi/2$ and $h_{r,i}$. The computational grid in the $\lambda - \phi$ plane can be seen in Figure 3-1(a), where the top and bottom boundary represent the North and South pole, respectively. The grid on the $\lambda - \phi$ plane is uniform,

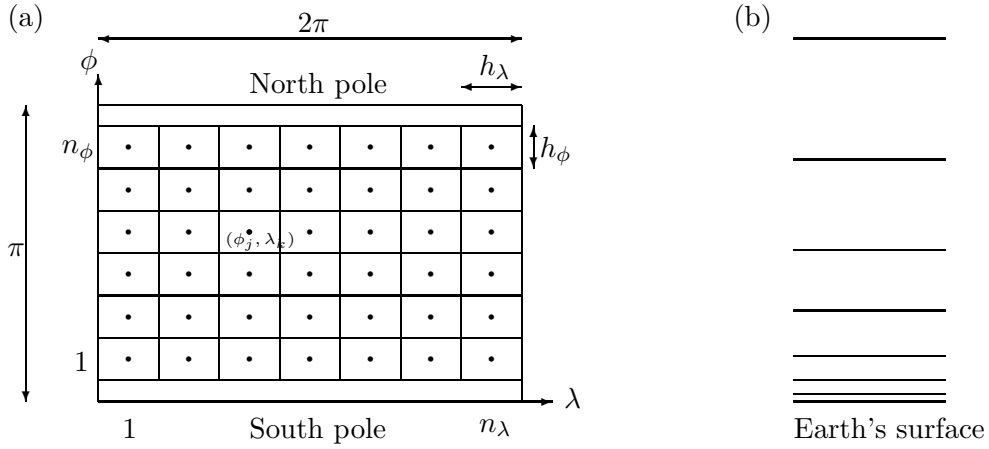

Figure 3-1: (a) The computational grid in the λ-ϕ plane, and (b) the graded mesh in the r-direction.

except at the poles. The nodes are located at the cell centres, where $\lambda_k = (k - \frac{1}{2})h_\lambda$ and $\phi_j = jh_\phi$. At the poles we use half cells, so that the poles themselves are located at the centres of the cells in the physical domain Ω, and so that a discrete equation can be derived at these points in the same fashion as at the other points. In the radial direction, the mesh is graded as shown in Figure 3-1(b) with the cell centres located at $r_i = \sum_{t=1}^{i-1} h_{r,t} + \frac{1}{2}h_{r,i}$, and with the mesh widths $h_{r,i}$ increasing with i. Thus, the total number of unknowns, including the unknowns at the poles, is $(n_\lambda \times n_\phi + 2) \times n_r$. This computational grid is the same as the Arakawa C-grid and the Charney-Phillips grid introduced in Chapter 2, where the cell centred nodes are located at the p'-points and ρ-levels.

As described in Section 3.2.1, we discretise (3.1.1) using the cell centred finite volume method, where firstly the PDE is integrated over each mesh cell (or control volume) corresponding to an interior grid point (r_i, ϕ_j, λ_k), i.e. $\Omega_{i,j,k}$, or corresponding to a grid point at the pole. Denoting the control volume corresponding to a vertical grid point at level i on the north and south poles as Ω_i^S and Ω_i^N respectively, we have

$$\Omega_i^S = \left[r_{i-\frac{1}{2}}, r_{i+\frac{1}{2}}\right] \times \left[0, \phi_{\frac{1}{2}}\right] \times [0,1] \ , \text{ and}$$
$$\Omega_i^N = \left[r_{i-\frac{1}{2}}, r_{i+\frac{1}{2}}\right] \times \left[0, \phi_{n_\phi+\frac{1}{2}}\right] \times [0,1] \ .$$

where $\phi_{\frac{1}{2}} = h_\phi/2$. Except at the poles, the boundary of each control volume, $\Gamma_{i,j,k}$, consists of six faces, as described in Section 3.2.1. Each control volume at the poles,

however, has $n_\lambda + 2$ faces, i.e.

$$\Gamma_i^S \;=\; \bigcup_{k=1}^{n_\lambda} \Gamma_{i,\frac{1}{2},k} \cup \Gamma^S_{i-\frac{1}{2}} \cup \Gamma^S_{i+\frac{1}{2}}\,,$$

where $\Gamma^S_{i\pm\frac{1}{2}} \;=\; \{r_{i\pm\frac{1}{2}}\} \times \left[0, \phi_{\frac{1}{2}}\right] \times [0,1]$ are the top and bottom faces of the control volume at the south pole on vertical grid level i.

Since the problem is discretised on a sphere, it is necessary to impose periodic boundary conditions on the lateral boundary, i.e.

$$\begin{aligned} u(r,\phi,0) &= u(r,\phi,1)\,, \\ \frac{\partial u}{\partial \lambda}(r,\phi,0) &= \frac{\partial u}{\partial \lambda}(r,\phi,1) \qquad \forall \phi \in [0,1], \forall r \in [0,1]. \end{aligned}$$

In addition, for the upper and lower boundaries of the atmosphere (corresponding to $r = 0$ and $r = 1$), the problems from Chapter 2 either have homogeneous Dirichlet boundary conditions, i.e.

$$u(0,\phi,\lambda) = u(1,\phi,\lambda) = 0.$$

or homogeneous Neumann boundary conditions, i.e.

$$\frac{\partial u}{\partial r}(0,\phi,\lambda) = \frac{\partial u}{\partial r}(1,\phi,\lambda) = 0.$$

The discretisation at the interior nodes is identical to the general case, resulting in the same stencil (3.2.9). Recall that for spherical polar coordinates we have

$$\begin{aligned} \alpha_1(\boldsymbol{\xi}_{i\pm\frac{1}{2},j,k}) &= \alpha_1(r_{i\pm\frac{1}{2}}, \phi_j, \lambda_k) = L_r(\phi_j,\lambda_k)(a + dr_{i+\frac{1}{2}})^2 \sin(\pi\phi_j)/d^2\,, \\ \alpha_2(\boldsymbol{\xi}_{i,j\pm\frac{1}{2},k}) &= \alpha_2(r_i, \phi_{j\pm\frac{1}{2}}, \lambda_k) = L_\lambda(r_i,\lambda_k)\sin(\pi\phi_j)/\pi^2\,, \\ \alpha_3(\boldsymbol{\xi}_{i,j,k\pm\frac{1}{2}}) &= \alpha_3(r_i, \phi_j, \lambda_{k\pm\frac{1}{2}}) = L_\phi(r_i,\phi_j)/(4\pi^2\sin(\pi\phi_j))\,, \end{aligned}$$

and $h_1 = h_r$, $h_2 = h_\lambda$, $h_3 = h_\phi$.

We now come to the discretisation at the boundaries. The homogeneous Dirichlet and Neumann boundary conditions at $r = 0$ and $r = 1$ have been covered in the general case, as well as the periodic boundary conditions on the lateral boundary. What remains is the treatment of the north-south boundaries at the polar regions.

Poles

We now tackle the discretisation at the poles, which occupy an entire $r - \lambda$ plane. Each pole cell (apart from those at the top or bottom boundary) has an entire λ-line of

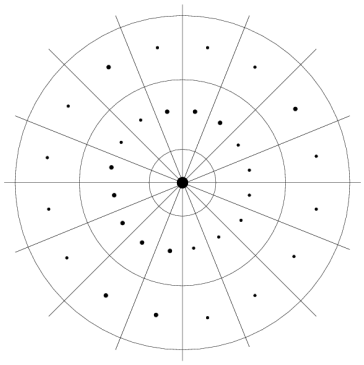

Figure 3-2: The polar region, with the pole at the centre of n_λ half cells

neighbours in addition to upper and lower neighbours, thus a total of $n_\lambda+2$ neighbours, leading to a $(n_\lambda+3)$-point stencil at these cells. The poles nodes on level i are located at the centre of the control volume Ω_i^S or Ω_i^N which is comprised of n_λ half cells (see Figure 3-2), with $\Gamma^S_{i\pm\frac{1}{2}}$ or $\Gamma^N_{i\pm\frac{1}{2}}$ corresponding to the top and bottom faces of the control volume. For the south pole (where $\phi = 0$), the integration over control volume Ω_i^S leads to the following discretisation:

$$
\begin{aligned}
&-\int_{\Gamma_i^S} K\nabla u\cdot\mathbf{n}\; dS \\
&= -\int_{\Gamma^S_{i+\frac{1}{2}}} \alpha_1(\boldsymbol{\xi})\frac{\partial u}{\partial r}\; dS^S_{i+\frac{1}{2}} \;-\; \int_{\Gamma^S_{i-\frac{1}{2}}} \alpha_1(\boldsymbol{\xi})\frac{\partial u}{\partial r}\; dS^S_{i-\frac{1}{2}} \;+\; \sum_{k=1}^{n_\lambda}\int_{\Gamma_{i,\frac{1}{2},k}} \alpha_2(\boldsymbol{\xi})\frac{\partial u}{\partial \phi}\; dS_{i,\frac{1}{2},k} \\
&= -\,\alpha_1(r_{i+\frac{1}{2}},\phi_{\frac{1}{4}},\lambda_k)\frac{U^S_{i+1}-U^S_i}{h^+_{r,i}}(1)\frac{h_\phi}{2} \;-\; \alpha_1(r_{i-\frac{1}{2}},\phi_{\frac{1}{4}},\lambda_k)\frac{U^S_i-U^S_{i-1}}{h^-_{r,i}}(1)\frac{h_\phi}{2} \\
&\qquad\qquad -\sum_{k=1}^{n_\lambda}\alpha_2(r_i,\phi_{\frac{1}{2}},\lambda_k)\frac{U_{i,1,k}-U^S_i}{h_\phi}h_\lambda h_{r,i} \\
&= -\,\alpha_1(r_{i+\frac{1}{2}},\phi_{\frac{1}{4}},\lambda_k)\frac{h_\phi}{2h^+_{r,i}}\left(U^S_{i+1}-U^S_i\right) \;-\; \alpha_1(r_{i-\frac{1}{2}},\phi_{\frac{1}{4}},\lambda_k)\frac{h_\phi}{2h^-_{r,i}}\left(U^S_i-U^S_{i-1}\right) \\
&\qquad\qquad -\;\frac{h_{r,i}h_\lambda}{h_\phi}\sum_{k=1}^{n_\lambda}\alpha_2(r_i,\phi_{\frac{1}{2}},\lambda_k)\left(U_{i,1,k}-U^S_i\right)\;,
\end{aligned}
$$

where U_i^S is the discrete solution at the south pole on level i and $\phi_{\frac{1}{4}} = h_\phi/4$. For the right hand side at the south pole, again we discretise by approximating the integral using the midpoint rule:

$$\int_{\Omega_i^S} g(a+dr)^2 \sin(\pi\phi)\; dV \;=\; g_i^S (a+dr_i)^2 \sin(\pi\phi_{\frac{1}{4}}) h_{r,i} \frac{h_\phi}{2}\,,$$

where $h_{r,i} \times \frac{h_\phi}{2}$ is an approximation to the volume of the control cell Ω_i^S, and we see from Figure 3-1(a) that $(r_i, \phi_{\frac{1}{4}}, 0.5)$ is the midpoint of Ω_i^S in the computational grid. Note that the midpoint of Ω_i^S in the physical domain is actually located at the south pole (see Figure 3-2), but the special treatment we use at the poles is consistent with the work of Barros [7] and also with the discretisation used at the Met Office.

Analogously for the north pole (where $\phi = \pi$) we obtain

$$\begin{aligned} -\;\alpha_1(r_{i+\frac{1}{2}}, \phi_{n_\phi+\frac{3}{4}}, \lambda_k) \frac{h_\phi}{2h_{r,i}^+} \left(U_{i+1}^N - U_i^N\right) \;-\; \alpha_1(r_{i-\frac{1}{2}}, \phi_{n_\phi+\frac{3}{4}}, \lambda_k) \frac{h_\phi}{2h_{r,i}^-} \left(U_i^N - U_{i-1}^N\right) \\ -\;\frac{h_{r,i} h_\lambda}{h_\phi} \sum_{k=1}^{n_\lambda} \alpha_2(r_i, \phi_{n_\phi+\frac{1}{2}}, \lambda_k) \left(U_{i,n_\phi,k} - U_i^N\right) \\ =\; g_i^N (a+dr_i)^2 \sin(\pi\phi_{n_\phi+\frac{3}{4}}) h_{r,i} \frac{h_\phi}{2}\,. \end{aligned}$$

Resulting System of Linear Equations

The discretisation results in a system of linear equations of the form

$$A\mathbf{u} = \mathbf{b}.$$

$A \in \mathbb{R}^{n\times n}$, and $n = (n_\lambda \times n_\phi + 2) \times n_r$ is the dimension of the problem. $\mathbf{u} \in \mathbb{R}^n$ is the unknown solution vector corresponding to the values of the unknown function u at the cell centres, and $\mathbf{b} \in \mathbb{R}^n$ is the right-hand side containing the source terms. It contains seven non-zero entries per row corresponding to an interior node, and $n_\lambda + 3$ non-zeros per row corresponding to pole nodes that are not at the upper or lower boundaries. For typical problem sizes used at the Met Office, $n_\lambda + 3 \gg 7$, hence the rows corresponding to the pole nodes are significantly more dense. For nodes whose cell face is on the upper or lower boundary, there are six non-zeros (for non-pole nodes) or $n_\lambda + 2$ non-zeros per row (for pole nodes). Figure 3-3 is a spy plot showing the sparsity pattern of matrix A with problem size $n = 152$ ($n_\lambda = 6,\ n_\phi = 6,\ n_r = 4$). Periodic boundary conditions are imposed at the lateral boundary, in addition to the polar boundary at the north-south boundaries. If the vertical boundary conditions are of Neumann-type, then these boundary conditions yield a singular system of linear equations, meaning that the solution to this system is unique only up to a constant. Techniques for dealing with this singularity will be discussed in Section 5.5.2.

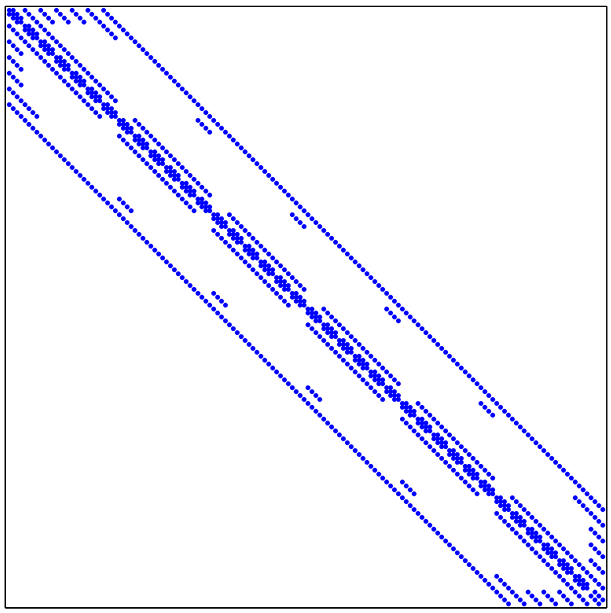

Figure 3-3: A spy plot showing the sparsity pattern of matrix A for the 3D problem.

3.2.4 Two Dimensional Problems on the Surface of the Sphere

In NWP, the solution to Poisson-type equations on the sphere in two dimensions are also highly important. One such use of a two dimensional Poisson solver is for the balanced and unbalanced equations (2.3.7) and (2.3.9), as well as regularly within the control variable transform (CVT) described in Chapter 2. Therefore in this section we derive the finite volume discretisation of the two dimensional Poisson-type equation.

Consider the following two dimensional abstract equation:

$$-\nabla \cdot (K \nabla u(\boldsymbol{\xi})) = g(\boldsymbol{\xi}) \quad \text{on } \Omega_{2D}\,, \tag{3.2.12}$$

with

$$K = K(\boldsymbol{\xi}) = \begin{pmatrix} \alpha_1(\boldsymbol{\xi}) & 0 \\ 0 & \alpha_2(\boldsymbol{\xi}) \end{pmatrix},$$

for $\boldsymbol{\xi} = (\lambda, \phi)$ and for separable functions $\alpha_1(\phi, \lambda) = \alpha_1^1(\phi)\alpha_1^2(\lambda) > 1$ a.e. and $\alpha_2(\phi, \lambda) = \alpha_2^1(\phi)\alpha_2^2(\lambda) > 1$ a.e. Of particular interest is the case of a Poisson-type equation in spherical polar coordinates which is given by $\alpha_1(\phi, \lambda) = L_\phi(\lambda)\sin(\pi\phi)/\pi^2$, $\alpha_2(\phi, \lambda) = L_\lambda(\phi)/(4\pi^2 \sin(\pi\phi))$ and a scaling by $\sin(\pi\phi)$:

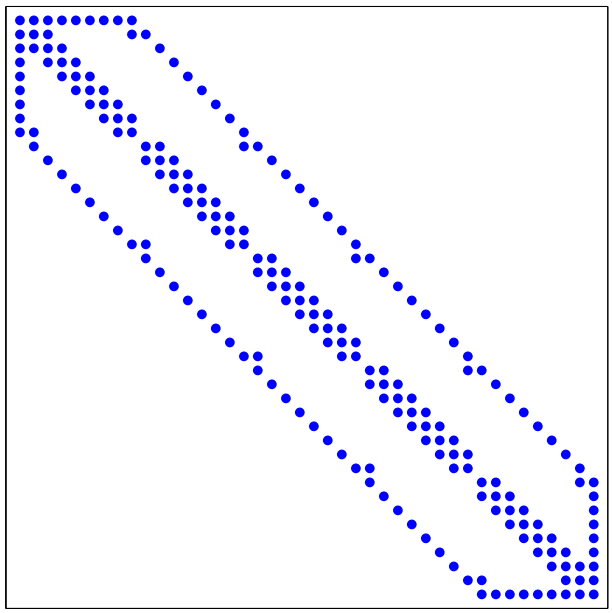

Figure 3-4: A spy plot showing the sparsity pattern of the matrix for the 2D problem.

$$-\frac{\partial}{\partial\phi}\left(\frac{L_\phi(\lambda)\sin(\pi\phi)}{\pi^2}\frac{\partial u}{\partial\phi}\right) - \frac{\partial}{\partial\lambda}\left(\frac{L_\lambda(\phi)}{4\pi^2\sin(\pi\phi)}\frac{\partial u}{\partial\lambda}\right) = g\sin(\pi\phi)\,, \tag{3.2.13}$$

This is solved on the unit square $\Omega_{2D} = (0,1)^2$, which is subdivided into $n_\phi \times n_\lambda$ cells

$$\{\,(\phi_j,\lambda_k)\ :\ j=1,\ldots,n_\phi,\ k=1,\ldots,n_\lambda\,\},$$

in addition to one cell for each pole. This computational grid is pictured in Figure 3-1(a), and the edge lengths h_ϕ and h_λ are defined as in Section 3.2.3. As in Section 3.2.1, (3.2.12) is integrated over each control volume, and using the Divergence theorem it is simplified to a first order term on the boundary of the control volume. Each of the line integrals is approximated by the midpoint rule and the derivatives involved on it by central differences, producing a 5-point stencil at an interior node (j,k):

$$\begin{bmatrix} & -L_\phi(\lambda_k)\frac{h_\lambda}{h_\phi}\frac{\sin(\pi\phi_{j+\frac{1}{2}})}{\pi^2} & \\ -L_\lambda(\phi_j)\frac{h_\phi}{h_\lambda}\frac{1}{4\pi^2\sin(\pi\phi_j)} & -\sum & -L_\lambda(\phi_j)\frac{h_\phi}{h_\lambda}\frac{1}{4\pi^2\sin(\pi\phi_j)} \\ & -L_\phi(\lambda_k)\frac{h_\lambda}{h_\phi}\frac{\sin(\pi\phi_{j-\frac{1}{2}})}{\pi^2} & \end{bmatrix}, \tag{3.2.14}$$

and the sparsity pattern for the matrix is plotted in Figure 3-4.

3.3 Finite elements and a Link to Finite Volumes Using Quadrature

Although the majority of the work and computations in this thesis is based on using the finite volume discretisation, it is also necessary to use the finite element scheme (described in e.g. [19, 23, 67]) to prove some theoretical results (which we will see in Chapter 5). However, as discussed in [14], a finite element discretisation of an elliptic problem will cover most finite volume discretisations, since the two schemes can be shown to agree if suitable quadrature formulas are used. Thus, with slight modifications, the theoretical results in Chapter 5 will carry over to a finite volume discretisation of the same problem. In Section 3.3.1 the discretisation of the two-dimensional problem (3.2.12) using bilinear finite elements is described, and in Section 3.3.2 a quadrature rule is devised which links the finite volume and finite element schemes. Finally in Section 3.3.3 we discuss on how the ideas are extended to three dimensions.

3.3.1 Piecewise Bilinear Finite Elements in Two Dimensions

Consider an abstract two-dimensional problem:

$$-\nabla \cdot (K(\boldsymbol{\xi})\nabla u(\boldsymbol{\xi})) = g(\boldsymbol{\xi}), \quad \text{on } \Omega = \Omega_x \times \Omega_y \,, \tag{3.3.1}$$

with a continuous boundary Γ, homogeneous Dirichlet boundary conditions $u = 0$ on Γ, and $\boldsymbol{\xi} = (x, y)$. As shown in [67], the finite element method involves writing problem (3.3.1) in a *weak form* and then approximating this weak form by formulating an *approximate weak form* in a finite dimensional space.

To find the weak form of (3.3.1) we introduce a space V of functions in Ω that vanish on the boundary. Here, the appropriate choice for V is the Hilbert space $H_0^1(\Omega)$:

$$H_0^1(\Omega) = \left\{ v : \Omega \to \mathbb{R} \ : \ \int_\Omega (|v|^2 + |\nabla v|^2) < \infty \quad \text{and} \quad v = 0 \text{ on } \Gamma \right\} .$$

If u solves (3.3.1), then it also satisfies

$$-\int_\Omega v \nabla \cdot (K \nabla u)\, dV = \int_\Omega g v \ dV \,, \tag{3.3.2}$$

for any arbitrary function $v \in V$. Now using Green's formula [19, Chapter 0], we obtain

from (3.3.2):

$$\int_\Omega \nabla v \cdot (K\nabla u)\, dV \; - \; \underbrace{\int_\Gamma v\,(K\nabla u)\cdot \mathbf{n} dS}_{=0 \text{ as } v|_\Gamma = 0} \; = \int_\Omega gv \; dV \,.$$

We now write the weak form of (3.3.1) as:
Find $u \in V$ such that

$$a(u,v) = (g,v)_{L^2(\Omega)}, \quad \forall v \in V \tag{3.3.3}$$

where $a : V \times V \to \mathbb{R}$ is a bilinear form defined as

$$a(u,v) = \int_\Omega \nabla v \cdot (K\nabla u)\, dxdy \,,$$

and $(\cdot,\cdot)_{L^2(\Omega)}$ is the scalar product of the function space $L^2(\Omega)$, i.e

$$(g,v)_{L^2(\Omega)} = \int_\Omega gv \; dxdy \,.$$

(Note that $dxdy$ will now be used to denote the 2D volume element instead of dV)
The solution $u \in V$ of the weak form (3.3.3) is then approximated by choosing a finite dimensional space $V_h \subset V$ and by introducing the approximate weak form:
Find $u_h \in V_h$ such that

$$a(u_h, v_h) = (g, v_h)_{L^2(\Omega)}, \quad \forall v_h \in V_h \,. \tag{3.3.4}$$

In finite element methods the finite dimensional space V_h is constructed by decomposing Ω into a *mesh* of triangular or rectangular *elements* whose vertices are the nodes of the mesh. We denote the collection of elements as $\mathcal{T}$ and a typical element including its boundary as $\tau \in \mathcal{T}$. The set of nodes in the mesh, excluding the boundary because of the Dirichlet boundary conditions, is denoted $\mathcal{N}$. We also require a suitable basis for V_h. In the lowest order (bilinear) case on rectangular elements, as used here, the basis functions are (usually) associated with nodes of the grid and have support only in elements containing that node. For any $\mathbf{n}_i \in \mathcal{N}$, denote the basis for V_h as

$$\{\phi_i(x,y) : \mathbf{n}_i \in \mathcal{N}\}.$$

The solution $u_h \in V_h$ to the approximate weak form can be written as a linear combination of the basis functions

$$u_h = \sum_{\mathbf{n}_j \in \mathcal{N}} U_j \phi_j \,,$$

and since a is a bilinear form (linear in both its arguments) and L is also linear, we get

$$\sum_{\mathbf{n}_j \in \mathcal{N}} \underbrace{a(\phi_j, \phi_i)}_{A_{i,j}} U_j = \underbrace{(g, \phi_i)_{L^2(\Omega)}}_{b_i} \quad \forall \mathbf{n}_i \in \mathcal{N},$$

or equivalently

$$A\mathbf{u} = \mathbf{b}, \tag{3.3.5}$$

where A is known as the *stiffness matrix*.

The system (3.3.5) is assembled via *element stiffness matrices*. We have

$$\begin{aligned} A_{i,j} &= \int_\Omega \nabla\phi_i \cdot (K\nabla\phi_j) dxdy \\ &= \sum_{\tau \in \mathcal{T}} \underbrace{\int_\tau \nabla\phi_i \cdot (K\nabla\phi_j) dxdy}_{A^\tau_{i,j}}, \end{aligned} \tag{3.3.6}$$

where $A^\tau_{i,j}$ is the element stiffness matrix for element $\tau \in \mathcal{T}$. Since ϕ_i has support only in elements containing node $\mathbf{n}_i$, $A^\tau_{i,j} = 0$ unless $\mathbf{n}_i$ and $\mathbf{n}_j$ are both nodes of element τ. Hence, $A^\tau_{i,j}$ can be stored as a 4×4 matrix with rows and columns corresponding to the four nodes of τ.

Depending on the entries in the coefficient matrix K, it may be possible to evaluate the integral over τ exactly. However, it is common to use quadrature rules to approximate the integral, and for a suitable quadrature rule, this will lead to the finite volume discretisation of (3.3.1) given above, as we will see in the following section.

3.3.2 Link to the Finite Volume Scheme

Let us now restrict our problem to a uniform rectangular mesh on the unit square, with mesh widths h_x and h_y in the x- and y-directions respectively. We denote the set of nodes of the mesh as

$$\{(x_i, y_j) : \ i = 1, \ldots, n_x, \ j = 1, \ldots, n_y\}.$$

where $x_i = ih_x$ and $y_j = jh_y$. The element centred at $(x_{i+\frac{1}{2}}, y_{j+\frac{1}{2}})$ is denoted $\tau_{i+\frac{1}{2},j+\frac{1}{2}}$ where

$$\tau_{i+\frac{1}{2},j+\frac{1}{2}} = [x_i, x_{i+1}] \times [y_j, y_{j+1}].$$

If the mesh width in the x- and y-directions are equal, then $h_x = h_y = h$. The mesh is visualized in Figure 3-5. We choose V_h to be a space of continuous and piecewise bilinear functions on Ω that vanish on the boundary. We can do this by resorting

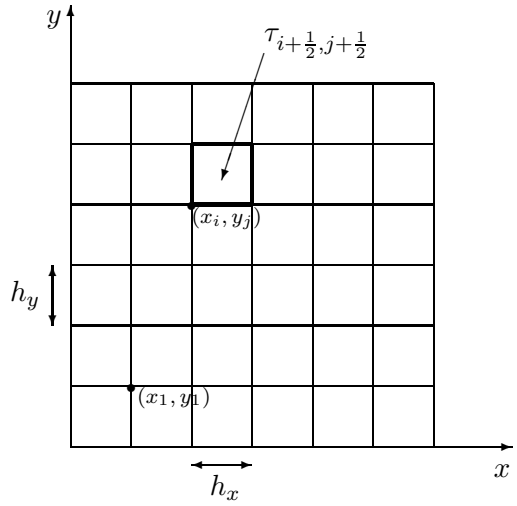

Figure 3-5: Finite element mesh

to $H_0^1(\Omega)$-conforming finite dimensional spaces built in a tensor product fashion, as done in [14]. That is, we decompose Ω_x and Ω_y into two 1D grids and introduce finite dimensional spaces $V_h^x \subset H_0^1(\Omega_x)$ and $V_h^y \subset H_0^1(\Omega_y)$ on these grids. We see from [67] that suitable nodal basis functions for V_h^x and V_h^y are the linear hat functions $\{\phi_i^x : \ i = 1, \ldots, n_x\}$ and $\{\phi_j^y : \ j = 1, \ldots, n_y\}$ respectively, defined by

$$\phi_i^x(x) = \begin{cases} \frac{(x - x_{i-1})}{h_x} & x \in [x_{i-1}, x_i] \\ \frac{(x_{i+1} - x)}{h_x} & x \in [x_i, x_{i+1}] \\ 0 & \text{elsewhere} \end{cases}, \qquad \phi_j^y(y) = \begin{cases} \frac{(y - y_{j-1})}{h_y} & y \in [y_{j-1}, y_j] \\ \frac{(y_{j+1} - y)}{h_y} & y \in [y_j, y_{j+1}] \\ 0 & \text{elsewhere} \end{cases},$$

The $H_0^1(\Omega)$-conforming finite dimensional space V_h can then be defined as $V_h = V_h^x \otimes V_h^y$, i.e. $V_h = span\{u(x)v(y) : u \in V_h^x, \ v \in V_h^y\}$. The nodal basis $\{\phi_{i,j} : \ i = 1, \ldots, n_x, \ j = 1, \ldots, n_y\}$ of V_h is the set of products of any two basis functions of V_h^x and V_h^y, i.e.

$$\phi_{i,j}(x,y) = \begin{cases} \frac{(x - x_{i-1})}{h_x} \frac{(y - y_{j-1})}{h_y} & x \in [x_{i-1}, x_i], \ y \in [y_{j-1}, y_j] \\ \frac{(x - x_{i-1})}{h_x} \frac{(y_{j+1} - y)}{h_y} & x \in [x_{i-1}, x_i], \ y \in [y_j, y_{j+1}] \\ \frac{(x_{i-1} - x)}{h_x} \frac{(y - y_{j-1})}{h_y} & x \in [x_i, x_{i+1}], \ y \in [y_{j-1}, y_j] \\ \frac{(x_{i-1} - x)}{h_x} \frac{(y_{j+1} - y)}{h_y} & x \in [x_i, x_{i+1}], \ y \in [y_j, y_{j+1}] \\ 0 & \text{elsewhere} \end{cases} \tag{3.3.7}$$

We calculate the 4×4 element stiffness matrix for element $\tau_{i+\frac{1}{2},j+\frac{1}{2}}$ by calculating

$$A_{(i,j)(k,l)}^{\tau_{i+\frac{1}{2},j+\frac{1}{2}}} = \int_{\tau_{i+\frac{1}{2},j+\frac{1}{2}}} \nabla\phi_{i,j} \cdot (K \nabla\phi_{k,l}) dx dy \,, \tag{3.3.8}$$

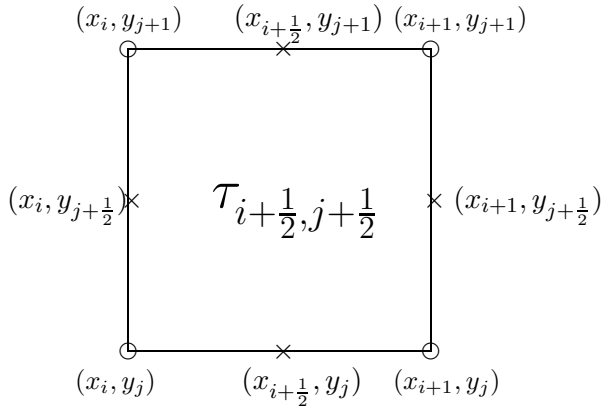

Figure 3-6: Element $\tau_{i+\frac{1}{2},j+\frac{1}{2}}$

where (x_i, y_j) and (x_k, y_l) are two nodes of the mesh. (3.3.8) expands as

$$A_{(i,j)(k,l)}^{\tau_{i+\frac{1}{2},j+\frac{1}{2}}} = \int_{\tau_{i+\frac{1}{2},j+\frac{1}{2}}} \alpha_1(x,y)\frac{\partial\phi_{i,j}}{\partial x}\frac{\partial\phi_{k,l}}{\partial x}dxdy \; + \; \int_{\tau_{i+\frac{1}{2},j+\frac{1}{2}}} \alpha_2(x,y)\frac{\partial\phi_{i,j}}{\partial y}\frac{\partial\phi_{k,l}}{\partial y}dxdy\,. \tag{3.3.9}$$

The integrals in (3.3.9) are approximated as usual by quadrature rules. However, to find a scheme that resembles the finite volume discretisation, the two integrals have to be approximated by different quadrature rules. For the first integral in (3.3.9) we use the midpoint rule in x and the trapezoidal rule in y, i.e.

$$\int_{\tau_{i+\frac{1}{2},j+\frac{1}{2}}} f(x,y)dxdy \approx \frac{h_x h_y}{2}\left(f(x_{i+\frac{1}{2}}, y_j) + f(x_{i+\frac{1}{2}}, y_{j+1})\right)\,. \tag{3.3.10}$$

For the second integral, we use the midpoint rule in y and the trapezoidal rule in x, i.e.

$$\int_{\tau_{i+\frac{1}{2},j+\frac{1}{2}}} f(x,y)dxdy \approx \frac{h_x h_y}{2}\left(f(x_i, y_{j+\frac{1}{2}}) + f(x_{i+1}, y_{j+\frac{1}{2}})\right)\,. \tag{3.3.11}$$

Both these rules are second order accurate. It is easy to verify that

$$\frac{\partial\phi_{k,l}}{\partial x}(x_{k+\frac{1}{2}}, y_l) = -\frac{1}{h_x}\,, \qquad \frac{\partial\phi_{k,l}}{\partial x}(x_{k-\frac{1}{2}}, y_l) = \frac{1}{h_x}\,, \tag{3.3.12}$$

$$\frac{\partial\phi_{k,l}}{\partial x}(x_k, y_{l+\frac{1}{2}}) = -\frac{1}{h_y}\,, \qquad \frac{\partial\phi_{k,l}}{\partial x}(x_k, y_{l-\frac{1}{2}}) = \frac{1}{h_y}\,, \tag{3.3.13}$$

and $\frac{\partial\phi_{k,l}}{\partial y}$, $\frac{\partial\phi_{k,l}}{\partial y}$ vanish at all other points needed in the above quadrature rules. Hence, using (3.3.10) and (3.3.11) in (3.3.9) we get for the diagonal entries in the element stiffness matrix

$$A_{(i,j)(i,j)}^{\tau_{i+\frac{1}{2},j+\frac{1}{2}}} = \int_{\tau_{i+\frac{1}{2},j+\frac{1}{2}}} \alpha_1(x,y)\left(\frac{\partial\phi_{i,j}}{\partial x}\right)^2 dxdy + \int_{\tau_{i+\frac{1}{2},j+\frac{1}{2}}} \alpha_2(x,y)\left(\frac{\partial\phi_{i,j}}{\partial y}\right)^2 dxdy$$
$$\approx \frac{h_x h_y}{2}\left[\alpha_1(x_{i+\frac{1}{2}},y_j)\left(\frac{\partial\phi_{i,j}}{\partial x}\right)^2(x_{i+\frac{1}{2}},y_j) + \alpha_1(x_{i+\frac{1}{2}},y_{j+1})\left(\frac{\partial\phi_{i,j}}{\partial x}\right)^2(x_{i+\frac{1}{2}},y_{j+1})\right]$$
$$+ \frac{h_x h_y}{2}\left[\alpha_1(x_i,y_{j+\frac{1}{2}})\left(\frac{\partial\phi_{i,j}}{\partial y}\right)^2(x_i,y_{j+\frac{1}{2}}) + \alpha_1(x_{i+1},y_{j+\frac{1}{2}})\left(\frac{\partial\phi_{i,j}}{\partial x}\right)^2(x_{i+1},y_{j+\frac{1}{2}})\right],$$

and using (3.3.12) and (3.3.13) we get

$$A_{(i,j)(i,j)}^{\tau_{i+\frac{1}{2},j+\frac{1}{2}}} = \frac{h_x h_y}{2h_x^2}\alpha_1(x_{i+\frac{1}{2}},y_j) + \frac{h_x h_y}{2h_y^2}\alpha_2(x_i,y_{j+\frac{1}{2}}). \tag{3.3.14}$$

Finally assembling the diagonal entries of the global stiffness matrix A by using (3.3.6), i.e. by adding the entries of the element stiffness matrices corresponding to the four elements containing node (x_i,y_j), we get

$$A_{(i,j)(i,j)} = \frac{h_y}{h_x}\left(\alpha_1(x_{i+\frac{1}{2}},y_j) + \alpha_1(x_{i-\frac{1}{2}},y_j)\right)$$
$$+ \frac{h_x}{h_y}\left(\alpha_1(x_i,y_{j+\frac{1}{2}}) + \alpha_2(x_i,y_{j-\frac{1}{2}})\right).$$

The off diagonal entries are calculated in the same way. Using the quadrature rules we have

$$A_{(i,j)(i+1,j)}^{\tau_{i+\frac{1}{2},j+\frac{1}{2}}} = \int_{\tau_{i+\frac{1}{2},j+\frac{1}{2}}} \alpha_1(x,y)\frac{\partial\phi_{i,j}}{\partial x}\frac{\partial\phi_{i+1,j}}{\partial x}\,dxdy + \int_{\tau_{i+\frac{1}{2},j+\frac{1}{2}}} \alpha_2(x,y)\frac{\partial\phi_{i,j}}{\partial y}\frac{\partial\phi_{i+1,j}}{\partial y}\,dxdy$$
$$\approx \frac{h_x h_y}{2}\left[\alpha_1(x_{i+\frac{1}{2}},y_j)\left(\frac{\partial\phi_{i,j}}{\partial x}\frac{\partial\phi_{i+1,j}}{\partial x}\right)(x_{i+\frac{1}{2}},y_j) + \alpha_1(x_{i+\frac{1}{2}},y_{j+1})\left(\frac{\partial\phi_{i,j}}{\partial x}\frac{\partial\phi_{i+1,j}}{\partial x}\right)(x_{i+\frac{1}{2}},y_{j+1})\right]$$
$$+ \frac{h_x h_y}{2}\left[\alpha_1(x_i,y_{j+\frac{1}{2}})\left(\frac{\partial\phi_{i,j}}{\partial y}\frac{\partial\phi_{i+1,j}}{\partial y}\right)(x_i,y_{j+\frac{1}{2}}) + \alpha_1(x_{i+1},y_{j+\frac{1}{2}})\left(\frac{\partial\phi_{i,j}}{\partial x}\frac{\partial\phi_{i+1,j}}{\partial x}\right)(x_{i+1},y_{j+\frac{1}{2}})\right],$$

which simplifies, using (3.3.12) and (3.3.13), to

$$A_{(i,j)(i+1,j)}^{\tau_{i+\frac{1}{2},j+\frac{1}{2}}} = -\frac{h_y}{2h_x}\alpha_1(x_{i+\frac{1}{2}},y_j).$$

Then by assembling the two elements that contain nodes (x_i,y_j) and (x_{i+1},y_j) we get

$$A_{(i,j)(i+1,j)} = -\frac{h_y}{h_x}\alpha_1(x_{i+\frac{1}{2}},y_j).$$

Similarly

$$A_{(i,j)(i-1,j)} = -\frac{h_y}{h_x}\alpha_1(x_{i-\frac{1}{2}},y_j), \quad A_{(i,j)(i,j\pm1)} = -\frac{h_x}{h_y}\alpha_2(x_i,y_{j\pm\frac{1}{2}}),$$

and

$$A_{(i,j)(i-1,j-1)} = A_{(i,j)(i+1,j-1)} = A_{(i,j)(i-1,j+1)} = A_{(i,j)(i+1,j+1)} = 0\,.$$

All the remaining entries in A will be zero because the support of the basis functions at the two nodes do not overlap. Thus the stencil at an interior node (x_i, y_j) is

$$\begin{bmatrix} & -\frac{h_x}{h_y}\alpha_2(x_i, y_{j+\frac{1}{2}}) & \\ -\frac{h_y}{h_x}\alpha_1(x_{i-\frac{1}{2}}, y_j) & -\sum & -\frac{h_y}{h_x}\alpha_1(x_{i+\frac{1}{2}}, y_j) \\ & -\frac{h_x}{h_y}\alpha_2(x_i, y_{j-\frac{1}{2}}) & \end{bmatrix}. \tag{3.3.15}$$

This is equivalent to the finite volume stencil (3.2.14), with $x = \lambda$ and $y = \phi$.

Remark 3.3.1. Note that this stencil is not what we would have obtained by integrating the integrals in (3.3.9) exactly. For example, in the simple case $K = I$ with mesh widths $h_x = h_y = h$, the stencil (using exact integration) resulting from the finite element discretisation at some interior node (x_i, y_j) is

$$\frac{1}{3}\begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix},$$

whereas using the quadrature rules (3.3.10) and (3.3.11), we obtain the finite volume stencil:

$$\begin{pmatrix} & -1 & \\ -1 & 4 & -1 \\ & -1 & \end{pmatrix}. \tag{3.3.16}$$

3.3.3 Extension to Three Dimensions

Consider the three dimensional problem

$$-\nabla \cdot (K\nabla u) = g \qquad \text{on} \qquad \Omega_{3D} = \Omega_{1D} \times \Omega_{2D}\,, \tag{3.3.17}$$

with Dirichlet boundary conditions $u(\boldsymbol{\xi}) = 0$ for $\boldsymbol{\xi} \in \Gamma$, where $\Omega_{1D} \subseteq \mathbb{R}$, $\Omega_{2D} \subseteq \mathbb{R}^2$ and $\boldsymbol{\xi} = (x, y, z)$. As in Section 3.3.1 we solve (3.3.17) by writing it in a weak form which is then approximated on a finite dimensional space. The weak form of (3.3.17) in 3D is: Find $u \in V$ such that

$$a(u, v) = (g, v)_{L^2(\Omega)} \qquad \forall v \in V\,,$$

where

$$a(u,v) = \int_{\Omega_{3D}} \nabla v \cdot (K\nabla u)\; dxdydz\,, \qquad (g,v)_{L^2(\Omega)} = \int_{\Omega_{3D}} gv\; dxdydz\,.$$

A suitable choice for V is the Hilbert space $H_0^1(\Omega_{3D})$. The abstract approximation of the weak form is:
Choose $V_h \subset V$ and find $u_h \in V_h$ such that

$$a(u_h, v_h) = (g, v_h)_{L^2(\Omega)} \qquad \forall v_h \in V_h\,.$$

To find V_h we first decompose Ω_{3D} into a mesh of cubic elements, where the nodes of the mesh are the vertices of the elements and denoted

$$\{(x_i, y_j, z_k) :\; i = 1,\ldots,n_x,\; j = 1,\ldots,n_y,\; k = 1,\ldots,n_z\}\,.$$

The element centred at $(x_{i+\frac{1}{2}}, y_{j+\frac{1}{2}}, z_{k+\frac{1}{2}})$ is denoted $\tau_{i+\frac{1}{2},j+\frac{1}{2},k+\frac{1}{2}}$. As in Section 3.3.1 we choose $V_h \subset H_0^1(\Omega_{3D})$ in a tensor product fashion by decomposing Ω_{1D} and Ω_{2D} into a 1D and 2D grid respectively, and introducing finite dimensional spaces $V_h^{1D} \subset H_0^1(\Omega_{1D})$ and $V_h^{2D} \subset H_0^1(\Omega_{2D})$ on these grids. The nodal basis functions of V_h^{2D} are $\{\phi_{i,j} : i = 1,\ldots,n_x,\; j = 1,\ldots,n_y\}$ (see (3.3.7)) and of V_h^{1D} are

$$\phi_k^z(z) = \begin{cases} \frac{(z-z_{k-1})}{h_z} & z \in [z_{k-1}, z_k] \\ \frac{(z_{k+1}-z)}{h_z} & z \in [z_k, z_{k+1}] \\ 0 & \text{elsewhere} \end{cases}\,.$$

We then choose $V_h = V_h^{2D} \otimes V_h^{1D}$ with the nodal basis functions $\{\phi_{i,j,k} : i = 1,\ldots,n_x,\; j = 1,\ldots,n_y,\; k = 1,\ldots,n_z\}$ of V_h as the set of products of any two basis functions of V_h^{1D} and V_h^{2D}.

As we have already seen, the solution $u_h \in V_h$ to the approximate weak form can be written as a linear combination of the basis functions of V_h, and this leads to a system of equations $A\mathbf{u} = \mathbf{b}$ assembled via the element stiffness matrices. We have

$$A_{(i,j,k)(l,m,n)} = \sum_{i=1}^{n_x}\sum_{j=1}^{n_y}\sum_{k=1}^{n_z} \underbrace{\int_{\tau_{i+\frac{1}{2},j+\frac{1}{2},k+\frac{1}{2}}} \nabla\phi_{i,j,k} \cdot (K\nabla\phi_{l,m,n})\; dxdydz}_{A^{\tau_{i+\frac{1}{2},j+\frac{1}{2},k+\frac{1}{2}}}_{(i,j,k)(l,m,n)}}\,, \qquad (3.3.18)$$

where (x_i, y_j, z_k) and (x_l, y_m, z_n) are two nodes of the mesh and where $A^{\tau_{i+\frac{1}{2},j+\frac{1}{2},k+\frac{1}{2}}}_{(i,j,k)(l,m,n)}$ is an 8×8 matrix with rows and columns corresponding to the eight nodes of $\tau_{i+\frac{1}{2},j+\frac{1}{2},k+\frac{1}{2}}$.

The element stiffness matrix for $\tau_{i+\frac{1}{2},j+\frac{1}{2},k+\frac{1}{2}}$ can be written as

$$A_{(i,j,k)(l,m,n)}^{\tau_{i+\frac{1}{2},j+\frac{1}{2},k+\frac{1}{2}}} = \int_{\tau_{i+\frac{1}{2},j+\frac{1}{2},k+\frac{1}{2}}} \alpha_1(x,y,z)\frac{\partial\phi_{i,j,k}}{\partial x}\frac{\partial\phi_{l,m,n}}{\partial x}\,dxdydz$$
$$+\int_{\tau_{i+\frac{1}{2},j+\frac{1}{2},k+\frac{1}{2}}} \alpha_2(x,y,z)\frac{\partial\phi_{i,j,k}}{\partial y}\frac{\partial\phi_{l,m,n}}{\partial y}\,dxdydz+\int_{\tau_{i+\frac{1}{2},j+\frac{1}{2},k+\frac{1}{2}}} \alpha_3(x,y,z)\frac{\partial\phi_{i,j,k}}{\partial z}\frac{\partial\phi_{l,m,n}}{\partial z}\,dxdydz\,. \tag{3.3.19}$$

The first integral is approximated by the midpoint rule in x and the trapezoidal rule in y and z, i.e.

$$\int_{\tau_{i+\frac{1}{2},j+\frac{1}{2},k+\frac{1}{2}}} f(x,y,z)\,dxdydz =$$
$$\frac{h_xh_yh_z}{4}\left[f(x_{i+\frac{1}{2}},y_j,z_k)+f(x_{i+\frac{1}{2}},y_{j+1},z_k)+f(x_{i+\frac{1}{2}},y_j,z_{k+1})+f(x_{i+\frac{1}{2}},y_{j+1},z_{k+1})\right].$$

The second integral is approximated by the midpoint rule in y and the trapezoidal rule in x and z, i.e.

$$\int_{\tau_{i+\frac{1}{2},j+\frac{1}{2},k+\frac{1}{2}}} f(x,y,z)\,dxdydz =$$
$$\frac{h_xh_yh_z}{4}\left[f(x_i,y_{j+\frac{1}{2}},z_k)+f(x_{i+1},y_{j+\frac{1}{2}},z_k)+f(x_i,y_{j+\frac{1}{2}},z_{k+1})+f(x_{i+1},y_{j+\frac{1}{2}},z_{k+1})\right],$$

and the third integral is approximated by the midpoint rule in z and the trapezoidal rule in x and y, i.e.

$$\int_{\tau_{i+\frac{1}{2},j+\frac{1}{2},k+\frac{1}{2}}} f(x,y,z)\,dxdydz =$$
$$\frac{h_xh_yh_z}{4}\left[f(x_i,y_j,z_{k+\frac{1}{2}})+f(x_i,y_{j+1},z_{k+\frac{1}{2}})+f(x_{i+1},y_j,z_{k+\frac{1}{2}})+f(x_{i+1},y_{j+1},z_{k+\frac{1}{2}})\right].$$

For the diagonal entries of the element stiffness matrix, we have

$$A_{(i,j,k)(i,j,k)}^{\tau_{i+\frac{1}{2},j+\frac{1}{2},k+\frac{1}{2}}} = \frac{h_yh_z}{4h_x}\alpha_1(x_{i+\frac{1}{2}},y_j,z_k)+\frac{h_xh_z}{4h_y}\alpha_1(x_i,y_{j+\frac{1}{2}},z_k)+\frac{h_xh_y}{4h_z}\alpha_1(x_i,y_j,z_{k+\frac{1}{2}})\,.$$

Assembling the diagonal entries of the global stiffness matrix A by using (3.3.18), we get

$$A_{(i,j,k)(i,j,k)} = \frac{h_yh_z}{h_x}\left(\alpha_1(x_{i+\frac{1}{2}},y_j,z_k)\ +\ \alpha_1(x_{i-\frac{1}{2}},y_j,z_k)\right)$$
$$+\frac{h_xh_z}{h_y}\left(\alpha_2(x_i,y_{j+\frac{1}{2}},z_k)\ +\ \alpha_2(x_i,y_{j-\frac{1}{2}},z_k)\right)+\frac{h_xh_y}{h_z}\left(\alpha_3(x_i,y_j,z_{k-\frac{1}{2}})\ +\ \alpha_3(x_i,y_j,z_{k+\frac{1}{2}})\right).$$

Similarly we have

$$A_{(i,j,k)(i\pm1,j,k)} = -\frac{h_y h_z}{h_x}\alpha_1(x_{i\pm\frac{1}{2}}, y_j, z_k)\,,$$
$$A_{(i,j,k)(i,j\pm1,k)} = -\frac{h_x h_z}{h_y}\alpha_2(x_i, y_{j\pm\frac{1}{2}}, z_k)\,,$$
$$A_{(i,j,k)(i,j,k\pm1)} = -\frac{h_x h_y}{h_z}\alpha_3(x_i, y_j, z_{k\pm\frac{1}{2}})\,.$$

All other entries in A will be zero. Then the stencil at the interior node (x_i, y_j, z_k) is

$$-H_z\alpha_3(x_i, y_j, z_{k-\frac{1}{2}}) \begin{bmatrix} & -H_y\alpha_2(x_i, y_{j+\frac{1}{2}}, z_k) & \\ -H_x\alpha_1(x_{i-\frac{1}{2}}, y_j, z_k) & -\sum & -H_x\alpha_1(x_{i+\frac{1}{2}}, y_j, z_k) \\ & -H_y\alpha_2(x_i, y_{j-\frac{1}{2}}, z_k) & \end{bmatrix} -H_z\alpha_3(x_i, y_j, z_{k+\frac{1}{2}}) \tag{3.3.20}$$

which is the same as the finite volume stencil (3.2.8) with $x = \xi_1$, $y = \xi_2$ and $z = \xi_3$, and where $H_x = \frac{h_y h_z}{h_x}$, $H_y = \frac{h_x h_z}{h_y}$ and $H_z = \frac{h_x h_y}{h_z}$.

Remark 3.3.2. For the case $K = I$ with mesh widths $h_x = h_y = h_z = h$, the stencil (using exact integration of (3.3.19)) resulting from the finite element discretisation at some interior node (x_i, y_j, z_k) is

$$\frac{h}{3}\begin{pmatrix} -\frac{1}{4} & -\frac{1}{2} & -\frac{1}{4} \\ -\frac{1}{2} & 0 & -\frac{1}{2} \\ -\frac{1}{4} & -\frac{1}{2} & -\frac{1}{4} \end{pmatrix}\begin{pmatrix} -\frac{1}{2} & 0 & -\frac{1}{2} \\ 0 & 8 & 0 \\ -\frac{1}{2} & 0 & -\frac{1}{2} \end{pmatrix}\begin{pmatrix} -\frac{1}{4} & -\frac{1}{2} & -\frac{1}{4} \\ -\frac{1}{2} & 0 & -\frac{1}{2} \\ -\frac{1}{4} & -\frac{1}{2} & -\frac{1}{4} \end{pmatrix}. \tag{3.3.21}$$

where the terms in the first and third brackets represent entries corresponding to neighbours on z-level $k-1$ and $k+1$ respectively. By using the three quadrature rules from above instead, we obtain the finite volume stencil:

$$-h\begin{pmatrix} & -h & \\ -h & 6h & -h \\ & -h & \end{pmatrix} - h\,.$$

Remark 3.3.3. The tensor product, $\otimes$, of square matrices for $A = (a_{i,j})_{i,j} \in \mathbb{R}^{n,n}$ and $B \in \mathbb{R}^{m,m}$ is defined by

$$A \otimes B = \begin{pmatrix} a_{11}B & a_{12}B & \cdots & a_{1n}B \\ a_{21}B & a_{22}B & \cdots & a_{2n}B \\ \vdots & \vdots & & \vdots \\ a_{n1}B & a_{n2}B & \cdots & a_{nn}B \end{pmatrix}. \tag{3.3.22}$$

Using this definition, the the 3D operator represented by the stencil (3.3.20) can be found using the 2D operator represented by stencil (3.3.15) and the finite volume discretisation of a 1D Poisson-type operator whose stencil representation is

$$\left[\begin{array}{ccc} -\alpha_3^3(z_{k-\frac{1}{2}})\frac{1}{h_z} & -\sum & -\alpha_3^3(z_{k+\frac{1}{2}})\frac{1}{h_z} \end{array} \right] .$$

Denoting A_{1D}, A_{2D}, A_{3D} as the above one, two and three dimensional operators, respectively, we find A_{3D} by

$$A_{3D} = A_{2D} \otimes h_z \alpha_1^3(z_k) I_{1D} + h_x h_y \alpha_3^1(x_k) \alpha_3^2(y_j) I_{2D} \otimes A_{1D} \,,$$

where I_{1D} and I_{2D} are the identity matrices of the same dimension as A_{1D} and A_{2D}, respectively.

Chapter 4

Numerical Solution of Large Sparse Systems of Linear Equations

4.1 Introduction and Model Problems

In this chapter we describe numerical methods for solving large sparse linear systems of equations arising from the discretisation of second order elliptic partial differential equations (PDEs). We will mainly follow Briggs, Henson and McCormick [20].

Consider the following boundary value problem for the Poisson-type elliptic equation in one dimension:

$$\begin{aligned} -u''(x) + cu(x) &= f(x) \quad \text{on} \quad \Omega_{1D} = (0,1)^2,\ c \geq 0\,, \\ u(0) &= u(1) = 0\,. \end{aligned} \tag{4.1.1}$$

With $c = 0$ this is the Poisson equation. The simplicity of this equation means it can be solved analytically, but for the purpose of this chapter we solve it numerically instead. The domain Ω_{1D} is split up into n subintervals which introduces the grid $\mathcal{T}_h$ with grid points $x_i = ih$, $i = 1, \cdots, n-1$, where h is the constant width of the subintervals (mesh width). For simplicity we use a uniform mesh which is sufficient for this chapter, but the results in this chapter also extend to non-uniform (but shape regular) meshes. A discrete second order finite difference [2] approximation replaces equation (4.1.1) as

[2]Note that for a uniform mesh, finite volume and finite element discretisations lead to identical algebraic systems (apart from a scaling by h^2). We saw in Chapter 3 how they can also be linked in more general cases via the use of suitable quadrature rules.

follows:

$$\frac{-U_{i+1}+2U_i-U_{i-1}}{h^2}+cU_i=f(x_i)\,,\quad 1\le i\le n-1\,,\qquad (4.1.2)$$
$$U_0=U_n=0\,,$$

where the solution vector $\mathbf{u}=(U_1,U_2,\cdots,U_{n-1})$ approximates the exact solution u at the grid points. (4.1.2) is a system of $n-1$ linear equations, represented in matrix form as

$$\underbrace{\frac{1}{h^2}\begin{bmatrix} 2+ch^2 & -1 & & & \\ -1 & 2+ch^2 & -1 & & \\ & & \ddots & & \\ & & & \ddots & -1 \\ & & & -1 & 2+ch^2 \end{bmatrix}}_{A}\underbrace{\begin{bmatrix} U_1 \\ \\ \vdots \\ \\ U_{n-1} \end{bmatrix}}_{\mathbf{u}}=\underbrace{\begin{bmatrix} f_1 \\ \\ \vdots \\ \\ f_{n-1} \end{bmatrix}}_{\mathbf{b}}.$$

The matrix A is sparse, symmetric ($A=A^T$) and tridiagonal. Analogously, the two-dimensional version of this problem is

$$-u_{xx}-u_{yy}+cu=f(x,y)\quad\text{on}\quad\Omega_{2D}=[0,1]^2,\ c\ge 0\,,$$
$$u=0\quad\text{on}\quad\Gamma_{2D}\,,$$

where Γ_{2D} is the boundary of Ω_{2D}, and the finite difference approximation yields the following system of $N=(n_x-1)\times(n_y-1)$ linear equations

$$\frac{-U_{i-1,j}+2U_{i,j}-U_{i+1,j}}{h_x^2}+\frac{-U_{i,j-1}+2U_{i,j}-U_{i,j+1}}{h_y^2}=f_{ij}\,,\qquad (4.1.3)$$
$$u_{i,0}=u_{i,n_x}=u_{0,j}=u_{n_y,j}=0,\quad 1\le i\le n_x-1,\quad 1\le j\le n_y-1\,,$$

where $h_x=1/n_x$ and $h_y=1/n_y$, and the grid points $(x_i,y_j)=(ih_x,jh_y)$ belong to the two-dimensional grid shown in Figure 4.1. This system in matrix form is sparse, symmetric and block-tridiagonal:

$$\begin{bmatrix} B & -I\alpha & & & \\ -I\alpha & B & -I\alpha & & \\ & \ddots & & & \\ & & & \ddots & -I\alpha \\ & & & -I\alpha & B \end{bmatrix}\begin{bmatrix} U_1 \\ \\ \vdots \\ \\ U_{(n_x-1)(n_y-1)} \end{bmatrix}=\begin{bmatrix} f_1 \\ \\ \vdots \\ \\ f_{(n_x-1)(n_y-1)} \end{bmatrix},$$

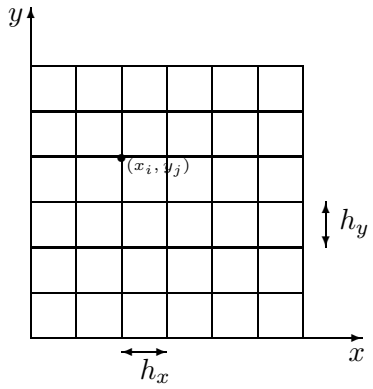

Figure 4-1: The two-dimensional grid on the unit square

where each diagonal block B is an $(n_x-1)\times(n_x-1)$ tridiagonal matrix which looks like the matrix A for the one-dimensional problem. Each off-diagonal block is a multiple, $\alpha = \frac{1}{h_y^2}$, of the $(n_x-1)\times(n_x-1)$ identity matrix.

It is convenient to use stencils to represent discrete equations at certain points on the grid, and the stencil notation was described in Chapter 3. The stencil representation for the linear system (4.1.2) is

$$\frac{1}{h^2}\begin{bmatrix} -1 & 2+ch^2 & -1 \end{bmatrix},$$

and for (4.1.3) is

$$\frac{1}{h^2}\begin{bmatrix} & -1 & \\ -1 & 4+ch^2 & -1 \\ & -1 & \end{bmatrix},$$

assuming $h = h_x = h_y$.

The matrices produced by the discretisation of boundary value problems have desirable properties for numerical methods. Matrices that are sparse and symmetric can be exploited by certain numerical methods. Further desirable properties include *diagonal dominance*, where the entries a_{ij} of the matrix A satisfy

$$\sum_{j\neq i}^{N} |a_{ij}| \leq |a_{ii}|, \quad 1 \leq i \leq N,$$

and *positive definiteness* where $\mathbf{x}^T A\mathbf{x} > 0$ for any non-zero vector $\mathbf{x}$, or equivalently all the eigenvalues of A are real and positive. Symmetric and diagonally dominant matrices with positive diagonal elements are positive definite, and matrices that are symmetric positive definite (SPD) with non-positive off-diagonal entries are M–matrices [20].

During the past 50 years, an enormous amount of work has been devoted to the solution of large, sparse linear systems. Existing methods fall into two main categories: direct or iterative methods. Direct methods, such as Gaussian Elimination or QR factorization [74], compute the solution exactly (up to rounding errors) in a finite number of arithmetic steps. These methods are very robust but generally inefficient, and only specific problems concerning simple geometries can be solved quickly (eg. using Fast Fourier Transforms [21]). Note, however, that there are sparse factorisations, based on a clever ordering of the unknowns that can be highly competitive even up to moderately large problem sizes especially in one and two dimensions, as shown in [48] (see also [29, 33]). Iterative methods on the other hand begin with an initial guess to the solution which generally converges to the actual solution through a succession of simple updating steps (iterations). A convergence criterion and a tolerance *tol* are specified in order to determine when a sufficiently accurate solution has been found. Examples include basic iterative methods such as the Jacobi or Gauss–Seidel methods [64] and Krylov subspace methods such as the conjugate gradient (CG) method [47, 64, 74].

Two factors have to be considered for an effective solution of the system of linear equations. The first is the memory $\mathbf{M}$ that is needed, and the second, and more important factor, is the number $\mathbf{Q}$ of arithmetics operations (FLOPs) required to solve the system. The number of FLOPs required is directly proportional to the CPU time on a simple computer. A method is *optimal* if the number of FLOPs and the memory grow linearly with problem size, ie. $\mathbf{Q} = \mathcal{O}(N)$ and $\mathbf{M} = \mathcal{O}(N)$ (see [66]). In general, both $\mathbf{Q}$ and $\mathbf{M}$ are too high for direct methods to be feasible. For iterative methods, the memory usage is normally optimal (ie. $\mathbf{M} = \mathcal{O}(N)$) and the number of FLOPs *per iteration* is normally $\mathcal{O}(N)$. Thus we require the number of iterations relative to the stopping criteria, $\mathbf{I}(tol)$, to be $\mathcal{O}(1)$. However, $\mathbf{I}(tol)$ usually depends on the condition number of the matrix which becomes worse as the problem size grows, thus iterative methods alone are usually not optimal.

One technique for achieving optimality is preconditioning. A matrix P is chosen such that preconditioning systems can be solved quickly (typically $\mathcal{O}(N)$ operations), and such that the iteration matrix $P^{-\frac{1}{2}}AP^{-\frac{1}{2}}$ is close to the identity matrix, meaning the conditioning of $P^{-\frac{1}{2}}AP^{-\frac{1}{2}}$ is significantly better than that of A. Preconditioning techniques can be very effective, especially for Krylov subspace methods.

Another technique for reducing the number of FLOPs is to use a multigrid technique [20, 75]. Standard iterative methods such as Jacobi or Gauss–Seidel suffer from disabling limitations as they are only effective in reducing the high frequency components of the error (ie. they only *smooth* the error). Thus in order to treat the low frequency components, a coarser grid is introduced. The smoothed residual is restricted

onto the coarse grid and the same procedure, known as the coarse grid correction, is applied recursively. The system on the coarsest grid can be solved using direct or iterative methods. Multigrid techniques are optimal for most systems resulting from the discretisation of elliptic boundary value problems.

In the following sections of this chapter, both techniques for accelerating the iterative methods are applied. In Section 4.2, we analyze the basic iterative methods which, when taking into account the particular structure of the problem, can be used as an effective preconditioner to Krylov subspace methods and as an effective smoother for multigrid methods. Section 4.3 describes the Krylov subspace methods and how preconditioners are used to accelerate them. Amongst the most effective preconditioners are multigrid methods, and in Section 4.4 we describe the main components of the method required for Poisson-type problems and a variety of algorithms that can be used, with some basic theory to confirm the optimal performance of the method. Section 4.5 describes some modifications to the method for solving elliptic problems with anisotropy, and a detailed convergence theory is given in Section 4.6. Finally in Section 4.7, we describe an alternative multigrid method known as *algebraic multigrid* (AMG) which is proven experimentally to be very robust for a wide class of problems and is therefore a very popular method for many industrial applications.

4.2 Basic Iterative Methods

Consider the system of linear equations

$$A\mathbf{u} = \mathbf{b} \tag{4.2.1}$$

and let $\tilde{\mathbf{u}}$ be an approximation to the exact solution $\mathbf{u}$. We denote the algebraic error of $\tilde{\mathbf{u}}$ as

$$\mathbf{e} = \mathbf{u} - \tilde{\mathbf{u}},$$

where $\mathbf{e}$ is a vector whose magnitude can be measured by any of the vector norms. By obtaining the error of $\tilde{\mathbf{u}}$ we can obtain the exact solution, but since the exact solution is not known, $\mathbf{e}$ is not accessible. Another measure of how well $\tilde{\mathbf{u}}$ approximates $\mathbf{u}$ is the *residual*, given by

$$\mathbf{r} = \mathbf{b} - A\tilde{\mathbf{u}}.$$

Hence the residual is the amount by which $\tilde{\mathbf{u}}$ fails to satisfy (4.2.1). If the system has a unique solution then $\mathbf{r} = \mathbf{0} \iff \mathbf{e} = \mathbf{0}$ and we have

$$A\mathbf{e} = A(\mathbf{u} - \tilde{\mathbf{u}}) = \mathbf{b} - A\tilde{\mathbf{u}} = \mathbf{r}.$$

This relationship between the error and the residual, called the *residual equation*, has a vital role in multigrid. By finding an approximate solution, $\tilde{\mathbf{e}}$, of the residual equation, a new approximation is obtained for $\mathbf{u}$ which is given by $\tilde{\mathbf{u}} + \tilde{\mathbf{e}}$. A second vital tool for multigrid methods is the use of basic iterative methods, and we begin by analyzing the performance of these methods on the system of equations (4.2.1).

4.2.1 Point-Wise Relaxation Schemes

Additive Splitting of the Matrix A

Basic iterative methods, or *relaxation schemes*, for solving (4.2.1) are based on splitting $A \in \mathbb{R}^{N\times N}$ into $B - C$ where B is nonsingular. Then the system of equations (4.2.1) is equivalent to

$$B\mathbf{u} = C\mathbf{u} + \mathbf{b},$$

and so the iterative method is cast in the following form:

$$B\mathbf{u}^{(k)} = C\mathbf{u}^{(k-1)} + \mathbf{b}, \tag{4.2.2}$$

where the matrix $S = B^{-1}C$ is known as the iteration matrix. An obvious condition on (4.2.2) is that the solution of (4.2.1) is a fixed point of (4.2.2).

The Jacobi Method

One such iterative method is known as the *Jacobi method*, which splits A into the following:

$$A = D + (L + U)\,,$$

with D the diagonal of A, L the strictly lower triangular part of A and U the strictly upper triangular part of A, ie.

$$D = \begin{pmatrix} a_{11} & & \\ & \ddots & \\ & & a_{NN} \end{pmatrix}, \quad L = \begin{pmatrix} 0 & & & \\ a_{21} & 0 & & \\ \vdots & \ddots & \ddots & \\ a_{N1} & \cdots & a_{N,N-1} & 0 \end{pmatrix}, \quad U = \begin{pmatrix} 0 & a_{21} & \cdots & a_{1N} \\ & \ddots & \ddots & \vdots \\ & & 0 & a_{N-1,N} \\ & & & 0 \end{pmatrix}.$$

Then the Jacobi method is defined by choosing

$$B = D \qquad \text{and} \qquad C = -(L + U)\,.$$

The method is well defined if and only if all the diagonal terms are non-zero. So given an initial guess $\mathbf{u}^{(0)}$, the Jacobi method in matrix/vector form is given by

$$\mathbf{u}^{(k)} = \underbrace{-D^{-1}(L+U)}_{S_J}\mathbf{u}^{(k-1)} + D^{-1}\mathbf{b},$$

where S_J is the Jacobi iteration matrix. We rewrite this as

$$\mathbf{u}^{(k)} = \mathbf{u}^{(k-1)} + D^{-1}(\mathbf{b} - A\mathbf{u}^{(k-1)})\,, \tag{4.2.3}$$

which can be written in component form as follows:

$$u_i^{(k)} = u_i^{(k-1)} + a_{ii}^{-1}\left(b_i - \sum_{j=1}^{N} a_{ij}u_j^{(k-1)}\right), \qquad i = 1, \ldots, N,$$

Since the method updates the approximate solution one component at a time, methods of this type are known as *point-wise relaxation schemes*. Using a weighting factor, ω, we obtain the *weighted* Jacobi method

$$\mathbf{u}^{(k)} = \underbrace{[(1-\omega)I + \omega S_J]}_{S_{J,\omega}}\mathbf{u}^{(k-1)} + D^{-1}\mathbf{b},$$

with $S_{J,\omega}$ being the weighted Jacobi iteration matrix. We will see that for an appropriate choice of ω, the weighted Jacobi method has better smoothing properties.

Gauss–Seidel Method

The (forward) Gauss–Seidel method is defined similarly, with the same splitting $A = D + L + U$, but choosing

$$B = D + L \qquad \text{and} \qquad C = -U\,.$$

Hence we have, after some algebraic manipulations,

$$\mathbf{u}^{(k)} = \mathbf{u}^{(k-1)} + D^{-1}(\mathbf{b} - L\mathbf{u}^{(k)} - U\mathbf{u}^{(k-1)} - D\mathbf{u}^{(k-1)})\,, \tag{4.2.4}$$

or equivalently in component form,

$$u_i^{(k)} = u_i^{(k-1)} + a_{ii}^{-1}\left(b_i - \sum_{j=1}^{i-1} a_{ij}u_j^{(k)} - \sum_{j=i}^{N} a_{ij}u_j^{(k-1)}\right), \qquad i = 1, \ldots, N.$$

Unlike the Jacobi method, the approximate solution is updated immediately after each component is determined, resulting in a faster algorithm. Here, the ordering of the components is important, and we use a *lexicographical* ordering of the grid points. Algorithm 4.1 shows the implementation of the Gauss–Seidel method.

Algorithm 4.1 The Gauss–Seidel Method: `gs(`A`,u,b)`

Choose $\mathbf{u}^{(0)}$
for $k = 1, 2, \ldots,$ until convergence $\ldots$
-2cm-2cm **for** $i = 1, N$
$u_i^{(k)} = u_i^{(k-1)} + a_{ii}^{-1}\left(b_i - \sum_{j=1}^{i-1} a_{ij}u_j^{(k)} - \sum_{j=i}^{N} a_{ij}u_j^{(k-1)}\right)$
end for
end for

4.2.2 Block Relaxation Schemes

A simple generalization of the above methods is known as *block relaxation*. These methods update an entire set of components simultaneously which typically correspond to a subvector containing an entire line of grid points. Thus we partition the vectors $\mathbf{u}$ and $\mathbf{b}$ into subvectors and ensure that these are compatible with the partitioning of A. Suppose each subvector corresponds to an entire x-line on the grid, then the partitioning is as follows:

$$A = \begin{pmatrix} A_{11} & A_{12} & \cdots & A_{1n_y} \\ A_{12} & A_{22} & \cdots & A_{2n_y} \\ \vdots & \ddots & \ddots & \vdots \\ A_{n_y1} & A_{n_y2} & \cdots & A_{n_yn_y} \end{pmatrix}, \quad \mathbf{u} = \begin{pmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \vdots \\ \mathbf{u}_{n_y} \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \vdots \\ \mathbf{b}_{n_y} \end{pmatrix}.$$

Similarly to the point-wise case, the splitting of the matrix A is given by

$$A = D + L + U\,,$$

with

$$D = \begin{pmatrix} A_{11} & & \\ & \ddots & \\ & & A_{n_yn_y} \end{pmatrix}, \; L = \begin{pmatrix} 0 & & & \\ A_{21} & 0 & & \\ \vdots & \ddots & \ddots & \\ A_{n_y1} & \cdots & A_{n_y,n_y-1} & 0 \end{pmatrix}, \; U = \begin{pmatrix} 0 & A_{21} & \cdots & A_{1n_y} \\ & \ddots & \ddots & \vdots \\ & & 0 & A_{n_y-1,n_y} \\ & & & 0 \end{pmatrix}.$$

The methods in matrix/vector form are the same as before, (4.2.3) for the block Jacobi method and (4.2.4) for the block Gauss–Seidel method. In component form the block Gauss–Seidel method is

$$\mathbf{u}_i^{(k)} = \mathbf{u}_i^{(k-1)} + A_{ii}^{-1}\left(\mathbf{b}_i - \sum_{j=1}^{i-1} A_{ij}\mathbf{u}_j^{(k)} - \sum_{j=i}^{n_y} A_{ij}\mathbf{u}_j^{(k-1)}\right), \qquad i = 1, \ldots, n_y.$$

The block relaxation methods therefore require solving a system of equations $A_{ii}\mathbf{v}_i = \mathbf{w}_i$ for $i = 1, \ldots, n_y$ that correspond to an entire line of grid points. These are normally tridiagonal systems which can be solved very efficiently using the Thomas algorithm (aka. the tridiagonal matrix algorithm) [50, Chapter 8]. We see how the block relaxation schemes are implemented in Algorithm 4.2.

Algorithm 4.2 The Block Gauss–Seidel Method: `blockgs(`A`,`u`,`b`)`

Choose $\mathbf{u}^{(0)}$
for $k = 1, 2, \ldots,$ until convergence ...
  for $i = 1, \ldots, n_y$
    $\mathbf{w}_i = \mathbf{b}_i - \sum_{j=1}^{i-1} A_{ij}\mathbf{u}_j^{(k)} - \sum_{j=i}^{n_y} A_{ij}\mathbf{u}_j^{(k-1)}$
    Solve $\mathbf{v}_i = A_{ii}^{-1}\mathbf{w}_i$
    $\mathbf{u}_i^{(k)} = \mathbf{u}_i^{(k-1)} + \mathbf{v}_i$
  end for
end for

4.2.3 Error Analysis

For any of the above iterative methods, we have that

$$\mathbf{u}^{(k)} = S\mathbf{u}^{(k-1)} + \mathbf{g} \qquad \text{and} \qquad \mathbf{u} = S\mathbf{u} + \mathbf{g}\,,$$

for the iteration matrix S. Thus the algebraic error, $\mathbf{e}^{(k)}$, after k iterations, is

$$\mathbf{e}^{(k)} = S\mathbf{e}^{(k-1)}, \tag{4.2.5}$$

and it follows by induction that

$$\mathbf{e}^{(k)} = S^k\mathbf{e}^{(0)}$$

and so taking norms on each side we have

$$\|\mathbf{e}^{(k)}\| \le \|S\|^k\|\mathbf{e}^{(0)}\|\,,$$

where $\|\cdot\|$ denotes any norm. This implies that the methods are convergent if $\|S\| < 1$, for any initial guess $\mathbf{u}^{(0)}$.

Now consider the weighted Jacobi method applied to the one-dimensional Poisson's equation (the one-dimensional problem is used for simplicity):

$$-u''(x) = f(x), \qquad 0 < x < 1, \qquad u(0) = u(1) = 0, \tag{4.2.6}$$

which is discretised using the finite difference method with uniform mesh width h and n subintervals to form a discrete operator $A \in \mathbb{R}^{(n-1)\times(n-1)}$. Recalling that $S_{J,\omega} = (1-\omega)I + \omega S_J$ and $S_J = -D^{-1}(L+U)$, we can obtain $S_{J,\omega}$ in terms of A as follows:

$$S_{J,\omega} = I - \frac{h^2}{2}\omega A.$$

It is known that the eigenvalues and eigenvectors of A are

$$\lambda_i(A) = \frac{4}{h^2}\sin^2\left(\frac{i\pi}{2n}\right), \qquad v_j^{(i)} = \sin\frac{ij\pi}{n}, \qquad i,j = 1,\ldots,n-1,$$

respectively, where i is known as the wavenumber. Therefore the eigenvalues of $S_{J,\omega}$ are

$$\lambda_i(S_{J,\omega}) = 1 - \frac{h^2}{2}\omega\lambda_i(A) = 1 - 2\omega\sin^2\left(\frac{i\pi}{2n}\right), \qquad i = 1,\ldots,n-1. \tag{4.2.7}$$

and the eigenvectors of $S_{J,\omega}$ are the same as those of A.

Now looking at the eigenvectors of A, observe that higher values of i correspond to more highly oscillatory sine waves while lower values of i produce longer smooth waves.

It is possible to expand an arbitrary vector in terms of the eigenvectors because the eigenvectors span $\mathbb{R}^{n-1}$. Thus we can write

$$\mathbf{e}^{(0)} = \sum_{i=1}^{n-1} c_i \mathbf{v}^{(i)},$$

where $\mathbf{e}^{(0)}$ is the initial error, and since m sweeps of the iteration yield $\mathbf{e}^{(m)} = S_{J,\omega}^m \mathbf{e}^{(0)}$, the error after m iterations becomes

$$\mathbf{e}^{(m)} = \sum_{i=1}^{n-1} c_i S_{J,\omega}^m \mathbf{v}^{(i)} = \sum_{i=1}^{n-1} c_i \lambda_i^m(S_{J,\omega})\mathbf{v}^{(i)} \tag{4.2.8}$$

because the eigenvectors of A and $S_{J,\omega}$ are equal. Thus $|\lambda_i^m(S_{J,\omega})|$ must be as small as possible in order to produce the best convergence rate. From (4.2.7), we establish that

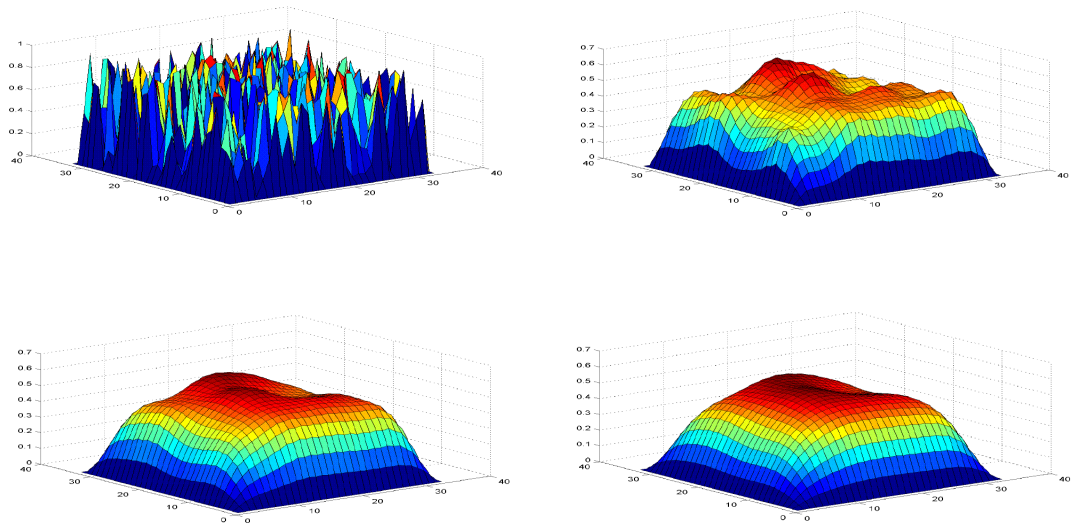

Figure 4-2: The error of a random initial guess after zero iterations (top left), two iterations (top right), four iterations (bottom left) and six iterations (bottom right) of the Gauss–Seidel method. The oscillatory components are removed after very few iterations, but the smooth components are damped very slowly.

$|\lambda_i^m(S_{J,\omega})| < 1$ for all $0 < \omega \leq 1$. Now for small values of i,

$$|\lambda_i(S_{J,\omega})| = |1 - 2\omega \sin^2(hi\pi/2)| \approx |1 - 2\omega h^2 i^2 \pi^2/4| \approx 1.$$

because $h \ll 1$, and so $h^2 i^2 \pi^2 \approx 0$. Thus for small i , λ_i will always be close to 1, meaning that the smooth components of the error cannot be damped very well. For higher values of i (we arbitrarily set these to be $\frac{n-1}{2} \leq i \leq n-1$), ie. for the oscillatory components of the error, the value of ω which provides the best damping is found by imposing the condition $|\lambda_{n/2}| = |\lambda_n|$, and solving this for ω gives an optimal value of $\omega = 2/3$. In this case $|\lambda_k| < 1/3$ for all $\frac{n-1}{2} \leq k \leq n-1$, so all the oscillatory components of the error are reduced by a factor of at least $1/3$ in each iteration.

We see that the weighted Jacobi method damps the oscillatory components of the error very quickly, within a couple of iterations, but the smooth components slowly. It is also possible to show (though not necessarily very easily) that other iterative schemes, such as Gauss–Seidel, possess the same smoothing property, and Figure 4-2 demonstrates this in effect for a two dimensional problem. Hence these methods are known as *smoothers* or *relaxation methods*, and they play a vital role within multigrid methods.

4.3 Preconditioned Krylov Subspace Methods

Another very important class of iterative methods is known as projection methods, as described in [64, Chapter 5]. Consider the linear system

$$A\mathbf{u} = \mathbf{b}.$$

for $A \in \mathbb{R}^{n\times n}$. Let $\mathcal{K}_m$ and $\mathcal{L}_m$ be m-dimensional subspaces of $\mathbb{R}^n$ that are different from each other. Then, given an initial guess $\mathbf{u}^{(0)}$ to the solution, the projection method seeks an approximate solution $\mathbf{u}^{(m)}$ in the space $\mathbf{u}^{(0)} + \mathcal{K}_m$ such that

$$\mathbf{r}^{(m)} \perp \mathcal{L}_m\,,$$

where $\mathbf{r}^{(m)} = \mathbf{b} - A\mathbf{u}^{(m)}$ is the residual. A Krylov subspace method is one which the subspace $\mathcal{K}_m$ is the Krylov subspace

$$\mathcal{K}_m(A, \mathbf{r}^{(0)}) = span\{\mathbf{r}^{(0)}, A\mathbf{r}^{(0)}, A^2\mathbf{r}^{(0)}, \ldots, A^{m-1}\mathbf{r}^{(0)}\}\,. \tag{4.3.1}$$

There are many different choices of Krylov subspace methods, each arising from a different choice of $\mathcal{L}_m$ (see [64, Chapters 6 and 7] for details of each Krylov subspace method). If A is SPD then it is possible to choose $\mathcal{L}_m = \mathcal{K}_m(A, \mathbf{r}^{(0)})$ which yields the conjugate gradient (CG) method [47], the best known iterative technique for solving sparse SPD linear systems. However, for more general non-symmetric linear systems, $\mathcal{L}_m$ is normally associated with the dual system $A^T\mathbf{u}^* = \mathbf{b}^*$, i.e. $\mathcal{L}_m = \mathcal{K}_m(A^T, \mathbf{r}^{(0)*})$, with $\mathbf{r}^{(0)*} = \mathbf{b}^* - A^T\mathbf{u}^{(0)*}$. However, such methods rely on the use of A^T which may not be readily available due to compressed storage techniques of sparse matrices. The biconjugate gradient stabilized (Bi-CGSTAB) method was developed in 1992 by Van der Vorst [76] which avoids the use of A^T, and achieves fast convergence rates for non-symmetric linear systems. The generalized minimal residual (GMRES) method developed by Saad in 1982 [65] also achieves competitive convergence rates for non-symmetric linear systems. It is mathematically equivalent to the generalized conjugate residual (GCR) method [34], which is used operationally at the Met Office, but is more robust and requires less storage.

Krylov subspace methods on their own are generally not optimal, as their convergence rates usually depend on the condition number of A. Thus, in order to reduce the condition number, preconditioning techniques are used.

An SPD preconditioning matrix P is chosen for solving

$$P^{-\frac{1}{2}}AP^{-\frac{1}{2}}\tilde{\mathbf{u}} = \tilde{\mathbf{b}}\,,$$

where

$$P^{\frac{1}{2}}\mathbf{u} = \tilde{\mathbf{u}} \quad \text{and} \quad P^{\frac{1}{2}}\tilde{\mathbf{b}} = \mathbf{b}.$$

We require

- Solves with $P^{\frac{1}{2}}$ to be cheap,
- $\kappa(P^{-\frac{1}{2}}AP^{-\frac{1}{2}})$ to be smaller than $\kappa(A)$.

Normally, either left preconditioning or right preconditioning, i.e.

$$P^{-\frac{1}{2}}A\mathbf{u} = P^{-\frac{1}{2}}\mathbf{b} \qquad \text{or} \qquad AP^{-\frac{1}{2}}P^{\frac{1}{2}}\mathbf{u} = \mathbf{b},$$

are used. Left preconditioning is used for the CG method which is shown in Algorithm 4.3. At the Met Office, the GCR method is accelerated by an ADI preconditioner [11], as discussed in [22, 69].

Algorithm 4.3 Preconditioned conjugate gradient method: `pcg(`$A, \mathtt{u}, \mathtt{b}$`)`

Choose $\mathbf{u}^{(0)}$ (initial solution) and $P^{\frac{1}{2}}$ (preconditioner)
$\mathbf{r}^{(0)} = \mathbf{b}^{(0)} - A\mathbf{u}^{(0)}$ (initial residual)
Solve $\mathbf{z}^{(0)} = P^{-\frac{1}{2}}\mathbf{r}^{(0)}$
$\mathbf{p}^{(0)} = \mathbf{z}^{(0)}$ (initial search direction)
for $k = 0, 1, \ldots,$ until convergence $\ldots$
  $\mathbf{q} = A\mathbf{p}^{(k)}$
  $\alpha_k = (\mathbf{z}^{(k)T}\mathbf{r}^{(k)})/(\mathbf{p}^{(k)T}\mathbf{q})$ (search parameter)
  $\mathbf{u}^{(k+1)} = \mathbf{u}^{(k)} + \alpha_k\mathbf{p}^{(k)}$ (update solution)
  $\mathbf{r}^{(k+1)} = \mathbf{r}^{(k)} - \alpha_k A\mathbf{p}^{(k)}$ (update residual)
  if $\mathbf{r}^{(k+1)}$ sufficiently small **exit**
  Solve $\mathbf{z}^{(k+1)} = P^{-\frac{1}{2}}\mathbf{r}^{(k+1)}$
  $\beta_k = (\mathbf{z}^{(k+1)T}\mathbf{r}^{(k+1)})/(\mathbf{z}^{(k)T}\mathbf{r}^{(k)})$
  $\mathbf{p}^{(k+1)} = \mathbf{z}^{(k+1)} + \beta_k\mathbf{p}^{(k)}$ (update search direction)
end for

4.4 Multigrid Methods

4.4.1 Heuristics

We have established in Section 4.2.3 that the basic iteration procedures suffer in the presence of smooth components of the error. Now consider a grid $\mathcal{T}_h$ with grid spacing h and $\mathcal{T}_{2h}$ with double the grid spacing, $2h$. After a small number of iterations of a

smoother, the error on grid $\mathcal{T}_h$ becomes smooth. However, in passing from a finer grid ($\mathcal{T}_h$) to a coarser grid ($\mathcal{T}_{2h}$), the error effectively becomes more oscillatory because the number of oscillations of the function per node has doubled.

This leads to the idea that when the relaxation method stalls, i.e. when the error is smooth, move onto a coarser grid where the error appears more oscillatory and the relaxation method will become more effective. However, since it is only the error that is smooth, not the solution itself, only the error can be passed between the grids. Thus, on the coarse grid we solve the residual equation, $A\mathbf{e} = \mathbf{r} = \mathbf{f} - A\hat{\mathbf{u}}$, where $\hat{\mathbf{u}}$ is an approximation to the exact solution $\mathbf{u}$ after a certain number of iterations of the smoother. By using a specific initial guess $\mathbf{e} = \mathbf{0}$ to the residual equation, we know that the error of this guess is equal to the value of $\mathbf{e}$ passed onto the coarse grid, which we know was smooth on the fine grid but will be more oscillatory on the coarse grid. Hence the smoother applied to the residual equation on the coarse grid will once again be effective. Let us define matrix A_h and A_{2h} as the operators defined on $\mathcal{T}_h$ and $\mathcal{T}_{2h}$, respectively, and similarly vectors with subscripts h and $2h$ as vectors defined on $\mathcal{T}_h$ and $\mathcal{T}_{2h}$, respectively. Then a basic strategy based on the above heuristics is summarized as follows:

- Smooth on $A_h\mathbf{u}_h = \mathbf{f}_h$ on $\mathcal{T}_h$ for an approximation to the exact solution $\mathbf{u}_h$, which we denote $\hat{\mathbf{u}}^h$.
- Compute the residual $\mathbf{r}_h = \mathbf{f}_h - A_h\hat{\mathbf{u}}_h$ and solve $A_{2h}\mathbf{e}_{2h} = \mathbf{r}_{2h}$ on $\mathcal{T}_{2h}$ to obtain an approximation $\hat{\mathbf{e}}_{2h}$ to $\mathbf{e}_{2h}$
- Update the approximation to $\mathbf{u}_h$ with the estimate of the error obtained on $\mathcal{T}_{2h}$: $\hat{\mathbf{u}}_h := \hat{\mathbf{u}}_h + \hat{\mathbf{e}}_h$

This is known as the *two-grid correction scheme*, where the solve on $\mathcal{T}_{2h}$ is called the *coarse grid correction*. The easiest way to introduce a coarse grid is to simply double the mesh width in each coordinate direction. This is called *full coarsening*. For the uniform grid in the unit square, Figure 4-3 presents a 8×8 grid fully coarsened to an 4×4 grid. Now that we have two grids to work with, the following questions naturally arise:

1. How do we transfer vectors from $\mathcal{T}_h$ to $\mathcal{T}_{2h}$ (e.g. the residual $\mathbf{r}_h$ on $\mathcal{T}_h$ to $\mathbf{r}_{2h}$ on $\mathcal{T}_{2h}$) and vice versa (e.g. the approximation to the error $\hat{\mathbf{e}}_{2h}$ on $\mathcal{T}_{2h}$ to $\hat{\mathbf{e}}_h$ on $\mathcal{T}_h$)?
2. What is A_{2h} on $\mathcal{T}_{2h}$?
3. How do we solve the residual equation on $\mathcal{T}_{2h}$?

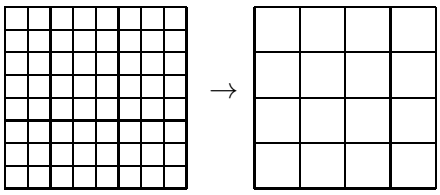

Figure 4-3: Full coarsening on an 8×8 uniform grid

4.4.2 Grid Transfer Operators

The first of the above questions are answered using transfer operators called *restriction* and *interpolation*. The interpolation matrix, P, transfers a vector from the coarse grid space to the fine grid space. Several interpolation methods exist with varying degrees of accuracy, but here we only consider linear interpolation as experiments have shown that it is sufficiently accurate. The restriction matrix, denoted R, transfers a vector from the fine grid space to the coarse grid space, and it is typically the transpose of the interpolation matrix. The construction of these matrices is different for vertex centred (finite difference and finite element methods) and cell centred (finite volume method) grids. Both of these are described for the 1D case – assuming a uniform mesh, full coarsening and homogeneous Dirichlet boundary conditions – as follows:

Vertex Centred Grids

A vertex centred grid with n subintervals will have $n-1$ grid points on $\mathcal{T}_h$ and $\frac{n}{2}-1$ grid points on $\mathcal{T}_{2h}$. The interpolation operator, $P : \mathbb{R}^{\frac{n}{2}-1} \to \mathbb{R}^{n-1}$, takes a coarse grid vector $\mathbf{v}_{2h} \in \mathbb{R}^{\frac{n}{2}-1}$ and produces a fine grid vector $\mathbf{v}_h \in \mathbb{R}^{n-1}$ according to the rule $P\mathbf{v}_{2h} = \mathbf{v}_h$, where

$$\begin{aligned} v_h(2j) &= v_{2h}(j), \qquad 1 \le j \le \frac{n}{2} - 1 \\ v_h(2j+1) &= \frac{1}{2}\left(v_{2h}(j) + v_{2h}(j+1)\right), \qquad 1 \le j \le \frac{n}{2} - 2. \end{aligned}$$

and suitable modifications for the nodes adjacent to the boundaries, i.e. $v_h(1) = \frac{1}{2}v_{2h}(1)$ and $v_h(\frac{n}{2}-1) = \frac{1}{2}v_{2h}(n-1)$. In two dimensions, the interpolation operator can be constructed by a recursive procedure (over the two dimensions) of the 1D linear interpolation.

Note that the interpolation works well only if certain conditions hold. Assume that the real error (which is unknown in practice) is smooth on the fine grid. Then assume that the coarse grid approximation on $\mathcal{T}_{2h}$ is exact at the coarse grid points. Then when this approximation is interpolated back onto the fine grid, the interpolant will be

smooth and a relatively good approximation to the actual error. However, if the real error is oscillatory on the fine grid, the coarse grid approximation will not capture the oscillations and so the interpolant will be a poor approximation to the actual error. Therefore interpolation is effective only when the error is smooth before restricting the residual to the coarse grid, and this is ensured by applying the smoother beforehand.

The restriction operator, $R : \mathbb{R}^{n-1} \to \mathbb{R}^{\frac{n}{2}-1}$, takes a fine grid vector and produces a coarse grid vector, i.e. $R\mathbf{v}^h = \mathbf{v}^{2h}$. The most obvious restriction operator, *injection*, is defined by

$$v_{2h}(j) = v_h(2j) \qquad 1 \leq j \leq \frac{n}{2} - 1.$$

The coarse grid vector simply takes its value directly from the corresponding fine grid point, with all other points ignored. An alternative operator is called *full weighting* which takes weighted averages of values at neighbouring points, ie.

$$v_{2h}(j) = \alpha\left(\frac{1}{2}v_h(2j-1) + v_h(2j) + \frac{1}{2}v_h(2j+1)\right) \qquad 1 \leq j \leq \frac{n}{2} - 1, \tag{4.4.1}$$

where α is an appropriate scaling factor ensuring $\sum_j R_{ij} = 1$ for all i. For the case of (4.4.1), $\alpha = \frac{1}{2}$. The full weighted restriction matrix without the scaling factor is the transpose of the interpolation matrix. In stencil notation it is

$$\frac{1}{4}[\ 1 \quad 2 \quad 1\ ].$$

In two dimensions, the full weighting restriction operator can be constructed using a tensor product of

$$\frac{1}{2}[\ 1 \quad 2 \quad 1\ ] \qquad \text{and} \qquad \frac{1}{2}\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}.$$

Cell Centred Grids

For cell centred grids, the interpolation and restriction is different because, as described in [41], the coarse grid points are not a subset of the fine grid points. With n subintervals, a vertex centred grid will have n grid points on $\mathcal{T}_h$ and $\frac{n}{2}$ grid points on $\mathcal{T}_{2h}$. For $\mathbf{v}_h \in \mathbb{R}^n$ and $\mathbf{v}_{2h} \in \mathbb{R}^{\frac{n}{2}}$, we define the linear interpolation operator $P : \mathbb{R}^{\frac{n}{2}} \to \mathbb{R}^n$ based on the arrangement of the unknowns as $P\mathbf{v}_{2h} = \mathbf{v}_h$ where

$$v_h(2j) = \frac{3}{4}v_{2h}(j) + \frac{1}{4}v_{2h}(j+1), \qquad 1 \leq j \leq \frac{n}{2} - 1$$

$$v_h(2j-1) = \frac{3}{4}v_{2h}(j) + \frac{1}{4}v_{2h}(j-1)\,, \qquad 2 \le j \le \frac{n}{2}$$

and for nodes adjacent to the boundary we have $v_h(1) = \frac{3}{4}v_{2h}(1)$ and $v_h(n) = \frac{3}{4}v_{2h}(\frac{n}{2})$. 2D linear interpolation is defined similarly.

For the restriction operator $R : \mathbb{R}^n \to \mathbb{R}^{\frac{n}{2}}$, either the two point average (or four point average in 2D), i.e.

$$v_{2h}(j) = \frac{1}{2}v_h(2j-1) + \frac{1}{2}v_h(2j), \qquad 1 \le j \le \frac{n}{2}$$

or full weighting, i.e.

$$\begin{aligned}
v_{2h}(j) &= \frac{1}{4}v_h(2j-2) + \frac{3}{4}v_h(2j-1) + \frac{3}{4}v_h(2j) + \frac{1}{4}v_h(2j+1)\,, \qquad 2 \le j \le \frac{n}{2} - 1 \\
v_{2h}(1) &= \frac{3}{4}v_h(1) + \frac{3}{4}v_h(2) + \frac{1}{4}v_h(3)\,, \\
v_{2h}(\frac{n}{2}) &= \frac{1}{4}v_h(n-2) + \frac{3}{4}v_h(n-1) + \frac{3}{4}v_h(n)\,,
\end{aligned}$$

are most commonly used. In stencil notation, these correspond to

$$\frac{1}{2}\begin{bmatrix} 1 & 1 \end{bmatrix} \quad \text{(1D)} \qquad \text{and} \qquad \frac{1}{4}\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \quad \text{(2D)}\,, \tag{4.4.2}$$

and

$$\frac{1}{4}\begin{bmatrix} 1 & 3 & 3 & 1 \end{bmatrix} \quad \text{(1D)} \qquad \text{and} \qquad \frac{1}{4}\begin{bmatrix} & 1 & 1 & \\ 1 & 2 & 2 & 1 \\ 1 & 2 & 2 & 1 \\ & 1 & 1 & \end{bmatrix} \quad \text{(2D)}\,, \tag{4.4.3}$$

Note that for cell centred restriction, a scaling factor is not needed. Figure 4-4 visualizes the difference between vertex centred and cell centred linear interpolation in 2D.

Remark 4.4.1. Let m_a be the order of a differential operator (e.g. $m_a = 2$ for the Poisson-type equations). Further, let P be an interpolation of order m_p (this assumption holds if P interpolates polynomials of degree $m_p - 1$ exactly), and suppose $R = \hat{P}^T$, where $\hat{P}$ is an interpolation of order m_r. Then the following condition given in [44, 75] must be satisfied

$$m_p + m_r > m_a\,. \tag{4.4.4}$$

For example, if $m_a = 2$, then we could choose P and R such that P is a piecewise linear interpolation and $R = P^T$. This would give $m_p + m_r = 4 > m_a$ and so (4.4.4) is satisfied. If R is simple injection and the adjoint of $\hat{P}^T$, then $\hat{P}$ only interpolates

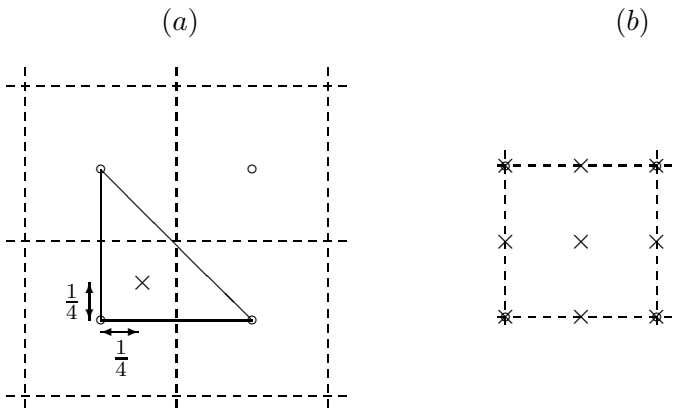

Figure 4-4: Linear interpolation in two-dimensions for (a) a cell centred grid and (b) a vertex centred grid. Circles denote the coarse grid nodes and crosses denote fine grid nodes

constants exactly, and so it is a prolongation of order 1 (i.e. $m_r = 1$). In this case P being a piecewise linear interpolation (i.e. $m_p = 2$) will still satisfy (4.4.4). For cell centred discretisations, the combination of linear interpolation ($m_p = 2$) with full weighting restriction or a four point average ($m_r = 2$ for both) is well suited for the Poisson-type equations.

The Coarse Grid Approximation

The remaining issue is to work out the discrete operator, A_{2h} on $\mathcal{T}_{2h}$. This can be done in two ways:

- Discretise on the coarse grid
- Compute the *Galerkin product* $A_{2h} = RA_hP$

The coarse grid system

$$A_{2h}\mathbf{e}_{2h} = \mathbf{r}_{2h}$$

is then solved using a direct method (exactly) or an iterative method (approximately).

Remark 4.4.2. The Galerkin product is effectively a mapping of the matrix from the fine grid to the coarse grid via the use of the transfer operators (see [75, §2.3.2]). Suppose vertex centred grids are being used, then by using the Galerkin product, the coarse grid correction is unaffected by scaling as we shall see below. Let $\mathbf{u}_h^{(0)}$ is the initial approximation to $\mathbf{u}_h$, then the approximation $\mathbf{u}_h^{(1)}$ after the coarse grid correction

(note that we have ignored smoothing) is

$$\mathbf{u}_h^{(1)} = \mathbf{u}_h^{(0)} + P(\alpha R A_h P)^{-1} \alpha R \mathbf{r}_h,$$

and so the scaling factor, α, is cancelled out. However, if A_{2h} is computed by discretising on each grid, the scaling factor remains. Note also that if cell centred grids are used, then the scaling factor is not present irrespective of the method for finding the coarse grids.

4.4.3 The Two Grid Method (TGM)

Now that we have well defined transfer operators between grids, we can return to the two-grid correction scheme and make it more precise. Algorithm 4.4 shows the method in algorithmic form.

Algorithm 4.4 The Two Grid Method (TGM)

Choose $\mathbf{u}_h^{(0)}$
for $k = 0, 1, \ldots,$ until convergence $\ldots$

$\mathbf{u}_h^{(k)} = S^{\nu_1}(\mathbf{u}_h^{(k)}, \mathbf{b}_h)$	(ν_1 pre-smoothing steps)
$\mathbf{r}_h^{(k)} = \mathbf{b}_h - A_h \mathbf{u}_h^{(k)}$	(calculate residual)
$\mathbf{r}_{2h}^{(k)} = R\mathbf{r}_h^{(k)}$	(restrict residual)
$\mathbf{e}_{2h}^{(k)} \approx A_{2h}^{-1} \mathbf{r}_{2h}^{(k)}$	(coarse grid solve on $\mathcal{T}_{2h}$)
$\mathbf{e}_h^{(k)} = P\mathbf{e}_{2h}^{(k)}$	(interpolate error)
$\mathbf{u}_h^{(k+1)} = \mathbf{u}_h^{(k)} + \mathbf{e}_h^{(k)}$	(correction)
$\mathbf{u}_h^{(k+1)} = S^{\nu_2}(\mathbf{u}_h^{(k+1)}, \mathbf{b}_h)$	(ν_2 post-smoothing steps)

if $\mathbf{r}^{(k+1)}$ sufficiently small **exit**
end for

There are several choices for each of the components in Algorithm 4.4, as discussed already, and we summarize them as follows:

- **Initial Approximation:** Any initial approximation will suffice but it is normally set to $\mathbf{u}_h^{(0)} = \mathbf{0}$.
- **Smoother:** Point-wise Jacobi, Gauss–Seidel and SOR are popular smoothing methods, though block versions of these smoothers are more efficient for particular problems. We denote the smoothing operation generically as $\mathcal{S}^\nu$, where ν denotes the number of smoothing iterations. Very few pre- and post-smoothing steps are necessary, but usually a least one of each are used. If the two-grid iteration is to

be symmetric, we must use an equal number of pre- and post-smoothing steps, noting that the ordering must be reversed in post-smoothing if Gauss–Seidel is used.

- **Transfer Operators:** These depend on whether vertex centred (FDM) or cell centred (FVM) methods are used. Simple prolongation such as piecewise linear or bilinear interpolation are the easiest choices, with the restriction normally chosen as the adjoint of the interpolation. For the vertex centred restriction, a suitable scaling factor α is needed, ie. $R = \alpha P^T$, but for cell centred restriction, no scaling factor is necessary.

- **Coarse Grid Approximation:** Firstly, a suitable approximation to A_h is needed on the coarse grid which can be obtained by discretising on the coarse grid or using the Galerkin product. The coarse grid system

$$A_{2h}\mathbf{e}_{2h} = \mathbf{r}_{2h}$$

is solved with a direct method (exactly) or an iterative method (approximately), which would typically be a preconditioned Krylov subspace method or a block relaxation scheme.

With suitable choices for each component in the TGM, the first smoothing step eliminates the oscillatory components of the error, and assuming the residual equation is solved accurately on $\mathcal{T}_{2h}$, the error transferred back to the fine grid will be smooth, so $\mathbf{e}_h$ will be a good approximation to the exact error.

4.4.4 The Multigrid Method (MGM)

The obvious question from the description of the TGM is: what is the best way to solve the coarse grid approximation? The answer is to apply the TGM recursively to any number of grids, with the mesh width doubling in each coordinate direction for each successively coarser grid. Solving the residual equation is no different to solving the main problem, so $A_{2h}\mathbf{e}^{2h} = \mathbf{r}^{2h}$ on $\mathcal{T}_{2h}$ can be solved by γ iterations of the TGM, with initial approximation $\mathbf{e}_{2h}^{(0)} = \mathbf{0}$. This process can be repeated on successive coarse grids until we reach a chosen 'coarsest' grid, at which the system defined on this grid is solved either directly or iteratively. We label the 'level' of each grid as level $l = F$ (finest grid) down to $l = 1$ (coarsest grid), and we define a sequence of nested grids $\mathcal{T}_1, \ldots, \mathcal{T}_F$ on each level with $\mathcal{T}_1$ being the coarsest grid and $\mathcal{T}_F$ the finest grid. This recursive technique is known as the *Multigrid method* (MGM). For this method we not only need to define the operator A_ℓ on each level $\ell = 1, \ldots, F$ but also the transfer

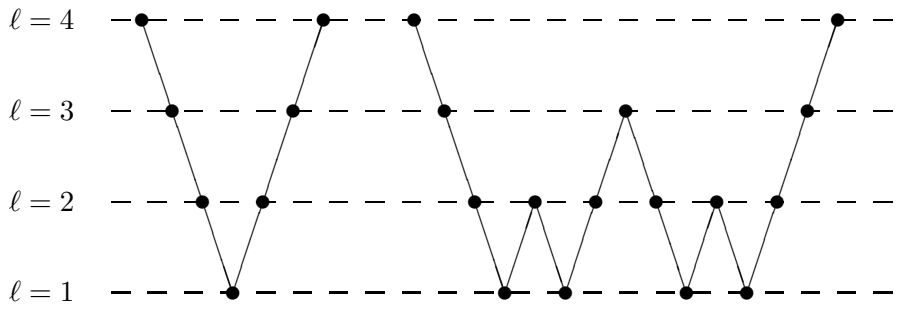

Figure 4-5: The multigrid (a) V-cycle and (b) W-cycle on four grids

operators P_ℓ and R_ℓ on levels $\ell = 2, \ldots, F$, which are mappings of vectors between grids $\mathcal{T}_{\ell-1}$ and $\mathcal{T}_\ell$. Using this approach, the coarsest grid will be significantly smaller than the finest grid. Thus the condition number of the problem on the coarsest grid will be much smaller, as well as the dimension being much smaller, and so it can be solved much faster using an iterative or direct method. Since several grid levels are being used in the MGM, it is more convenient to define all the multigrid components using the grid level rather than the mesh width as subscripts, i.e. the linear system

$$A_F \mathbf{u}_F = \mathbf{b}_F \tag{4.4.5}$$

is defined on the fine grid F. There are several variants to the multigrid method, and the most popular are the V-cycle (when $\gamma = 1$) and the W-cycle (when $\gamma = 2$). Figure 4-5 shows the path between grids for one V-cycle and one W-cycle in the case of four grids (i.e. $F = 4$).

Algorithm 4.5 describes how (4.4.5) is solved using the multigrid V-cycle with F grid levels:

Algorithm 4.5 The Multigrid method (MGM)

Choose $\mathbf{u}_F = \mathbf{u}_F^{(0)}$ and set $\mathbf{r}_F = \mathbf{b}_F$
for $k = 1, \ldots$ until convergence...
    `Vcycle(`$A_F, \mathtt{u}_F, \mathtt{b}_F, F$`)` (apply one V-cycle)
    $\mathbf{r}_F = \mathbf{b}_F - A_F \mathbf{u}_F$ (residual update)
    if ($\| \mathbf{r}_F \| <$ tolerance) **exit**
end for

The core of this method is the recursive subroutine `Vcycle` where all the components of this algorithm are the same as those described for the TGM:

Algorithm 4.6 The Multigrid V-cycle: `Vcycle`$(A_\ell, \mathtt{u}_\ell, \mathtt{b}_\ell, \ell)$

if $(\ell = 1)$ **then**
  $\mathbf{u}_\ell = A_\ell^{-1}\mathbf{b}_\ell$ (solve on coarsest grid $\mathcal{T}_1$)
else
  $\mathbf{u}_\ell = \mathcal{S}_\ell^{\nu_1}(\mathbf{u}_\ell, \mathbf{b}_\ell)$ (ν_1 pre-smoothing steps)
  $\mathbf{r}_\ell = \mathbf{b}_\ell - A_\ell\mathbf{u}_\ell$ (calculate residual on grid $\mathcal{T}_\ell$)
  $\mathbf{r}_{\ell-1} = R_\ell\mathbf{r}_\ell$ (restrict residual onto grid $\mathcal{T}_{\ell-1}$)
  $\mathbf{e}_{\ell-1} = \mathbf{0}$ (set initial guess to zero on grid $\mathcal{T}_{\ell-1}$)
  `Vcycle`$(A_{\ell-1}, \mathtt{e}_{\ell-1}, \mathtt{r}_{\ell-1}, \ell-1)$ (apply one V-cycle on grid $\mathcal{T}_{\ell-1}$)
  $\mathbf{e}_\ell = P_\ell\mathbf{e}_{\ell-1}$ (interpolate error onto grid $\mathcal{T}_\ell$)
  $\mathbf{u}_\ell = \mathbf{u}_\ell + \mathbf{e}_\ell$ (update the solution)
  $\mathbf{u}_\ell = \mathcal{S}_\ell^{\nu_2}(\mathbf{u}_\ell, \mathbf{b}_\ell)$ (ν_2 post-smoothing steps)
end if

The MGM can be used as a stand-alone iterative solver with a sufficient number of V- or W- cycles to reach the stopping criterion, or as a preconditioner to a Krylov subspace method. Multigrid as a preconditioner would typically involve a single iteration of the V- or W-cycle, and the multigrid iteration matrix must be symmetric (i.e. $R_\ell = P_\ell^T$ for each ℓ, $\nu_1 = \nu_2$ and a reverse ordering if Gauss–Seidel is used).

4.4.5 Basic Theory

With the basic multigrid algorithm outlined, we now analyze its performance for the 1D model problem (4.2.6). By denoting the exact solution of (4.2.6) as $u(x)$, and the associated solution to the discrete problem as $\mathbf{u}_F = (U_1, U_2, \ldots, U_{n-1})$, the discretisation error, $\mathbf{E}$, is given by

$$E_i = u(x_i) - U_i, \qquad 1 \leq i \leq n-1.$$

The discretisation error measures how well the exact solution of the discretised problem approximates the exact solution of the continuous problem. With a mesh width of h, it can be shown to be bounded in the discrete l_2 norm (i.e. the Euclidean norm) $\|\cdot\|_2$ by

$$\|\mathbf{E}\|_2 \leq Kh^2. \tag{4.4.6}$$

(cf. [74]) It is unlikely that the discretised problem can be solved exactly, so we also require the algebraic error $\mathbf{e}_F = \mathbf{u}_F - \tilde{\mathbf{u}}_F$, where $\tilde{\mathbf{u}}_F$ approximates $\mathbf{u}_F$ in the discrete problem. We now specify the following tolerance on the total error (algebraic error + discretisation error)

$$\|\mathbf{u} - \tilde{\mathbf{u}}_F\|_2 < tol,$$

where $\mathbf{u}$ is a vector of the exact solution sampled at the grid points. The total error is given by

$$\begin{aligned} \|\mathbf{u} - \tilde{\mathbf{u}}_F\|_2 &\leq \|\mathbf{u} - \mathbf{u}_F\|_2 + \|\mathbf{u}_F - \tilde{\mathbf{u}}_F\|_2 \quad \text{(by triangle inequality)} \\ &= \|\mathbf{E}_F\|_2 + \|\mathbf{e}_F\|_2. \end{aligned}$$

Therefore, we must ensure $\|\mathbf{E}\|_2 + \|\mathbf{e}_F\|_2 < tol$ or $\|\mathbf{E}\|_2 < \frac{tol}{2}$ and $\|\mathbf{e}_F\|_2 < \frac{tol}{2}$ individually. Using (4.4.6), we require that

$$Kh^2 < \frac{tol}{2} \quad \Longrightarrow \quad h < \left(\frac{tol}{2K}\right)^{\frac{1}{2}}. \tag{4.4.7}$$

Thus $\|\mathbf{E}\|_2 < \frac{tol}{2}$ is satisfied if a sufficiently small mesh width is used such that (4.4.7) holds.

Now in order to show $\|\mathbf{e}_F\|_F < \frac{tol}{2}$, consider the V-cycle scheme applied to a one-dimensional problem with approximately n unknowns and $h = \frac{1}{n}$. We assume that the V-cycle scheme has a convergence factor δ that is independent of h (in fact we will prove this rigorously in Section 4.6). The V-cycle scheme must reduce the algebraic error from $\mathcal{O}(1)$, the initial error, to $\mathcal{O}(\frac{tol}{2})$. This means the number of V-cycles required, β, must satisfy

$$\delta^\beta = \frac{tol}{2} \;\Rightarrow\; \beta = \frac{log\ (tol/2)}{log\ \delta} = \mathcal{O}(1).$$

The cost of a single V-cycle scheme is $\mathcal{O}(n)$ so the cost of converging to the level of the discretisation error is $\mathcal{O}(n)$. Since there are n grid points, the optimal cost is $\mathcal{O}(n)$ and so the V-cycle scheme is optimal.

4.5 Anisotropic Problems

The problem treated thus far is isotropic, where direction is unimportant in the equation. More specifically, in the stencil representation of the matrix, the values of the off diagonal entries will be of the same order of magnitude. Note that the entries in the stencil corresponding to the two neighbours in the x-direction will be called *connections* in the x-direction, and likewise for the two neighbours in the y-direction. Now consider a small variation to the two-dimensional Poisson's equation

$$-u_{xx} - \epsilon u_{yy} = f(x, y) \qquad \text{on } \Omega = [0, 1]^2, \tag{4.5.1}$$

with Dirichlet boundary conditions $u = 0$. Suppose this is discretised on the computational domain given in Figure 4.1 to produce a system of equations $A_h \mathbf{u}_h = \mathbf{b}_h$. The

stencil of A_h now looks like

$$\begin{bmatrix} & -\frac{\epsilon}{h_x^2} & \\ -\frac{1}{h_x^2} & \frac{2}{h_x^2}+\frac{2\epsilon}{h_y^2} & -\frac{1}{h_x^2} \\ & -\frac{\epsilon}{h_x^2} & \end{bmatrix}.$$

By choosing $\epsilon \ll 1$, there is now a weak connection in the y-direction (i.e. the connections in the y-direction are much smaller than the connections in the x-direction) causing *anisotropy*. Unfortunately the basic multigrid algorithm doesn't deal with such anisotropic problems very well, and convergence rates degrade as $\epsilon \to 0$. To explain this phenomenon, consider the weighted Jacobi method with weight ω. For the isotropic problem, the eigenvalues of the relaxation matrix are given by

$$\lambda_{k,l} = 1 - \omega\left(\sin^2\left(\frac{k\pi}{2n_x}\right) + \sin^2\left(\frac{l\pi}{2n_y}\right)\right), \quad 1 \le k \le n_x - 1, \quad 1 \le l \le n_y - 1\,.$$

For the anisotropic problem, the eigenvalues (given in [20]) are

$$\lambda_{k,l} = 1 - \frac{2\omega}{1+\epsilon}\left(\sin^2\left(\frac{k\pi}{2n_x}\right) + \epsilon\sin^2\left(\frac{l\pi}{2n_y}\right)\right), \quad 1 \le k \le n_x - 1, \quad 1 \le l \le n_y - 1\,.$$

Here we observe that as $\epsilon \to 0$, there is little variation in the eigenvalues with respect to the wavenumber l, whereas the variation in eigenvalues with respect to k is what we observed for the one-dimensional problem in Section 4.2.3. Thus, the eigenvalues do not change significantly along any of the y-lines (the direction of the weak connection), and using (4.2.8), we observe that this means that neither the high or low frequency components of the error are damped very well along these lines. There are two possible techniques that can be used to recover optimality for anisotropic problems:

Semi coarsening: Coarsen in the strongly coupled direction only.

Line relaxation: Smooth along entire lines in the strongly coupled direction.

4.5.1 Semi Coarsening with Point Relaxation

For the method of semi coarsening, coarsening is applied only in the direction of the strong connection. In problem (4.5.1), semi coarsening will only occur in the x-direction as in Figure 4-6.

The interpolation and restriction matrices are similar to those for a one-dimensional problem, as no coarsening is applied in the y-direction. Thus on each x-line, the one-dimensional interpolation and restriction strategy is applied. The stencil representation for each of the above grids looks as follows:

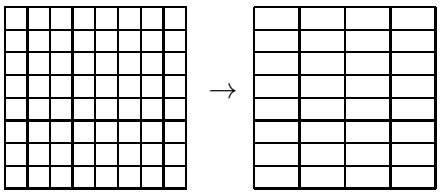

Figure 4-6: Semi coarsening on an 8×8 uniform grid

$$\begin{bmatrix} & -\frac{\epsilon}{h_y^2} & \\ -\frac{1}{h_x^2} & \frac{2}{h_x^2}+\frac{2\epsilon}{h_y^2} & -\frac{1}{h_x^2} \\ & -\frac{\epsilon}{h_y^2} & \end{bmatrix} \rightarrow \begin{bmatrix} & -\frac{\epsilon}{h_y^2} & \\ -\frac{1}{(2h_x)^2} & \frac{2}{(2h_x)^2}+\frac{2\epsilon}{h_y^2} & -\frac{1}{(2h_x)^2} \\ & -\frac{\epsilon}{h_y^2} & \end{bmatrix}.$$

After semi coarsening, the values of the off-diagonal entries become closer in magnitude, so the problem on each coarse level is closer to an isotropic problem. However, semi coarsening too many times may switch the direction of the strong connection (ie. $\frac{1}{(2^n h_x)^2} < \frac{\epsilon}{h_y^2}$ for large n), in which case the method will no longer work. Therefore, this method should be used when the anisotropy is very strong (that is, ϵ is very small), as there is a limit to the number of coarse levels that can be used before the direction of the strong connection switches around.

4.5.2 Line Relaxation with Full Coarsening

By ordering the unknowns along lines of constant x, the matrix A_h and right-hand-side $\mathbf{b}_h$ can be written in block format as

$$A_h = \begin{bmatrix} D & -cI & & & \\ -cI & D & -cI & & \\ & \ddots & D & \ddots & \\ & & \ddots & \ddots & -cI \\ & & & -cI & D \end{bmatrix}, \quad \mathbf{b}_h = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \vdots \\ \vdots \\ \mathbf{b}_{n_y} \end{bmatrix}. \tag{4.5.2}$$

where $c = -\frac{\epsilon}{h_y^2}$ and $D \in \mathbb{R}^{n_x}$ is a tridiagonal matrix whose off-diagonal entries are the connections in the x-direction. $\mathbf{b}_i \in \mathbb{R}^{n_x}$ is a subvector corresponding to an entire x-line of grid points at the i^{th} y-line. Each tridiagonal block is represented by the stencil

$$\frac{1}{h_x^2}\begin{pmatrix} -1 & 2+2\epsilon\frac{h_x^2}{h_y^2} & -1 \end{pmatrix}.$$

The Jacobi method is applied, but now it is applied to entire blocks in the matrix of (4.5.2) instead of the individual entries. One sweep of the 'block' Jacobi method involves solving the tridiagonal system along each x-line (hence the name 'line' relaxation). The j-th sweep of line relaxation involves solving $D\mathbf{u}_i^{(j)} = \mathbf{g}_i^{(j)}$ for $i = 1, \ldots, n_y$ where $\mathbf{g}_i^{(j)} = \mathbf{b}_i - c(\mathbf{u}_{i-1}^{(j)} + \mathbf{u}_{i+1}^{(j-1)})$. The system can be solved exactly as it is a tridiagonal system with Dirichlet boundary conditions, and can be solved very efficiently using the Thomas algorithm [50, Chapter 9] with a cost of $\mathcal{O}(n_x)$.

This strategy is effective when the direction of the strong coupling appears in the diagonal blocks. To see why this is, we need to look at the convergence properties of the iteration matrix, which recall from (4.2.8) is determined by its eigenvalues. The eigenvalues of the block Jacobi iteration (when the direction of the strong coupling appears in the diagonal blocks) are given in [20] as

$$\lambda_{k,l} = 1 - \frac{2\omega}{2\sin^2\left(\frac{k\pi}{2n_x}\right) + \epsilon}\left(\sin^2\left(\frac{k\pi}{2n_x}\right) + \epsilon\sin^2\left(\frac{l\pi}{2n_y}\right)\right), \quad 1 \le k \le n_x - 1, \quad 1 \le l \le n_y - 1\,.$$

The maximum magnitude of these eigenvalues can be shown to be much smaller than the maximum magnitude of the eigenvalues of the point Jacobi iteration, resulting in a greater reduction of the errors. Furthermore, as $\epsilon \to 0$, we have $\lambda_{k,l} \to 1 - \omega$ for all wavenumbers k and l, meaning that all components of the error are reduced by a factor $1 - \omega$ at each iteration, resulting in a very good stand alone solver.

However, if the direction of the weak coupling appears in the diagonal blocks, the eigenvalues of the block Jacobi iteration are

$$\lambda_{k,l} = 1 - \frac{2\omega}{2\sin^2\left(\frac{l\pi}{2n_y}\right) + \epsilon}\left(\sin^2\left(\frac{k\pi}{2n_x}\right) + \epsilon\sin^2\left(\frac{l\pi}{2n_y}\right)\right), \quad 1 \le k \le n_x - 1, \quad 1 \le l \le n_y - 1\,,$$

in which case the maximum magnitude of these eigenvalues is more comparable to those of the point Jacobi iteration, resulting in a less effective smoother.

4.5.3 Semi Coarsening with Line Relaxation

Both the above strategies rely on knowing the direction of the weak and strong connections. Suppose the direction of the weak connection is unknown, then semi coarsening could be used in the one direction and line relaxation in the other (e.g. coarsening only in the x-direction and solving tridiagonal systems along each y-line). Semi coarsening would take care of any strong coupling in the one direction and line relaxation would take care of any strong coupling on the other. This method is therefore the most robust

as it works independently of the size or direction of the anisotropy. However, it still requires the anisotropy to be grid-aligned, and for such cases it is also possible to prove the robustness of the method theoretically. The theory is based on a tensor product analysis for a separation of coordinate directions (see [14] and Section 5.4), and also relies on the classical V-cycle convergence theory for isotropic problems presented in the following section.

4.6 Convergence Analysis of the Multigrid Method

In this section we give a rigorous convergence analysis of the multigrid method, following the work of Hackbusch [44, 45, 43]. Note that more recently, further techniques have been developed (cf. [16, 58, 59, 68, 82, 86, 88]) which generalize the analysis of Hackbusch to cover a more general class of elliptic boundary value problems. However, the majority of multigrid convergence relies on two properties being satisfied, namely the *smoothing property* and the *approximation property*. If these properties are satisfied, then it is possible to show that the convergence rate of the of the method is bounded by a value independent of the problem size and the number of grid levels used. Moreover, if the problem is SPD, a tighter bound is obtained and the analysis is valid for any $\nu > 0$ (where ν is the total number of smoothing steps). In this section we restrict the analysis only to SPD problems, since the majority of problems we deal with in this thesis are SPD. In section 4.6.1, the analysis is given for the two-grid method, and this is extended in Section 4.6.2 to cover the multigrid V-cycle. The convergence rate will be measured by finding the contraction number of the iteration matrix with respect to particular norms.

4.6.1 Analysis of the Two-Grid Method

In this section we give a rigorous convergence analysis of the two-grid method (TGM), where we use the two grids $\mathcal{T}_\ell$ (fine grid) and $\mathcal{T}_{\ell-1}$ (coarse grid). The two-grid iteration is a linear iteration having a representation

$$\mathbf{u}_\ell^{(j+1)} = M_\ell^{TG}\mathbf{u}_\ell^{(j)} + N_\ell^{TG}\mathbf{b}_\ell\,,$$

where the iteration matrix M_ℓ^{TG} depends on the number of pre- and post-smoothing steps ν_1 and ν_2, i.e. $M_\ell^{TG} = M_\ell^{TG}(\nu_1, \nu_2)$. The two-grid iteration matrix is

$$M_\ell^{TG}(\nu_1, \nu_2) = S_\ell^{\nu_2}(I - P_\ell A_{\ell-1}^{-1} R_\ell A_\ell) S_\ell^{\nu_1}\,,$$

(cf. [44, Chapter 6]) where S_ℓ^ν denotes a ν-fold application of S_ℓ. It is well known that the iteration converges if and only if

$$\rho(M_\ell^{TG}(\nu_1,\nu_2)) < 1$$

holds for the spectral radius $\rho(M_\ell^{TG}(\nu_1,\nu_2)) = max\{|\lambda| : \lambda \text{ an eigenvalue of } M_\ell^{TG}\}$. However, it is more common to use norm estimates of the iteration matrix for determining the rate of convergence. The spectral norm $\|\cdot\|_2$ is the matrix norm on $\mathbb{R}^{n\times n}$ induced by the Euclidean vector norm on $\mathbb{R}^n$. Since $\|X\|_2 = \rho(X)$ for a symmetric matrix (c.f. [44, Lemma 1.3.4]), we deduce that the two-grid iteration converges if and only if

$$\| M_\ell^{TG}(\nu_1,\nu_2) \|_2 < 1\,,$$

Note that we will often refer to the Energy norm $\|\cdot\|_{A_\ell}$, which is related to the spectral norm by

$$\| B \|_{A_\ell} = \| A_\ell^{\frac{1}{2}} B A_\ell^{-\frac{1}{2}} \|_2\,. \tag{4.6.1}$$

(cf. [45, Chapter2]) Let us first consider the case of $\nu_2 = 0$ (i.e. no post-smoothing).

We can split M_ℓ^{TG} into two factors

$$M_\ell^{TG}(\nu_1,0) = (A_\ell^{-1} - P_\ell A_{\ell-1}^{-1} R_\ell)(A_\ell S_\ell^{\nu_1})\,,$$

and therefore

$$\| M_\ell^{TG}(\nu_1,0) \|_2 \le \| A_\ell^{-1} - P_\ell A_{\ell-1}^{-1} R_\ell \|_2 \| A_\ell S_\ell^{\nu_1} \|_2\,,$$

by the submultiplicativity of the matrix norm. We consider each factor separately which gives rise to two key properties.

The Smoothing Property

Let us first consider the factor $\| A_\ell S_\ell^{\nu_1} \|_2$. A vector $\mathbf{w}_\ell$ is *smooth* if the product of A_ℓ and $\mathbf{w}_\ell$ satisfies

$$\|A_\ell \mathbf{w}_\ell\|_2 \ll \|A_\ell\|_2 \|\mathbf{w}_\ell\|_2\,,$$

where $\|\mathbf{x}\|_2$ is the Euclidean norm for a vector $\mathbf{x}$. Thus the smoothing procedure S_ℓ is really smoothing if

$$\begin{aligned} &\|A_\ell S_\ell\|_2 \ll \|A_\ell\|_2 \|S_\ell\|_2\,, \qquad \text{and} \\ &\|S_\ell\|_2 \le 1\,. \end{aligned} \tag{4.6.2}$$

Usually $\|A_\ell\|$ is of order $\mathcal{O}(h_\ell^{-\alpha})$ where α is the order of the differential operator (e.g. $\alpha = 2$ for the Poisson equation), and h_ℓ is the mesh width on grid level ℓ. Also, $\|S_\ell\|_2$ is of order $\mathcal{O}(1)$ as long as (4.6.2) holds, leading to the following condition:

Definition 4.6.1 (The smoothing property (SP))**.** An iteration S_ℓ satisfies the SP if there exists functions $\eta(\nu)$ and $\nu_{max}(h)$ and a number α such that

$$\begin{aligned} &\|A_\ell S_\ell\|_2 \le \eta(\nu)\|A_\ell\|_2\,, \quad \forall \ell \ge 1\,, \quad 1 \le \nu \le \nu_{max}(h_\ell)\,, \\ &\eta(\nu) \to 0 \quad \text{as} \quad \nu \to \infty\,, \\ &\nu_{max}(h_\ell) = \infty \quad \text{or} \quad \nu_{max}(h_\ell) \to \infty \quad \text{as} \quad h_\ell \to 0\,. \end{aligned} \tag{4.6.3}$$

Note $\eta(\nu)$ does not depend on ℓ or h_ℓ.

The Approximation Property

Now let us consider the second factor $\| A_\ell^{-1} - P_\ell A_{\ell-1}^{-1} R_\ell \|_2$. The error $\mathbf{e}_\ell$ on the fine grid is approximated by $P_\ell \mathbf{e}_{\ell-1}$ obtained from the residual equation $A_{\ell-1}\mathbf{e}_{\ell-1} = \mathbf{r}_{\ell-1} = R_\ell \mathbf{r}_\ell$ on the coarse grid. This implies that $A_\ell^{-1}\mathbf{r}_\ell \approx P_\ell A_{\ell-1}^{-1} R_\ell \mathbf{r}_\ell$ should be a good approximation. Thus the requirement of $P_\ell \mathbf{e}_{\ell-1}$ being a good approximation to $\mathbf{e}_\ell$ is quantified by

$$\|A_\ell^{-1}\mathbf{r}_\ell - P_\ell A_{\ell-1}^{-1} R_\ell \mathbf{r}_\ell\|_2 \le C'_A \|\mathbf{r}_\ell\|_2 / \|A_\ell\|_2\,.$$

This leads to the approximation property written in terms of the spectral norm:

Definition 4.6.2 (The approximation property (AP))**.** The AP is satisfied if

$$\|A_\ell^{-1} - P_\ell A_{\ell-1}^{-1} R_\ell\|_2 \le C'_A / \|A_\ell\|_2, \qquad \forall \ell \ge 2 \tag{4.6.4}$$

for some constant C'_A.

Theorem 4.6.3 (Proof of the AP)**.** *Let A_ℓ be the matrix of a finite element discretisation, e.g. (3.3.5). Suppose the regularity assumption*

$$\| u_\ell - u \|_{L_2(\Omega)} \le C h_\ell^2 \| f \|_{L_2(\Omega)} \tag{4.6.5}$$

holds, where $\| \cdot \|_{L_2(\Omega)}$ is the $L_2(\Omega)$ norm, and u and u_ℓ are the solutions of (3.3.3) and (3.3.4). Then the AP (4.6.4) holds.

Proof. The proof (given in appendix C) uses a finite element setting and bijective mappings from finite element spaces to vector spaces of the same dimension. □

The SP and AP are the two essential properties of the two-grid and multigrid iterations, and we will see in this section that these conditions are sufficient for two-grid and multigrid convergence.

SPD Systems

From this point on in this section, we only consider SPD matrices. The notation $X \geq Y$ for matrices X and Y signifies that both X and Y are symmetric and that $X - Y$ is positive definite. Using this notation it can be shown that, for any symmetric matrices X and Y,

$$\| X \|_2 \leq c \qquad \text{is equivalent to} \qquad -cI \leq X \leq cI\,, \tag{4.6.6}$$

$$X \leq Y \qquad \text{implies} \qquad C^T X C \leq C^T Y C\,, \tag{4.6.7}$$

$$C^T C \leq X \qquad \Leftrightarrow \qquad C X^{-1} C^T \leq I\,, \tag{4.6.8}$$

$$\|X\|_2^2 = \|X^T X\|_2 = \|X X^T\|_2\,, \tag{4.6.9}$$

where c is an arbitrary constant and C an arbitrary matrix with the same dimensions as X and Y (cf. [44, Chapter 1]). For the two-grid analysis an SPD problem must satisfy the following conditions:

$$A_\ell = A_\ell^T > 0\,, \qquad \forall \ell \geq 1 \tag{4.6.10}$$

$$R_\ell = P_\ell^T\,, \qquad \forall \ell \geq 2 \tag{4.6.11}$$

$$A_{\ell-1} = R_\ell A_\ell P_\ell\,, \qquad \forall \ell \geq 2 \tag{4.6.12}$$

Lemma 4.6.4 (The smoothing property for the symmetric case)**.** *Smoothing iterations are written as $S_\ell = I - W_\ell^{-1} A_\ell$ for an SPD matrix W_ℓ. Suppose that for an SPD matrix A_ℓ, there exists a constant C_W such that*

$$\|W_\ell\|_2 \leq C_W \|A_\ell\|_2\,, \tag{4.6.13}$$

Then if the smoothing iteration satisfies

$$\boxed{W_\ell = W_\ell^T \geq A_\ell > 0}\,, \tag{4.6.14}$$

the smoothing property (4.6.3) holds with $\eta(\nu) = C_W \eta_0(\nu)$ and $\nu_{max}(h_\ell) = \infty$.

Proof. Firstly let us show that for an SPD matrix X, if $0 \leq X \leq I$, then

$$\|X(I - X)^\nu\|_2 = \eta_0(\nu)\,, \tag{4.6.15}$$

with

$$\eta_0(\nu) = \nu^\nu/(1+\nu)^{1+\nu}\,.$$

We show (4.6.15) by setting $f(\xi) = \xi(1-\xi)^\nu$. Then

$$\|X(I-X)^\nu\|_2 = \rho(X(I-X)^\nu) = \max_{0\le\xi\le1} f(\xi)\,,$$

since $0 \le X \le 1$. The maximum of this function occurs at $x = 1/(\nu+1)$ which yields $f(x) = \eta_0(\nu)$ and so (4.6.15) holds.

Now, let $R_\ell = W_\ell - A_\ell$ and define $Y := W_\ell^{-\frac{1}{2}} R_\ell W_\ell^{-\frac{1}{2}}$. Then we deduce that

$$\|A_\ell S_\ell^\nu\|_2 = \|W_\ell^{\frac{1}{2}}(I-Y)Y^\nu W_\ell^{\frac{1}{2}}\|_2 \le \|W_\ell\|_2 \|(I-Y)Y^\nu\|_2\,.$$

We have $\|W_\ell\|_2 \le C_W\|A_\ell\|_2$ by (4.6.13) and $0 \le Y \le 1$ by (4.6.14). Therefore $\|(I-Y)Y^\nu\| \le \eta_0(\nu)$ by (4.6.15), which implies

$$\|A_\ell S_\ell^\nu\|_2 \le C_W\eta_0(\nu)\|A_\ell\|_2\,, \tag{4.6.16}$$

i.e. the smoothing property (4.6.3) holds with $\eta(\nu) = C_W\eta_0(\nu)$. □

Remark 4.6.5. Note that using Lemma 4.6.4 the smoothing property can be satisfied for any SPD matrix A_ℓ and any symmetric smoothing iteration S_ℓ satisfying (4.6.14). In particular, the point- or block-wise damped Jacobi iterations applied to an SPD system satisfy the smoothing property. This is because we recall that $W_\ell = \omega^{-1}D_\ell$, where D_ℓ is the diagonal (or block-diagonal) of A_ℓ, so (4.6.14) is satisfied if $\omega \le \|D_\ell^{-1}A_\ell\|^{-1}$. Then, assuming $D_\ell \le C_D h_\ell^{-2} I$ for some constant C_D, (4.6.16) is satisfied with $C_W = C_D\omega^{-1}h_\ell^{-2}$ (see [44, Proposition 6.2.14]).

Lemma 4.6.6 (The approximation property for the symmetric case)**.** *Suppose (4.6.10) – (4.6.12) hold. Then, provided that (4.6.13) is satisfied, the following inequality is a sufficient condition for the approximation property (4.6.4):*

$$\boxed{0 \le A_\ell^{-1} - P_\ell A_{\ell-1}^{-1} R_\ell \le C_A W_\ell^{-1}}\,. \tag{4.6.17}$$

with $C_A = C_A' C_W$.

Proof. Firstly we deduce that if (4.6.13) holds, (4.6.4) is equivalent to

$$\|W_\ell^{\frac{1}{2}}(A_\ell^{-1} - P_\ell A_{\ell-1}^{-1} R_\ell)W_\ell^{\frac{1}{2}}\|_2 \le C_A' C_W = C_A. \tag{4.6.18}$$

By (4.6.6), (4.6.18) is equivalent to

$$-C_A I \leq W_\ell^{\frac{1}{2}}(A_\ell^{-1} - P_\ell A_{\ell-1}^{-1} R_\ell) W_\ell^{\frac{1}{2}} \leq C_A I\,.$$

Multiplying by $W_\ell^{-\frac{1}{2}}$ from the left and right gives the right-hand inequality of (4.6.17). As for the left-hand inequality, we have by (4.6.12) that $R_\ell A_\ell^{\frac{1}{2}} A_\ell^{\frac{1}{2}} P_\ell = A_{\ell-1}$ and then using (4.6.8) we have that $A_\ell^{\frac{1}{2}} P_\ell A_{\ell-1}^{-1} R_\ell A_\ell^{\frac{1}{2}} \leq I$. Hence $P_\ell A_{\ell-1}^{-1} R_\ell \leq A_\ell^{-1}$ which proves the left-hand inequality. □

For the proofs that follow, we assume the SP and AP hold, but since we are only dealing with SPD problems, it suffices to assume that (4.6.14) and (4.6.17) hold, since (4.6.13) holds for symmetric smoothing iterations. Let us now prove the uniform convergence of the TGM for any $\nu > 0$, firstly for the case of the symmetric iteration matrix, i.e. $\nu_1 = \nu_2 = \frac{\nu}{2}$.

Theorem 4.6.7. *Suppose (4.6.10) – (4.6.12), the smoothing property for SPD problems (4.6.14) and the approximation property for SPD problems (4.6.17) hold. Then the two-grid iteration converges with respect to the energy norm with the rate:*

$$\| M_\ell^{TG}\left(\frac{\nu}{2}, \frac{\nu}{2}\right) \|_{A_\ell} \leq \begin{cases} C_A \frac{\nu^\nu}{(1+\nu)^{1+\nu}} < 1 & 0 \leq C_A \leq 1+\nu \\ \left(1 - \frac{1}{C_A}\right)^\nu < 1 & C_A \geq 1+\nu \end{cases}.$$

Proof. Introduce transformed matrices

$$X_\ell = A_\ell^{\frac{1}{2}} W_\ell^{-1} A_\ell^{\frac{1}{2}}\,, \tag{4.6.19}$$

$$Q_\ell = A_\ell^{\frac{1}{2}}(A_\ell^{-1} - P_\ell A_{\ell-1}^{-1} R_\ell) A_\ell^{\frac{1}{2}} = I - A_\ell^{\frac{1}{2}} P_\ell A_{\ell-1}^{-1} R_\ell A_\ell^{\frac{1}{2}}\,, \tag{4.6.20}$$

$$\hat{S}_\ell = A_\ell^{\frac{1}{2}} S_\ell A_\ell^{-\frac{1}{2}} = A_\ell^{\frac{1}{2}}(I - W_\ell^{-1} A_\ell) A_\ell^{-\frac{1}{2}} = I - X_\ell\,, \tag{4.6.21}$$

$$\hat{M}_\ell^{TG} = A_\ell^{\frac{1}{2}} M_\ell^{TG}\left(\frac{\nu}{2}, \frac{\nu}{2}\right) A_\ell^{-\frac{1}{2}} = \hat{S}_\ell^{\frac{\nu}{2}} Q_\ell \hat{S}_\ell^{\frac{\nu}{2}}\,, \tag{4.6.22}$$

where (4.6.22) follows because A_ℓ and S_ℓ are commutative if a Jacobi smoother is used. We deduce from the definitions of Q_ℓ, X_ℓ and (4.6.17) that

$$0 \leq Q_\ell \leq X_\ell, \tag{4.6.23}$$

and similarly from (4.6.11), we deduce that Q_ℓ is symmetric and from (4.6.10) and (4.6.12) that $A_\ell^{\frac{1}{2}} P_\ell A_{\ell-1}^{-1} R_\ell A_\ell^{\frac{1}{2}} \geq 0$, thus

$$0 \leq Q_\ell \leq 1. \tag{4.6.24}$$

A linear combination of (4.6.23) and (4.6.24) yields

$$0 \leq Q_\ell \leq \alpha C_A X_\ell + (1-\alpha) I \qquad \forall\ 0 \leq \alpha \leq 1. \tag{4.6.25}$$

Multiplication by $\hat{S}_\ell^{\frac{\nu}{2}}$ from both sides gives

$$0 \leq \hat{S}_\ell^{\frac{\nu}{2}} Q_\ell \hat{S}_\ell^{\frac{\nu}{2}} = \hat{M}_\ell \leq (I - X_\ell)^{\frac{\nu}{2}} [\alpha C_A X_\ell + (1-\alpha) I](I - X_\ell)^{\frac{\nu}{2}}\,, \tag{4.6.26}$$

Now $0 \leq X_\ell = A_\ell^{\frac{1}{2}} W_\ell^{-1} A_\ell^{\frac{1}{2}}$, and by (4.6.14) we have $W_\ell^{-1} \leq A_\ell^{-1}$ therefore $0 \leq X_\ell \leq I$. The spectral norm of the right-hand side of the inequality (4.6.26) is equal to

$$sup\{(1-x)^\nu(\alpha C_A x + 1 - \alpha) : x \in \text{ spectrum of } X_\ell\}, \qquad \forall\ 0 \leq \alpha \leq 1.$$

Since $0 \leq X_\ell \leq 1$, the above value for the spectral norm is bounded above by

$$m(\alpha) = max_{0 \leq \xi \leq 1}\{(1-x)^\nu(\alpha C_A x + 1 - \alpha)\}, \qquad \forall\ 0 \leq \alpha \leq 1.$$

Therefore we have

$$\|\hat{M}_\ell^{TG}\|_2 = \|M_\ell^{TG}\|_{A_\ell} \leq m(\alpha).$$

If $0 \leq C_A \leq 1 + \nu$, choose $\alpha = 1$ and $max\ m(1) = C_A \frac{\nu^\nu}{(\nu+1)^{\nu+1}}$ at $\xi = \frac{1}{1+\nu}$. Otherwise choose $\alpha = \frac{\nu}{C_A - 1} \in [0,1]$, then we have $max\ m(\alpha) = (1 - \frac{1}{C_A})^\nu$ at $\xi = \frac{1}{C_A}$, which proves the result. □

Now the proof of the two-grid convergence need not be restricted to the case where the number of pre- and post-smoothing steps are equal. For the more general case of different pre- and post-smoothing steps, one can apply:

Corollary 4.6.8. *Suppose (4.6.10) – (4.6.12), the smoothing property for SPD problems (4.6.14), the approximation property for SPD problems (4.6.17) and* $\nu_1,\ \nu_2 > 0$. *Then the contraction number of the two-grid iteration with respect to the energy norm is bounded by*

$$\|\ M_\ell^{TG}(\nu_1,\nu_2)\ \|_{A_\ell} \leq \|\ M_\ell^{TG}(\nu_1,\nu_1)\ \|_{A_\ell}^{\frac{1}{2}} \|\ M_\ell^{TG}(\nu_2,\nu_2)\ \|_{A_\ell}^{\frac{1}{2}}\ . \tag{4.6.27}$$

Proof. By simple manipulations of M_ℓ^{TG}, we can deduce that

$$M_\ell^{TG}(\nu_1,\nu_2) = M_\ell^{TG}(0,\nu_2) M_\ell^{TG}(\nu_1,0)\,, \tag{4.6.28}$$

$$M_\ell^{TG}(0,\nu)^T = M_\ell^{TG}(\nu,0)\,, \tag{4.6.29}$$

assuming (4.6.11) and (4.6.12) and the commutativity of A_ℓ and S_ℓ. Then using

the transformed iteration matrix $\hat{M}_\ell^{TG}(\nu_1,\nu_2) = A_\ell^{\frac{1}{2}} M_\ell^{TG}(\nu_1,\nu_2) A_\ell^{-\frac{1}{2}}$, we obtain from (4.6.28) and (4.6.1) that

$$\|M_\ell^{TG}(\nu_1,\nu_2)\|_{A_\ell} \leq \|M_\ell^{TG}(0,\nu_2)\|_{A_\ell} \|M_\ell^{TG}(\nu_1,0)\|_{A_\ell}. \tag{4.6.30}$$

Then using (4.6.9), (4.6.28) and (4.6.29) we get

$$\|M_\ell^{TG}(0,\nu_2)\|_{A_\ell}^2 = \|M_\ell^{TG}(\nu_2,\nu_2)\|_{A_\ell}. \tag{4.6.31}$$

(4.6.30) and (4.6.31) imply the result (4.6.27). By Theorem 4.6.7, both the terms on the right-hand-side of (4.6.27) are bounded by contraction numbers less than 1. □

4.6.2 Analysis of the Multigrid V-Cycle

We now analyze the convergence of the multigrid method, where we specifically focus on the V-cycle. Let

$$\mathbf{u}_\ell^{(j+1)} = M_\ell^{MG} \mathbf{u}_\ell^{(j)} + N_\ell^{MG} \mathbf{b}_\ell\,, \tag{4.6.32}$$

where $M_\ell^{MG} = M_\ell^{MG}(\nu_1,\nu_2)$ is the multigrid iteration matrix defined recursively as:

Lemma 4.6.9. *The iteration matrix $M_\ell^{MG}(\nu_1,\nu_2)$ of the multigrid method is*

$$M_\ell^{MG} = \begin{cases} 0 & \text{for } \ell = 1 \text{ (the coarsest grid)} \\ M_\ell^{TG} & \text{for } \ell = 2 \\ M_\ell^{TG} + S_\ell^{\nu_2} P_\ell (M_{\ell-1}^{MG})^\gamma A_{\ell-1}^{-1} R_\ell A_\ell S_\ell^{\nu_1} & \text{for } \ell > 2 \end{cases}.$$

Proof. A full proof is given in Appendix B. □

$\gamma = 1$ and $\gamma = 2$ represent the V-cycle and the W-cycle respectively, but for the remainder of this section we only consider the V-cycle, and we denote $M_\ell^V = M_\ell^{MG}$. We now give the famous result given by Hackbusch [44, Theorem 7.2.2] for the convergence of the V-cycle:

Theorem 4.6.10 (Convergence of the V-cycle)**.** *Suppose (4.6.10) – (4.6.12), the smoothing property for SPD problems (4.6.14) and the approximation property for SPD problems (4.6.17) hold. If $\nu_1 = \nu_2 = \frac{\nu}{2}$, then the V-cycle converges with respect to the energy norm for $\nu > 0$ and the contraction number is bounded by*

$$\|M_\ell^V \left(\frac{\nu}{2}, \frac{\nu}{2}\right)\|_{A_\ell} \leq \frac{C_A}{C_A + \nu} < 1\,. \tag{4.6.33}$$

Proof. Let us first recall the matrices X_ℓ, Q_ℓ, $\hat{S}_\ell$ and $\hat{M}_\ell^{TG}$ from (4.6.19) – (4.6.22). In addition, denote

$$\hat{P}_\ell = A_\ell^{\frac{1}{2}} P_\ell A_{\ell-1}^{\frac{1}{2}}, \qquad \hat{R}_\ell = A_{\ell-1}^{\frac{1}{2}} R_\ell A_\ell^{\frac{1}{2}},$$

then we have that

$$Q_\ell = I - \hat{P}_\ell \hat{R}_\ell. \tag{4.6.34}$$

We prove (4.6.33) by induction. Starting with $\ell = 1$, we know $M_\ell^V = 0$, thus we have

$$\|M_\ell^V \left(\frac{\nu}{2}, \frac{\nu}{2}\right)\|_{A_\ell} = 0 < \frac{C_A}{C_A + \nu}.$$

Now suppose the inequality holds for $\ell - 1$. Using (4.6.22), (4.6.34), (4.6.25) and the inductive hypothesis, the transformed V-cycle iteration matrix on level ℓ is bounded above by

$$\hat{M}_\ell^V \leq \hat{S}_\ell^{\nu_2}((1 - \left(\frac{C_A}{C_A + \nu}\right))(\alpha C_A X_\ell + (1-\alpha)I) + \left(\frac{C_A}{C_A + \nu}\right) I)\hat{S}_\ell^{\nu_1} = f(X_\ell, \alpha)\,,$$

and since $\hat{S}_\ell = I - X_\ell$ and $0 \leq X_\ell \leq 1$ we have

$$f(\xi, \alpha) \leq (1-\xi)^\nu \left[\left(\frac{\nu}{C_A + \nu}\right)(\alpha C_A \xi + (1-\alpha)) + \frac{C_A}{C_A + \nu}\right],$$

for all $\xi \in [0,1]$ and $\alpha \in [0,1]$. The spectral norm of $\hat{M}_\ell^V$ is bounded above by the maximum of $f(\xi, \alpha)$, which occurs at $\xi = 0$. Then choosing $\alpha = 1$ gives

$$f(0,1) \leq \frac{C_A}{C_A + \nu}\,,$$

and so (4.6.33) holds for all $\ell > 1$, which completes the proof. □

For the more general case of different pre- and post-smoothing steps, the convergence rate is found via the following Corollary:

Corollary 4.6.11 (Convergence of the V-cycle for a different number of pre- and post-smoothing iterations)**.** *Under the conditions of Theorem 4.6.10 and if $\nu_1 + \nu_2 > 0$, the V-cycle converges with respect to the energy norm, and the contraction number is bounded by*

$$\|M_\ell^V(\nu_1, \nu_2)\|_{A_\ell} \leq \frac{C_A}{\sqrt{C_A + \nu_1}\sqrt{C_A + \nu_2}} < 1$$

Proof. Firstly we can show by induction that

$$M_\ell^V(\nu_1,\nu_2) = M_\ell^V(0,\nu_2)M_\ell^V(\nu_1,0)\,,$$

and also by simple algebraic manipulations and the commutativity of A_ℓ and S_ℓ that

$$M_\ell^V(0,\nu)^T = M_\ell^V(\nu,0)\,.$$

Then using the transformed matrix $\hat{M}_\ell^V(\nu_1,\nu_2) = A_\ell^{\frac{1}{2}} M_\ell^V(\nu_1,\nu_2) A_\ell^{-\frac{1}{2}}$, we follow the same lines as the proof for Corollary 4.6.8. □

All the analysis in Section 4.6 can be applied to the 1D and 2D model problems discussed throughout this chapter, but only for the isotropic cases. For anisotropic problems, such as model problem (4.5.1), the AP fails and so we must seek alternative techniques to prove the uniform convergence of multigrid methods when applied to these problems, as shown in [58, 70] where line relaxation and full coarsening are used. We will also show in Section 5.4 that the robustness of multigrid methods for anisotropic problems can be proved by using techniques motivated by a method known as *Fourier analysis*. The Fourier analysis of multigrid methods differs from the approach described in this section, in that the eigenvectors of the operator A_ℓ play a pivotal role in transforming the main problem into a family of smaller problems with which the analysis can be done more trivially (A two-grid Fourier analysis for the simple 1D Poisson equation (4.2.6) is given in Appendix A, noting that the analysis can be extended to the multidimensional case). Using this idea, the uniform convergence of two particular anisotropic model problems from Chapter 5 will be proved in Section 5.4 based on a separation of coordinate directions using a tensor product approach. This effectively reduces the analysis of the anisotropic problems to that of a family of simpler isotropic problems, for which the analysis from this section can be applied.

4.7 Algebraic Multigrid (AMG)

Thus far we have been analyzing problems with structured grids and simple coefficients, where the coarse grid selection has been based on simple coarsening strategies and grid transfer operators. These multigrid techniques are called *geometric multigrid* methods. They, however, are in general not optimal for solving problems with highly varying coefficients or unstructured grids, and for such problems we refer to a technique known as *algebraic multigrid* (AMG). Both these methods have two distinct phases. In the *setup* phase, all the multigrid components are defined, such as the hierarchy

of coarse grids, the operators in each grid level and the intergrid transfer operators. The *solution* phase follows, consisting of a stand alone iteration of multigrid cycles or a preconditioning of a Krylov subspace method by multigrid cycles. AMG provides robust solution methods for a wide class of problems posed on unstructured grids, and as a result it is becoming increasingly popular for many industrial applications. The fundamental approach to AMG, as shown in [20, 71, 63, 18], is as follows:

- Fix the smoother (normally a simple point-wise smoother).
- Choose the coarse grids and interpolation operators based on information from the operator.
- Define the coarse grid operator and restriction operator.

This technique is conceptually the opposite of geometric multigrid. Geometric approaches employ pre-determined and fixed grid hierarchies, and choose an appropriate smoother to obtain an efficient interplay between the smoothing and coarse grid correction. Conversely, AMG fixes the smoother to a simple relaxation scheme, and enforces an efficient interplay with the coarse grid correction by choosing the coarser levels and interpolation operators appropriately. The coarse levels are chosen from information based solely from the operator, and the operator-dependant interpolation is generated on the principle that its range is forced to contain functions which are unaffected by relaxation, i.e. smooth functions. The coarsening process is fully automatic, and this is the reason why AMG can adapt itself to the specific requirements of a problem and be robust for a large class of problems despite using simple relaxation methods.

The flexibility of AMG comes at a cost: its setup phase. The cost of constructing the coarse grids and interpolation operators is, in general, larger for AMG, and so it is usually less efficient than geometric multigrid when applied to problems for which geometric multigrid can be applied efficiently. However, the strengths of AMG are its applicability to more complex problems that are beyond the scope of the geometric approach, so it should be considered as a variant to geometric multigrid rather than a competitor.

Throughout the discussion of AMG, we assume that the operator A_ℓ is an SPD *M-matrix*, and we identify the grid points with the set of unknowns, i.e. if we solve $A_\ell \mathbf{u}_\ell = \mathbf{b}_\ell$, and

$$\mathbf{u} = [u_1, u_2, \ldots, u_n]^T,$$

then the fine grid points can be denoted $\{1, 2, \ldots, n\}$. The connections within the grid at each node are then defined by *undirected adjacency graphs*. For an entry a_{ij} of A_ℓ, the vertices of the graph are associated with the grid points, and an edge between the

i^{th} and j^{th} vertex means that $a_{ij} \neq 0$ (which implies that $a_{ji} \neq 0$ since A_ℓ is symmetric). By describing the problem without the use of grids, it is possible to develop the AMG algorithm even for problems which are defined on highly unstructured grids.

4.7.1 Algebraic Smoothness

In AMG, one of the fundamental concepts used for the coarsening and interpolation strategies is the idea of *algebraic smoothness*. Having chosen a simple relaxation scheme, we now define an algebraically smooth error as one that is not effectively reduced by relaxation. Recall the algebraic error (4.2.5) after k iterations of the smoother S_ℓ:

$$\mathbf{e}_\ell^{(k)} = S_\ell^k \mathbf{e}_\ell^{(0)} .$$

By definition, $\mathbf{e}_\ell^{(k)}$ is smooth if it is not significantly different to $\mathbf{e}_\ell^{(k-1)}$, i.e.

$$\mathbf{e}_\ell^{(k)} \approx S_\ell \mathbf{e}_\ell^{(k-1)} .$$

Thus for an algebraically smooth error, $\mathbf{e}_\ell$, the iteration satisfies $\mathbf{e}_\ell \approx S_\ell \mathbf{e}_\ell$, which is equivalent to $D^{-1}A_\ell \mathbf{e}_\ell \approx 0$ for the weighted Jacobi iteration $S_\ell = I - \omega D^{-1} A_\ell$. This leads to the condition

$$A_\ell \mathbf{e}_\ell \approx \mathbf{0} . \tag{4.7.1}$$

We appeal to the errors satisfying (4.7.1) as algebraically smooth error. Note that such an error may not be geometrically smooth (see [71, Section 1.3]). Although we have used the weighted Jacobi method to lead to condition (4.7.1), similar analysis can be performed for the Gauss–Seidel method which is more commonly used in AMG.

An implication of (4.7.1) is that $< D^{-1}\mathbf{r}, \mathbf{r} > \ll < \mathbf{e}, \mathbf{r} >$, and we can show from this that

$$\sum_{j \neq i} \left(\frac{|a_{ij}|}{a_{ii}} \right) \left(\frac{e_i - e_j}{e_i} \right) \ll 2, \quad 1 \leq i \leq n . \tag{4.7.2}$$

The product on the left-hand-side must be very small, which implies that if $\frac{|a_{ij}|}{a_{ii}}$ is relatively large, then $e_i \approx e_j$. In other words, the error varies slowly from variables i to j if $|a_{ij}|$ is large. This heuristic idea is one of the main concepts used for defining the interpolation operator.

4.7.2 Strong Coupling

Aside from algebraic smoothness, a second important concept in AMG is that of *strong dependence* or *strong coupling*. Noting that we associate the i^{th} equation of A_ℓ with

the i^{th} unknown u_i, we define strong coupling as follows:

Definition 4.7.1. A variable u_i is strongly coupled to another variable u_j if

$$-a_{ij} \geq \theta \max_{k \neq i} \{-a_{ik}\},$$

for some $0 < \theta \leq 1$.

In other words, if a_{ij} $(i \neq j)$ is large relative to the other coefficients a_{ik}, then u_i is strongly coupled with u_j.

With the two key concepts of algebraic smoothness and strong coupling defined, we can now define the multigrid components for AMG.

4.7.3 Selecting the Coarse Grid

When selecting the coarse grids, we rely on the twin concepts of algebraic smoothness and strong coupling. We also assume vertex centred grids, so the coarse grid nodes will be s subset of the fine grid nodes. Suppose we split the set of variables on the fine grid as a set C of coarse grid variables and a set F of variables *only* on the fine grid, i.e. we have a partitioning of the indices $\{1, \ldots, n\} = C \cup F$. Let us denote

$$\begin{aligned} N_i &= \{j : j \neq i \text{ and } a_{ij} \neq 0\}, \text{ the neighbours of } i. \\ S_i &= \{j \in N_i : i \text{ is strongly dependant on } j\}, \\ S_i^T &= \{j \in N_i : j \text{ is strongly dependant on } i\}. \end{aligned}$$

The coarse grid selection algorithm splits the fine grid variables into sets of fine grid only (F) variables and coarse grid (C) variables using the information in the strong coupling set S_i. The selection process is based on the following heuristics:

1. For each $i \in F$, a point $j \in S_i$ that is strongly coupled with i should be a coarse grid point or strongly coupled with at least one coarse grid point that is strongly coupled with i.

2. No coarse grid point should be strongly coupled with another coarse grid point.

Satisfying both these heuristics is not always possible, and the second heuristic is used to control the size of the coarse grid. The larger the coarse grid, the more accurate the interpolation of the smooth errors onto the fine grid, but this also reduces the coarsening factor, defined in Remark 4.7.2 below. Thus the second heuristic is used to maximize the number of coarse grid points without any two such points being strongly coupled.

The coarse grid selection starts by choosing some variable i to become a C-variable. Then all variables j that are strongly coupled with i become F-variables. From the remaining variables, one is chosen as a C-variable and all others strongly coupled to it become F-variables. We repeat this process until all the variables have been taken care of.

However, we would like to avoid randomly distributed patches of C/F-variables, thus the selection process must be performed in a certain order. Let U be the set of undecided variables, where initially $U = \{1, 2, \ldots, n\}$. Then we introduce a *measure of importance*, λ_i, of any $i \in U$ to become the next C-variable, as follows:

$$\lambda_i = |S_i^T \cap U| + 2|S_i^T \cap F|\,, \qquad i \in U\,,$$

where $|X|$ denotes the number of elements in set X. λ_i is a measure of how important i would be as a C-variable. The variable $i \in U$ with the highest measure will be selected as a C-variable, and all strongly coupled points of i become F-variables. We then update U, calculate λ_i for each $i \in U$ and select another C-variable to repeat the process again until $U = \phi$. Note that there are further heuristics to deal with special cases and to improve the quality of the coarse grid (e.g. avoiding isolated nodes).

Once the coarse grid has been selected, we use the Galerkin method to calculate the coarse grid operator:

$$A_{\ell-1} = R_\ell A_\ell P_\ell\,.$$

Remark 4.7.2. The coarsening factor, cf_ℓ, between levels $\ell - 1$ and ℓ is defined as the ratio of the number of nodes at these levels, i.e.

$$cf_\ell = \frac{\#\text{ nodes on level } \ell}{\#\text{ nodes on level } \ell - 1}\,, \quad \forall \ell \geq 2$$

The greater the coarsening factor, the less work is required on the coarser level.

4.7.4 The Interpolation Operator

The final ingredient in AMG is an operator-dependant interpolation operator. To construct this, we split the set of variables on the fine grid as a set C of coarse grid variables and a set F of variables *only* on the fine grid. Now, for each $i \in F$, we aim to define the interpolation weights ω_{ik} in

$$e_i = \sum_{k \in C} w_{ik} e_k, \qquad \forall i \in F \tag{4.7.3}$$

such that the interpolation operator $P : C \to C \cup F$ is such that (4.7.3) is a good approximation to any algebraically smooth error on the coarse grid. Let us recall N_i from Section 4.7.3 and denote

$$C_i = \{j \in C \cap N_i : j \text{ strongly influences } i\}, \text{ the coarse interpolatory set for } i\,,$$
$$D_i = \{j \in F \cap N_i : j \text{ strongly influences } i\}, \text{ the strongly connected neighbours of } i\,,$$
$$Q_i = \{j \in F \cap N_i : j \text{ weakly influences } i\}, \text{ the weakly connected neighbours of } i\,.$$

Assuming vertex centred interpolation is used, we define the interpolation operator by

$$(P_\ell \mathbf{e})_i = \begin{cases} e_i & \text{if } i \in C \\ \sum_{j \in N_i} \omega_{ij} e_j & \text{if } i \in F \end{cases}, \tag{4.7.4}$$

for some weights ω_{ij}. By the algebraic smoothness of the error, we can write (4.7.1) as

$$a_{ii} e_i \approx - \sum_{j \in N_i} a_{ij} e_j\,,$$

or

$$a_{ii} e_i \approx - \sum_{j \in C_i} a_{ij} e_j - \sum_{j \in D_i} a_{ij} e_j - \sum_{j \in Q_i} a_{ij} e_j\,.$$

In order to find the weights ω_{ij}, we must replace e_j in the second and third sums with e_i or e_j, for $j \in C_i$. We do this by considering (4.7.2), where we deduce that the algebraically smooth error varies slowly in the direction of strong coupling. This also implies that algebraically smooth error can be well approximated by a weighted average of its strongly coupled neighbours.

Now, if $j \in Q_i$, then e_i doesn't depend strongly on e_j and so by (4.7.2), $|a_{ij}|$ must be small. Therefore the sum $\sum_{j \in Q_i} a_{ij} e_j$ will also be small, and so can be replaced with $\sum_{j \in Q_i} a_{ij} e_i$ with a relatively insignificant error, giving

$$\left(a_{ii} + \sum_{j \in Q_i} a_{ij} \right) e_i \approx - \sum_{j \in C_i} a_{ij} e_j - \sum_{j \in D_i} a_{ij} e_j\,. \tag{4.7.5}$$

If $j \in D_i$, then there is a strong coupling between e_i and e_j, and so e_j can be approximated accurately by a weighted sum of the values e_k from the coarse interpolatory set of the point i, i.e.

$$e_j = \frac{\sum_{k \in C_i} a_{jk} e_k}{\sum_{k \in C_i} a_{jk}}\,. \tag{4.7.6}$$

Combining (4.7.5) with (4.7.6), we find that

$$e_i = -\frac{\sum_{j\in C_i} a_{ij}e_j + \sum_{j\in D_i} a_{ij}\frac{\sum_{k\in C_i} a_{jk}e_k}{\sum_{k\in C_i} a_{jk}}}{a_{ii} + \sum_{j\in Q_i} a_{ij}},$$

and so the weights from (4.7.4) are

$$\omega_{ij} = -\frac{a_{ij} + \sum_{m\in D_i}\left(\frac{a_{im}a_{mj}}{\sum_{k\in C_i} a_{mk}}\right)}{a_{ii} + \sum_{n\in Q_i} a_{in}}.$$

Again, there are further heuristics, for example to deal with positive off-diagonal entries.

As for the restriction operator, we simply take the transpose of the interpolation, i.e.

$$R_\ell = P_\ell^T.$$

The idea of AMG was devised by Brandt, McCormick and Ruge [18]. AMG methods are very popular because of their applicability to complex industrial problems on unstructured grids and more recently because there has been a major surge of interest in solving increasingly larger systems of billions of unknowns with variable coefficients. Many practical implementations of AMG exist, such as the `AMG1R5` by Ruge and Stüben [63] and `BoomerAMG` by Henson and Yang [46], a very robust and efficient parallel implementation of AMG within the `hypre` library [37]. However, for certain types of problems where further information is available, it is possible to take the ideas from AMG and use them within a geometric approach instead. This will eliminate the need for a large setup cost whilst giving the robustness of an AMG code. Experiments have been devoted to testing such ideas in several application fields (e.g. [90]), but to the best of my knowledge not for problems posed on spherical polar grids that are of interested for this thesis. Such problems will have degenerating coefficients towards the poles, causing large anisotropies, and in Chapter 5 we will give a detailed account how they can be treated effectively using the ideas from AMG.

Chapter 5

Non-Uniform Multigrid for Spherical Polar Grids

In this chapter we introduce a novel conditional coarsening strategy within a geometric multigrid method for elliptic problems in tensor product form, and apply it to problems in spherical geometries that are of interest in numerical weather prediction. It is well known that simple isotropic problems can be solved optimally using geometric multigrid methods with full coarsening and point-wise smoothers, as described in Chapter 4 and several books and articles, e.g. [20, 75]. We also showed in Chapter 4 that the optimal convergence of this method can been proven both experimentally and theoretically (cf. [44, 88]), and the theory extends to the V-cycle (see [16]), which is the method used for all experiments in this thesis. This standard method, however, is not robust for problems with anisotropy, such as the ones studied in this thesis, but there are well-known remedies in certain situations. For example, if the anisotropy is grid–aligned, then there are two standard ways of retaining optimality: (a) semi-coarsening and (b) line or plane relaxation. For problems in spherical geometries, geometric multigrid methods with line and plane smoothers, and "uniform" semi-coarsening, have already been studied in [7], and theoretical results for line smoothers and semi-coarsening can be found in [17, 58].

In this chapter, these existing methods are adapted to deal with the particular "inhomogeneous" anisotropies arising from spherical polar grids, by using a "conditional" semi-coarsening strategy. Conceptually, this is a simple idea which uses semi coarsening in the anisotropic regions near the poles and full coarsening in the isotropic regions near the equator. We demonstrate numerically that this strategy gives rise to an optimal method for solving elliptic problems on the sphere. We have not yet succeeded in fully proving this robustness, but we will also make some theoretical considerations

that strongly support our numerical findings.

Theoretical results for planar polar coordinates can be found in [14], but only with line smoothers and uniform semi-coarsening. The idea of conditional semi-coarsening in the latitudinal direction proposed here has only been explored for edge and corner singularities so far (cf. [40, 55, 90]) but not for spherical polar grids (even in two dimensions). It is inspired by algebraic multigrid (AMG) ideas (see e.g. [20, 71]). AMG methods are fully automatic and only based on algebraic information in the matrix A. Coarse grid unknowns are chosen based on the relative size of the off-diagonal entries in the matrix which in the application here will lead to very similar coarse grids to our conditionally semi-coarsened grids. However, AMG methods are known to require a large setup cost to design these coarse grids and the operator-dependent interpolation and restriction operators, especially in three dimensions (3D). Our geometric method on the other hand, requires almost no setup cost to obtain the same robustness, which is why it is expected to comfortably outperform AMG methods in terms of CPU-time.

The chapter is organized as follows. We motivate the type of problems we wish to study in Section 5.1. The two dimensional (2D) model problem in this section will be solved on the unit square and will have particular anisotropies similar to those arising from spherical polar grids. We use traditional multigrid methods to solve this model problem, focusing particularly on the approach of line relaxation in one direction together with semi coarsening in the other, a method whose convergence rate has been proved analytically in [14]. In Section 5.2, we introduce the new approach of conditional semi-coarsening with point relaxation on this model problem, which aims to beat all the existing multigrid methods, including algebraic multigrid (AMG) methods, by exploiting the structure of the anisotropies. We give a heuristic argument to justify that this method performs optimally, and back this up with numerical results. The model problem is extended to 3D in Section 5.3, where we discuss the techniques for tackling additional anisotropies in the third dimension, and give some numerical results. We then give theoretical results in Section 5.4. We begin by following the 2D tensor product proof from [14] (see also [13, §2]) to show that a general 2D elliptic problem with grid-aligned anisotropy can be solved optimally with the multigrid V-cycle using line smoothers and semi coarsening. This theory will cover the 2D model problem from Section 5.1. We then extend the theory to prove that general anisotropic 3D elliptic problems, including the model problem from Section 5.3, can be solved optimally using a line smoother and no coarsening in the third coordinate direction. Finally in Section 5.5, we apply the techniques developed in this chapter to 2D and 3D problems in spherical polar coordinates which are of a particular interest to the Met Office, noting that particular issues must be accounted for at the poles. Numerical results will be

given, and comparisons will be made with methods such as AMG and a particular variant of the preconditioned generalized conjugate residual (GCR) Krylov subspace method [34] used operationally at the Met Office.

5.1 Motivation by Studying Anisotropic Problems on the Unit Square

Let us first consider two dimensions only, where problems with grid aligned anisotropy can be written in the form

$$-\nabla \cdot \left(\begin{pmatrix} \alpha_1(x,y) & 0 \\ 0 & \alpha_2(x,y) \end{pmatrix} \nabla u \right) = g, \tag{5.1.1}$$

with $\alpha_1 = \alpha_1^1(x)\alpha_1^2(y)$, $\alpha_2 = \alpha_2^1(x)\alpha_2^2(y)$ uniformly positive almost everywhere. The simplest model problem with grid aligned anisotropy is $\alpha_1 \equiv \delta \ll 1$ constant and $\alpha_2 \equiv 1$. For this model problem, it is shown theoretically in [70] that x-line relaxation with full coarsening leads to an optimal multigrid convergence. In the more general case of varying coefficients α_1, α_2, i.e. when the anisotropy is inhomogeneous, it seems that it is necessary to combine line relaxation with semi-coarsening to still get an optimal method at least theoretically (cf. [14]). The methods outlined in [14] appear to be most popular for dealing with inhomogeneous grid aligned anisotropy despite the larger cost of line relaxation (compared with point relaxation) resulting from solving tridiagonal systems of equations at each relaxation sweep (cf. Chapter 4).

We use the following model problem in this section to analyze the effects of inhomogeneous grid-aligned anisotropy on the convergence of existing multigrid methods:

$$-\frac{\partial^2 u}{\partial x^2} - \frac{\partial}{\partial y}\left(\sin^2(\pi y)\frac{\partial u}{\partial y}\right) = g \quad \text{on} \quad \Omega = (0,1)\times(\varepsilon, 1-\varepsilon), \quad 0 < \varepsilon \ll 1 \tag{5.1.2}$$

subject to the Dirichlet boundary condition $u = 0$ on $\partial\Omega$. This is clearly of the type (5.1.1), with $\alpha_1(x,y) = 1$ and $\alpha_2(x,y) = \sin^2(\pi y)$ and for $\varepsilon > 0$ the coefficient is uniformly positive. The finite volume discretisation of problem (5.1.2) on the unit square yields a matrix with the following stencil at the interior nodes:

$$\begin{bmatrix} & -\frac{h_x}{h_y}\sin^2(\pi y_{j+\frac{1}{2}}) & \\ -\frac{h_y}{h_x} & -\sum & -\frac{h_y}{h_x} \\ & -\frac{h_x}{h_y}\sin^2(\pi y_{j-\frac{1}{2}}) & \end{bmatrix}. \tag{5.1.3}$$

where subscript j denotes the mesh line in the y-direction. We observe from this stencil

that the strength of the anisotropy varies only in the y-direction since $\sin^2(\pi y)$ is the only variable in the stencil. We observe a strong anisotropy for $y \to \varepsilon$ and $y \to 1-\varepsilon$ where $\sin^2(\pi y) \to 0$. For $\varepsilon \ll 1$ the singular behaviour of the differential operator close to $y = 0$ and $y = 1$ becomes more pronounced, and the anisotropy becomes stronger. We test the traditional and existing multigrid methods discussed in chapter 4 on this model problem to determine which methods deal with inhomogeneous anisotropy most effectively.

5.1.1 Standard Geometric Multigrid Approaches

Firstly, we test full coarsening in conjunction with point relaxation (FCPR), which is expected to perform badly for solving (5.1.2) since we saw in Chapter 4 that it cannot even deal with simpler anisotropies. Similarly, semi coarsening with point relaxation (SCPR) is expected to be suboptimal for problems with inhomogeneous anisotropy. Line relaxation, however, is known to be very effective for dealing with anisotropy as long as the direction of the weak coupling does not change throughout the domain. This is indeed the case for problem (5.1.2), so the performance of line relaxation with full coarsening is (FCLR) likely to be optimal, though the cost of line relaxation is significantly higher than point relaxation. Finally we test using line relaxation in conjunction with semi coarsening (SCLR), a more robust method whose convergence rate can be proved theoretically (cf. [14]).

For each of the methods FCPR, SCPR, FCLR and SCLR, the model problem (5.1.2) is solved using the standard multigrid V-cycle as a stand alone solver. Note, however, that it is also common to use multigrid as a preconditioner to a Krylov subspace method, with one V-cycle as the preconditioner. Results using multigrid as a stand alone solver or as a preconditioner typically give similar results. As in Chapter 4, we denote the sequence of grids as $\mathcal{T}_\ell$, $\ell = 1, \ldots, F$, where $\mathcal{T}_1$ and $\mathcal{T}_F$ are the coarsest and finest grids respectively, and the operator on grid $\mathcal{T}_\ell$ as A_ℓ. The main problem we are solving is on the finest grid $\mathcal{T}_F$, and we denote this as

$$A_F \mathbf{u}_F = \mathbf{b}_F.$$

The matrices A_ℓ are each computed by the finite volume discretisation of PDE (5.5.1) on the sequence of grids. The interpolation operator, P_ℓ is a mapping from $\mathcal{T}_{\ell-1}$ to $\mathcal{T}_\ell$, and the restriction operator, R_ℓ is a mapping from $\mathcal{T}_\ell$ to $\mathcal{T}_{\ell-1}$. We use linear interpolation and full weighting restriction, which means the restriction matrix is the

transpose of the interpolation matrix, i.e.

$$P_\ell = R_\ell^T, \qquad \forall \ell = 2, \ldots, F.$$

The ratio of the fine grid cells to the coarse grid cells is known as the *coarsening factor* (cf. Remark 4.7.2). The coarsening factor varies depending on the coarsening method used, but the higher the coarsening factor the fewer grid cells on the coarse grids and so fewer computations are required on the coarse levels. We choose the right-hand-side, $\mathbf{b}_F$ by choosing a random solution $\mathbf{u}_F$ and pre-multiplying it by A_F. This ensures that the exact solution is known which can be compared with the approximate solution using multigrid, as well as ensuring that the right-hand-side is in the range of A_F (if A_F is a singular matrix). The structure of the algorithm for the multigrid V-cycle method is unchanged from algorithm 4.5 given in Chapter 4. Depending on the method used, suitable changes will be made to the individual components of the algorithm, such as the choice of smoother and coarsening strategy.

All tests in this section are carried out on a single 1.8GHz processor of a Dual dual-core 64bit AMD Opteron 2210 (Cache size 1.0MB and 2GB memory) using the Fortran95 compiler `ifort`. The initial guess for the iteration is taken to be zero, with a relative residual reduction of $tol = 10^{-8}$ as the stopping criterion. The number of pre- and post-smoothing steps will be set to $\nu_1 = 3$ and $\nu_2 = 2$, but tests will also be carried out for other values, especially for comparisons with the algebraic multigrid (AMG) methods in later sections. On $\mathcal{T}_1$, we use the conjugate gradient (CG) method [74] to solve the coarse grid problem. Numerical experiments showed that the coarse grid problem only needs to be solved to a residual tolerance of 0.1, and any greater accuracy made no difference to the total number of V-cycle iterations required. Finally we set $\varepsilon = 0$ in (5.1.2). Note that this means we no longer have uniform ellipticity of the operator as we allow $y = 0, 1$, but this does not make a difference from an algorithmic point of view since none of the degrees of freedom are located at these points.

The performance of each of these methods for solving (5.1.2) with $\varepsilon = 0$ is given in Tables 5.1, 5.2 and 5.3. Table 5.1 measures the number of iterations, N_{its}, of each method with respect to problem size, and for an optimal method N_{its} will be robust to problem size. Table 5.2 measures the total CPU time taken for each method. Note that we normally give the total CPU time as a breakdown of the setup time and solve time, which is is done particularly for results concerning AMG problems where the setup time makes up a significant fraction of the total time. However, for the problems discussed here, the setup time is small and typically less than 25% of the solve time, so we do not give a breakdown of the total CPU time. For an optimal method, the CPU

time will grow linearly with problem size. Finally, Table 5.3 measures the geometric average of the V-cycle convergence factor, defined in [62], i.e.

$$\mu_{\text{avg}} \;=\; \left(\|\mathbf{r}_F^{(N_{\text{its}})}\|/\|\mathbf{r}_F^{(1)}\|\right)^{1/(N_{\text{its}}-1)} .$$

Note that the first cycle is excluded, and this is a common indicator of the convergence factor. If the number of iterations is robust with respect to problem size, then the same will apply to μ_{avg}.

Full Coarsening with Point Relaxation (FCPR)

FCPR performes optimally for isotropic problems on a regular grid, as described in Chapter 4. It is a very fast method because the work done on the coarse grids is small due to a high coarsening factor (cf. Remark 4.7.2), where the number of nodes on the coarse grid is halved in both directions. FCPR is fast also because the cost of the point smoother is small, being proportional to the number of unknowns (Note that the cost of line relaxation is also proportional to the number of unknowns, but the constant of proportionality is much higher). However, for problems even with a small uniform anisotropy, FCPR will not perform optimally, and so the number of iterations is expected to rise with respect to the problem size. Table 5.1 clearly confirms this, as the number of iterations increases rapidly with problem size. Consequently the CPU time also grows more than linearly with respect to problem size, and μ_{avg} is not robust with respect to problem size, as shown in Tables 5.2 and 5.3. FCPR is therefore not optimal.

Semi Coarsening with Point Relaxation (SCPR)

SCPR imposes coarsening only in the direction of the strong coupling, i.e. the x-direction for problem (5.1.2), and as a result the coarsening factor is only 2 instead of 4 as in the case of FCPR. As we saw in Chapter 4, semi coarsening will reduce the strength of anisotropy and optimally solve grid aligned and uniformly anisotropic problems. Conversely, semi coarsening will not be optimal if used on isotropic problems because it will *introduce* anisotropy to the problem. Near the region at which $y = 0.5$, problem (5.1.2) is almost isotropic, hence SCPR will not be an optimal method for this problem. The growing number of iterations with respect to problem size in Table 5.1 confirms this.

Full Coarsening with Line Relaxation (FCLR)

Here, a standard multigrid V-cycle with a linewise Gauss−Seidel smoother and full coarsening is used. The linewise smoother will solve a tridiagonal system for each line of constant y using the Thomas algorithm [50, Chapter 9]. The cost of the Thomas algorithm is proportional to the number of unknowns, but with a larger constant than the point smoother.

In Chapter 4, it was shown that FCPR will be optimal for solving anisotropic problems provided that line relaxation was imposed in the direction of the strong coupling, and if the direction of strong coupling doesn't change at different points on the domain. Tables 5.1 – 5.3 demonstrate that FCLR indeed performs optimally, with the number of iterations and average convergence factors unaffected by problem size, and the CPU time increasing linearly with problem size.

Semi Coarsening with Line Relaxation (SCLR)

Finally, we test line relaxation with semi coarsening, where we coarsen only in one direction and apply line relaxation in the other. Since line relaxation with full coarsening works very efficiently, line relaxation with semi coarsening is only likely to slow down the method due extra work required on the coarse grids as a result of the smaller coarsening factor. However, the advantage of this approach is that it makes no assumptions about the direction of the strong or weak coupling, and is optimal even when the direction of the weak coupling switches in different regions of the domain. This is a particularly important factor that needs to be taken into account for the 3D problems, as we will see later in this chapter.

In addition, tests using this method were carried out in order to confirm the theory from [14]. In that paper, a V-cycle convergence proof using line relaxation with semi coarsening for a grid-aligned inhomogeneous anisotropic problem is given. We outline their proof for model problem (5.1.2) in Section 5.4.1 and extend it to 3D in Section 5.4.2.

Problem size	No. of coarse grids	FCPR	SCPR	FCLR	SCLR
32x32	2	65	8	5	5
64x64	3	203	11	5	4
128x128	4	631	17	5	4
256x256	5	1828	50	5	4
512x512	6	5309	139	5	4

Table 5.1: Number of iterations, N_{its}, required to solve (5.1.2).

Problem size	No. of coarse grids	FCPR	SCPR	FCLR	SCLR
32x32	2	3.04E-2	8.26E-3	6.57E-3	8.76E-3
64x64	3	3.40E-1	3.93E-2	2.56E-2	3.19E-2
128x128	4	4.94	2.41E-1	1.08E-1	1.40E-1
256x256	5	66.13	5.68	4.59E-1	5.94E-1
512x512	6	940.2	74.1	1.95	2.59

Table 5.2: Total CPU time taken (including setup time) to solve (5.1.2).

Problem size	No. of coarse grids	FCPR	SCPR	FCLR	SCLR
32x32	2	0.796	0.102	0.030	0.029
64x64	3	0.931	0.213	0.034	0.029
128x128	4	0.978	0.405	0.032	0.030
256x256	5	0.992	0.742	0.030	0.030
512x512	6	0.997	0.900	0.028	0.030

Table 5.3: Average convergence factor, μ_{avg}, when solving (5.1.2).

The theory in [14] suggests it may be necessary to combine semi-coarsening with line relaxation to prove the uniform convergence of general elliptic problems of type (5.1.1). The numerical experiments using this method on (5.1.2), given in Tables 5.1 – 5.3, have shown that this method is indeed robust with respect to problem size, thus confirming the theory. However, as we have already seen with FCLR, other more efficient variants of multigrid exist. Thus it is not absolutely necessary (from an experimental point of view) to combine line relaxation with semi coarsening for an optimal solver to grid aligned anisotropic problems, and so we investigate a new multigrid strategy in Section 5.2.

5.1.2 Algebraic Multigrid (AMG) for Anisotropic Problems

Before we do this let us study another very popular multigrid approach which has been experimentally proven in several articles (e.g. [20, 71, 63]) to be efficient in cases of strong anisotropies (even extending to anisotropies that are not grid aligned), namely AMG. As discussed in Section 4.7, AMG methods are highly successful as they can be used in any complex geometric situations which are out of reach of geometric multigrid methods. The versatility and robustness of AMG methods are a result of using matrix-dependent prolongation operators, particularly for problems with large (non-smooth) coefficient variation. AMG methods are normally robust with respect to problem size and solve almost all elliptic problems nearly optimally.

There are several versions of AMG, and we test its performance on problem (5.1.2) using the popular `BoomerAMG` implementation from the `hypre` library [46, 37]. We

use the default settings for `BoomerAMG`, i.e. a symmetric-SOR smoother (with 1 pre- and post-smoothing step), Falgout coarsening, classical Ruge-Stüben interpolation and Gaussian Elimination as the coarse grid solver. Some experiments with other settings were also carried out, but did not lead to a significant improvement of the method.

Thus far in this chapter, we have only used multigrid as a stand alone iterative solver. Thus we test the AMG method as a stand-alone solver for a direct comparison with the geometric multigrid approaches from the previous section, but also as a preconditioner to the conjugate gradient (CG) method with one V-cycle per iteration. In Table 5.4, the notation CG + AMG means the CG algorithm is preconditioned with one V-cycle of multigrid, with the number of CG iterations measured. The notation V-cycle only means the multigrid V-cycle is used as a stand-alone solver, with the number of V-cycle iterations measured.

We observe from Table 5.4 that CG + AMG is robust with respect to grid refinement. However, the solve time of CG + AMG is nearly twice as large as FCLR. As for the setup time, FCLR is approximately four times as fast as CG + AMG, and in total FCLR is faster than CG + AMG by a factor of approximately 2. Note that when FCLR is used as a preconditioner to CG (we call this CG + FCLR), then the results are virtually the same as FCLR, hence CG + FCLR also performes approximately 2 times faster than CG + AMG. When used as a stand-alone solver, AMG is not fully robust. The number of iterations and the average convergence factor per iteration grow slightly with the problem size, and the number of iterations required is always greater than CG + AMG. This clearly affects the CPU time and it is consequently slower than CG + AMG.

		`BoomerAMG` (V-cycle only)			CG + AMG		
Problem size	Setup time	Solve time	N_{its}	μ_{avg}	Solve time	N_{its}	μ_{avg}
32x32	5.54E-3	1.29E-2	8	0.091	1.12E-3	5	0.020
64x64	1.73E-2	4.70E-2	8	0.090	4.03E-2	6	0.030
128x128	7.47E-2	2.23E-1	10	0.151	1.71E-1	6	0.044
256x256	3.12E-1	1.07	11	0.172	8.29E-1	7	0.054
512x512	1.27	4.90	12	0.209	3.52	7	0.067

Table 5.4: The `BoomerAMG` method applied to model problem (5.1.2) as a stand alone solver and a preconditioner for the CG method. CPU times in seconds.

Thus, although AMG can be used for many complex problems, the flexibility of AMG comes at a price: its setup cost. The selection of coarse nodes, the construction of interpolation operators and the construction of coarse level operators is slower for AMG than for geometric methods (especially in 3D), since everything has to be deduced algebraically from the system matrix A via graph theoretical techniques. Also, the

coarse grid operators generally become very dense and expensive to apply. Therefore, AMG is usually less efficient than geometric multigrid on problems for which geometric multigrid can be suitably adapted.

5.2 New Approach – Conditional Semi-Coarsening

Model problem (5.1.2) is highly anisotropic with degenerate coefficients, but as discussed in Section 5.1 the anisotropy is grid aligned and it can be dealt with efficiently and robustly using geometric approaches. Line relaxation with full coarsening clearly performs very well, but can certain components of the multigrid method be combined differently to yield an even more efficient method? In contrast to standard geometric multigrid approaches, which adapt the smoother and keep the structure of the coarse grid simple, AMG methods adapt the coarse grid so that the smoother is kept simple (ie. a point smoother). Such techniques used in AMG methods motivate the key idea of this chapter – a "conditional semi-coarsening" strategy. We aim to show that in practice line relaxation is not necessary for problems of type (5.1.2), provided an alternative coarsening strategy is used to exploit the particular structure of the grid anisotropy.

Recall the stencil (5.1.3) for problem (5.1.2):

$$\begin{bmatrix} & -\frac{h_x}{h_y}\sin^2(\pi y_{j+\frac{1}{2}}) & \\ -\frac{h_y}{h_x} & -\sum & -\frac{h_y}{h_x} \\ & -\frac{h_x}{h_y}\sin^2(\pi y_{j-\frac{1}{2}}) & \end{bmatrix}.$$

at an interior point $\{(i,j,k) : i = 1,\ldots,n_x,\ j = 1,\ldots,n_y,\ k = 1,\ldots,n_z\}$ with suitable modifications at the boundary. n_x, n_y and n_z denote the number of grid points in the x-, y- and z-directions respectively. Since $y \in [\varepsilon, 1-\varepsilon]$, we observe a strong anisotropy for $y \to \varepsilon$ and $y \to 1-\varepsilon$ caused by the degeneracy of the coefficients of the differential operator, where $\sin(\pi y) \to 0$. Thus the entries in the x-direction are significantly larger than the entries in the y-direction at these regions, suggesting that semi-coarsening would be effective. However, semi-coarsening is only effective for strongly anisotropic problems, and will introduce anisotropy if the original problem is isotropic. Near $y = 0.5$, the problem is close to isotropic (as $\sin(\pi y) \approx 1$), suggesting that it would be better to use full coarsening there instead of semi-coarsening. This motivates the key idea of conditional semi-coarsening, i.e. to perform full uniform coarsening near $y = 0.5$ and semi-coarsening (in x-direction only) near the y-boundaries. More specifically, we compare the ratio of the x and y off-diagonal entries in stencil (5.1.3) at each y-line. On line j, the x off-diagonal entries are $-\frac{h_y}{h_x}$ and the two y off-

diagonal entries are $-\frac{h_x}{h_y}\left(\sin^2(\pi y_{j-\frac{1}{2}})\right)$ and $-\frac{h_x}{h_y}\left(\sin^2(\pi y_{j+\frac{1}{2}})\right)$, so we take the average which is approximately $-\frac{h_x}{h_y}\left(\sin^2(\pi y_j)\right)$. We fully coarsen that line only if the ratio is sufficiently close to 1. We observe from (5.1.3) that on the fine grid with a uniform mesh, ie. $h_x \approx h_y$, the ratio is approximately $\sin^2(\pi y_j)$. In the actual computations, since $0 \leq \sin^2(\pi\phi_j) \leq 1$, we fully coarsen only if $\sin^2(\pi y_j)$ is greater than $\frac{1}{2}$ which in numerical experiments proved to be the optimal value. Note the anisotropy only varies in the y-direction, ie. there is no value in the stencil that is a function of i, so the grid is always uniformly fully coarsened in the x-direction.

Now on subsequent coarse grids we no longer have a uniform mesh. Additionally the mesh width and number of grid points must be distinguished on each grid level, thus we introduce a small change in notation to account for this. The mesh width in the y-direction varies as a function of j on the coarse grids, so we denote the mesh width in the y-direction at the j-th mesh line on $\mathcal{T}_\ell$ by $h^{(\ell)}_{y,j}$. Similarly, $h^{(\ell)}_x$ denotes the mesh width in the x-direction on $\mathcal{T}_\ell$, which is constant on each grid, and the stencil for the finite volume discretisation of (5.1.2) on the coarse grids becomes

$$\begin{bmatrix} & -\frac{h^{(\ell)}_x}{h^{(\ell)}_{y,j}}\sin^2(\pi y_{j+\frac{1}{2}}) & \\ -\frac{h^{(\ell)}_{y,j}}{h^{(\ell)}_x} & -\sum & -\frac{h^{(\ell)}_{y,j}}{h^{(\ell)}_x} \\ & -\frac{h^{(\ell)}_x}{h^{(\ell)}_{y,j}}\sin(\pi y_{j-\frac{1}{2}}) & \end{bmatrix}.$$

at a point $\{(i,j,k) : i = 1,\ldots,n^{(\ell)}_x,\ j = 1,\ldots,n^{(\ell)}_y,\ k = 1,\ldots,n^{(\ell)}_z\}$, where $n^{(\ell)}_x$, $n^{(\ell)}_y$ and $n^{(\ell)}_z$ are the number of grid points in the x-, y- and z-directions respectively on grid $\mathcal{T}_\ell$. The ratio of the off-diagonal entries on line j is now compensated by a factor $(h^{(\ell)}_x/h^{(\ell)}_{y,j})^2$ which varies as a function of j. We call this strategy 'conditional semi-coarsening' and it is summarized as follows:

On grid $\mathcal{T}_\ell$:
x-direction – Double the mesh width ie. $h^{(\ell-1)}_x = 2h^{(\ell)}_x$
y-direction – Scan through each y-line (i.e. from $j = 1$ to $j = n^{(\ell)}_y$):
$ratio = (h^{(\ell)}_x/h^{(\ell)}_{y,j})^2 \sin^2(y_j)$
if $ratio < 0.5$ **then**
  semi coarsening: i.e. no coarsening on that y-line
else
  full coarsening: i.e. coarsen on that y-line
end if

The two key factors that make this strategy so simple and feasible are:

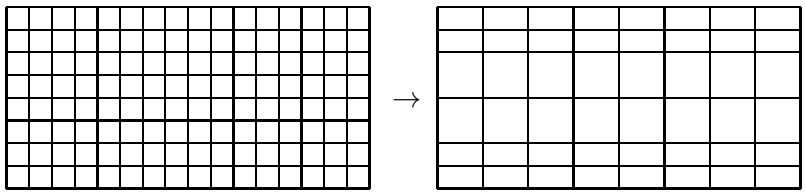

Figure 5-1: Conditional semi-coarsening on a 16×8 grid

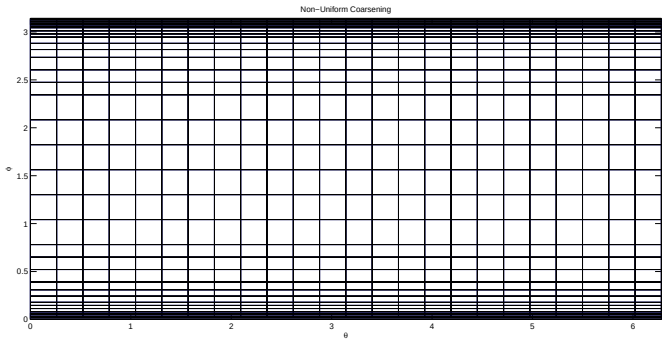

Figure 5-2: Conditional semi-coarsening on a 400×200 grid (after four refinements)

- The anisotropy is grid-aligned and changes only as a function of y. Thus it is sufficient to scan through the ratio of off diagonal entries at each y-line only.
- The direction of the strong coupling, i.e. the x-direction, doesn't change throughout the domain. If the y off diagonal entry was stronger at certain points in the domain, then additional strategies will be required to fully take care of all the anisotropies.

The result of this simple strategy is a non-uniform coarse grid, and Figure 5-1 shows the non-uniform coarsening strategy applied to a 16×8 uniform grid where half the region is fully coarsened and the other half semi coarsened. Repeating the coarsening strategy successively yields a graded mesh with coarse square cells near the equator and fine elongated cells near the poles, as shown in Figure 5-2 for a 400×200 uniform grid coarsened four times using conditional semi-coarsening. The region of full coarsening grows on successively coarser grids, which gives the coarse grids a graded appearance. The coarsening factor from the finest grid to the first coarse grid is approximately 3. The full algorithm for constructing the coarse grids using conditional semi-coarsening is given in Algorithm 5.1. The inputs of the routine are the number of grid points $n_x^{(\ell)}$ and $n_y^{(\ell)}$, the grid level ℓ, the mesh width $h_x^{(\ell)}$ and the vector of mesh widths

Algorithm 5.1 Conditional semi-coarsening for model problem (5.1.2):
`Conditional`$(h_x^{(\ell)}, h_y^{(\ell)}, n_x^{(\ell)}, n_y^{(\ell)}, \ell, h_x^{(\ell-1)}, h_x^{(\ell-1)}, n_x^{(\ell-1)}, n_y^{(\ell-1)})$

$incr = 1$ (incrementing up the y-line)
for $j = 1,\ n_y^{(\ell)}$
  $ratio = (h_x^{(\ell)}/h_{y,incr}^{(\ell)})^2 \sin^2(\pi y_{incr})$ (ratio of off-diag entries at line $incr$)
  if $ratio \geq 0.5$ **then**
    $h_{y,j}^{(\ell-1)} = h_{y,incr}^{(\ell)} + h_{y,incr+1}^{(\ell)}$ (mesh width doubled in y
    $incr = incr + 2$ at line $incr$ (i.e. full coarsening))
  else
    $h_{y,j}^{(\ell-1)} = h_{y,incr}^{(\ell)}$ (mesh width unchanged in y
    $incr = incr + 1$ at line $incr$ (i.e. semi coarsening))
  end if
  if $incr > n_y^{(\ell)}$ **exit**
end for
$h_x^{(\ell-1)} = 2h_x^{(\ell)}$ (mesh width doubled in x on grid $\mathcal{T}_{\ell-1}$)
$n_y^{(\ell-1)} = j$ (no. of grid points in x- and y-
$n_x^{(\ell-1)} = n_x^{(\ell)}/2$ direction on grid $\mathcal{T}_{\ell-1}$)

$\{(h_{y,j}^{(\ell)}),\ j = 1, \ldots, n_y^{(\ell)}\}$. The outputs of the routine are the number of grid points $n_x^{(\ell-1)}$ and $n_y^{(\ell-1)}$ on $\mathcal{T}_{\ell-1}$ and the mesh widths $h_x^{(\ell-1)}$ and $\{(h_{y,j}^{(\ell-1)}),\ j = 1, \ldots, n_y^{(\ell-1)}\}$.

This coarsening strategy has a significant effect on the transfer operators, where linear interpolation and full weighting restriction are used. When imposing full coarsening on a uniform mesh, the fine node is a quarter of the way along the distance between the coarse nodes (cf. Chapter 4.4.2 and Figure 5-3). However, when imposing full coarsening on a non-uniform mesh, this is not necessarily the case. Adjacent cells in the y-direction may not be the same size thanks to conditional semi-coarsening, so a fine node will a fraction t along the distance between two coarse nodes, where $0 < t < 1$. Let j be the mesh line of the coarse grid and k be the corresponding mesh line on the fine grid, then t is calculated at line k as follows:

$$t = \frac{h_{y,k}^{(\ell)}}{h_{y,j}^{(\ell-1)} + h_{y,j+1}^{(\ell-1)}}.$$

This value will change at each y-line because of the variable cell sizes caused by conditional semi-coarsening. However, in the x-direction, full coarsening is always imposed so the fine node will always be a quarter of the way along the distance between the coarse nodes. Note that $t = \frac{1}{4}$ when $h_{y,k}^{(\ell)} = \frac{1}{2}h_{y,j}^{(\ell-1)}\ \forall j, k$, i.e. for a uniform mesh.

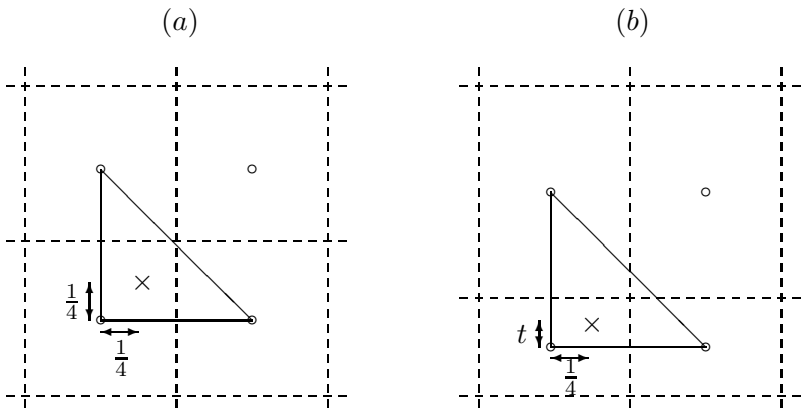

Figure 5-3: The weighting of values from adjacent nodes in (a) a uniform mesh, and (b) a non-uniform mesh.

Figure 5-3 demonstrates the differences between a uniform and non-uniform mesh. For a uniform mesh with full coarsening, recall from Section 4.4.2 the restriction matrix (4.4.3) in stencil notation for the finite volume discretisation in 2D:

$$\begin{bmatrix} & 0.25 & 0.25 & \\ 0.25 & 0.5 & 0.5 & 0.25 \\ 0.25 & 0.5 & 0.5 & 0.25 \\ & 0.25 & 0.25 & \end{bmatrix}.$$

With conditional semi-coarsening, however, the stencil takes the following form:

$$\begin{bmatrix} & t_k & t_k & \\ 0.25 & 1-t_k & 1-t_k & 0.25 \\ 0.25 & 1-t_k & 1-t_k & 0.25 \\ & t_k & t_k & \end{bmatrix}.$$

where we denote t_k as the ratio calculated on line k of the fine grid.

5.2.1 Heuristic Explanation of the Effectiveness

The motivation for using the non-uniform coarsening strategy is to make the problem on the coarser grids more isotropic, which is the same heuristic as semi coarsening for a problem which is uniformly anisotropic (cf. Section 4.5.1). The more isotropic the problem is on each coarser grid, the more effective the smoother will be in smoothing

the errors in all directions. Since we have already observed that the region of full coarsening increases on coarser grids, the problem on the coarse grids clearly becomes more isotropic. Looking at stencil (5.1.3), we observe that an isotropic problem is obtained if

$$-\frac{h_{y,j}^{(\ell)}}{h_x^{(\ell)}} \approx -\frac{h_x^{(\ell)}}{h_{y,j}^{(\ell)}} \sin^2(\pi y_j) \;\Leftrightarrow\; \frac{h_{y,j}^{(\ell)}}{h_x^{(\ell)}} \approx \sin(\pi y_j)\,, \tag{5.2.1}$$

where $h_x^{(\ell)}$ is constant on each grid whilst $h_{y,j}^{(\ell)}$ varies as a function of j on the coarser grids. Equality in (5.2.1) cannot be achieved on a uniform grid, thus we seek

$$\frac{h_{y,j}^{(\ell)}}{h_x^{(\ell)}} \to \sin(\pi y_j) \tag{5.2.2}$$

as the grid is coarsened. Near $y = \frac{1}{2}$, equality in (5.2.1) is achieved immediately since $\sin(\pi y_j) \approx 1$ and the aspect ratio $\frac{h_{y,j}}{h_x} \approx 1$ because of full coarsening in this region. However, $\sin(\pi y_j) \to 0$ as $y \to \varepsilon$ or $y \to 1-\varepsilon$, hence semi coarsening is required at these regions on successive grids to ensure that the factor $\frac{h_{y,j}^{(\ell)}}{h_x^{(\ell)}}$ is halved in size until it is of the same order of magnitude as $\sin(\pi y_j)$.

Figure 5-4 monitors the aspect ratio $\frac{h_{y,j}^{(\ell)}}{h_x^{(\ell)}}$ obtained by our algorithm on progressively coarser grids, with $\mathcal{T}_F$ having a grid resolution of 400×200. The stars represent the aspect ratio at each y-line, and it becomes clear that this ratio does indeed converge to $\sin(\pi y_j)$ as the grid is coarsened. After seven refinements, the stars are all very close to the curve of $\sin(\pi y_j)$ and this is still the case even when we zoom in to the bottom right hand corner of the graph. Hence our new coarsening strategy yields an isotropic problem on the coarser grids, which is a heuristic explanation of the optimal convergence. We confirm this claim with numerical tests in the next section. Unfortunately, we did not manage to turn these arguments into a rigorous proof yet but we believe that this should be possible, perhaps using Fourier analysis as in Appendix A but without uniform grids.

5.2.2 Numerical Experiments

The model problem (5.1.2) is solved with $\varepsilon = 0$ as in Section 5.1 using the standard multigrid V-cycle as a stand-alone solver with the pointwise Gauss−Seidel smoother combined with the conditional semi-coarsening described above. The coarsening factor from grid level to grid level is about 3. The sequence of matrices $A_\ell, \ell = 1, \ldots, F$, corresponding to PDE (5.1.2) are all computed by finite volume discretisation on each of the grids. We use linear interpolation and full weighting restriction to move between

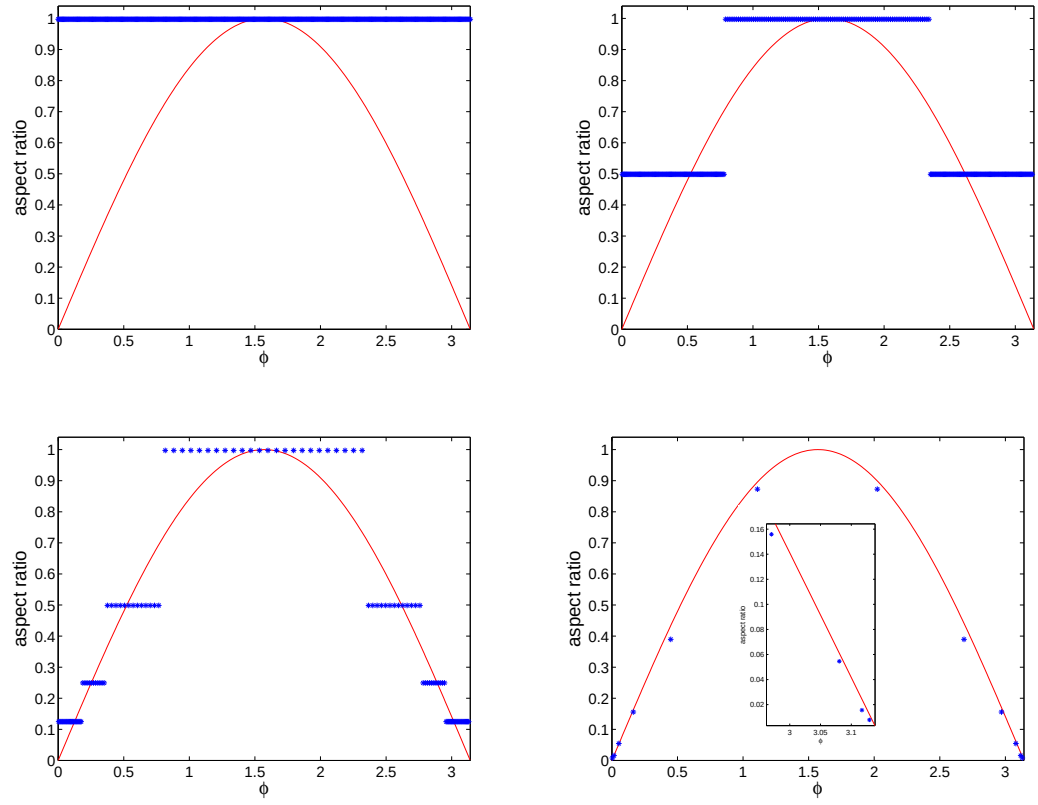

Figure 5-4: Aspect ratio $\frac{h_y}{h_x}$ for (top left) zero, (top right) one, (bottom left) four and (bottom right) seven refinements

grids. The right hand side is formed by multiplying A_F with a random chosen exact solution. We use the same machine as in Section 5.1, ie. a single 1.8GHz processor of a Dual dual-core 64bit AMD Opteron 2210 (Cache size 1.0MB and 2GB memory) and the Fortran95 compiler `ifort`. The initial guess is zero, with a relative residual reduction of $tol = 10^{-8}$ as the stopping criterion. The number of pre- and post-smoothing steps used is $\nu_1 = 3$ and $\nu_2 = 2$, but we also test other values. We give the CPU times, the numbers of iterations, N_{its}, required for convergence, and the (geometric) average of the V-cycle convergence factor (excluding the first cycle), μ_{avg}.

Table 5.5 shows the results for this method. The setup time and solve times increase linearly with the problem size and the number of iterations and the average convergence factor remain constant, indicating that the multigrid method is robust with respect to problem size, and performs optimally. It also outperformes line relaxation with full coarsening (FCLR) as we can see from Table 5.2. We therefore treat this as the optimal method to solve (5.1.2), and we call this method '*non-uniform multigrid*' (NUMG).

We conclude this section with the same tests as above, but using multigrid as a

Problem size	# Coarse grids	Setup time (s)	Solve time (s)	N_{its}	μ_{avg}
32x32	2	2.38E-3	3.67E-3	7	0.075
64x64	3	8.49E-3	1.35E-2	7	0.070
128x128	4	3.16E-2	6.17E-2	7	0.066
256x256	5	1.22E-1	2.44E-1	6	0.057
512x512	6	4.87E-1	1.10	6	0.057

Table 5.5: Model problem (5.1.2) solved using NUMG as a stand alone solver. CPU time in seconds.

preconditioner to the CG method (with one V-cycle per iteration) for a direct comparison. The setup times are identical for the two methods, so only the solve times are given, together with the number of iterations and the average convergence factor. The results, shown in Table 5.6, indicate that the number of iterations, the average convergence factor and the number of iterations are all near identical to using multigrid as a stand alone solver.

Problem size	# Coarse grids	Solve time (s)	N_{its}	μ_{avg}
32x32	2	3.66E-3	7	0.074
64x64	3	1.42E-2	7	0.075
128x128	4	6.30E-2	7	0.076
256x256	5	2.66E-1	7	0.078
512x512	6	1.290	7	0.078

Table 5.6: Model problem (5.1.2) solved using NUMG as a preconditioner to the CG method. CPU time in seconds.

5.3 Extensions to Three Dimensions

Let us now consider how the non-uniform multigrid method can be extended to optimally solve 3D elliptic problems. We start again by studying a suitable model problem on a cube:

$$-\frac{\partial^2 u}{\partial x^2} - \frac{\partial}{\partial y}\left(\sin^2(\pi y)\frac{\partial u}{\partial y}\right) - L_z \sin^2(\pi y)\frac{\partial^2 u}{\partial z^2} = g \quad \text{on} \quad \Omega\,, \tag{5.3.1}$$

where $\Omega = (0,1)\times(\varepsilon, 1-\varepsilon)\times(0,1)$. A finite volume discretisation of (5.3.1), described in detail in Section 3.2.1, gives the stencil

$$-L_z \sin^2(\pi y_j)\frac{h_x h_y}{h_z}\begin{bmatrix} & -\sin^2(\pi y_{j+\frac{1}{2}})\frac{h_x h_z}{h_y} & \\ -\frac{h_y h_z}{h_x} & -\sum & -\frac{h_y h_z}{h_x} \\ & -\sin^2(\pi y_{j-\frac{1}{2}})\frac{h_x h_z}{h_y} & \end{bmatrix} - L_z \sin^2(\pi y_j)\frac{h_x h_y}{h_z}.$$

In section 5.3.1 we will solve (5.3.1) with $L_z = 10^4$, using a z-line smoother and uniform full coarsening in the z-direction, with the conditional semi-coarsening strategy on the $x-y$ plane as before. We observe optimal convergence results using this approach, but not so when L_z is decreased to, say $L_z = 1$, as we see in Section 5.3.2. In Section 5.3.3 we find that by not coarsening in the z-direction we retain optimal convergence even for small L_z. This subsection is concluded in Section 5.3.4 which investigates whether conditional semi-coarsening is really necessary on the $x-y$ plane by comparing it with full and semi-coarsening.

5.3.1 Dealing with the Additional Anisotropy in the z-Direction

Let us assume first that $L_z = 10^4$, which is a typical situation in meteorology, i.e. in a very shallow 3D region, and is of a similar type to the Helmholtz equation from Section 2.1. In this case, in addition to the anisotropy on the $x-y$ plane (which is exactly the same as that of the 2D model problem (5.1.2)), a second source of anisotropy is introduced in the z-direction. This is created by the large coefficient, L_z, and near the region $y = 0.5$ in particular, where $\sin(\pi y) \approx 1$, this term is the main source of anisotropy, i.e:

$$\mathcal{O}\left(L_z h\right) \begin{bmatrix} & \mathcal{O}(h) & \\ \mathcal{O}(h) & -\sum & \mathcal{O}(h) \\ & \mathcal{O}(h) & \end{bmatrix} \mathcal{O}\left(L_z h\right).$$

where we have used a generic mesh width h because $h_x \approx h_y \approx h_z$. Since $L_z \gg 1$, the off diagonal entries in the z-direction are very large in comparison to those in the x- and y-directions, thus causing the large anisotropy. We deal with this by simply introducing z–line relaxation, namely z–line Gauss–Seidel, and combine this with full-coarsening in the z-direction. However, the further away from $y = 0.5$, the more prominent the anisotropy on the $x-y$ plane is, i.e.

$$\mathcal{O}\left(L_z \sin^2(\pi y_j)h\right) \begin{bmatrix} & \mathcal{O}(\sin^2(\pi y_{j+\frac{1}{2}})h) & \\ \mathcal{O}(h) & -\sum & \mathcal{O}(h) \\ & \mathcal{O}(\sin^2(\pi y_{j+\frac{1}{2}})h) & \end{bmatrix} \mathcal{O}\left(L_z \sin^2(\pi y_j)h\right),$$

and we tackle this anisotropy with conditional semi-coarsening on the $x-y$ plane. The coarsening factor using this strategy is approximately 6. For numerical experiments, we use the same settings as in the experiments in Sections 5.1 and 5.2, in particular $\nu_1 = 3$ and $\nu_2 = 2$, but with a z-line Gauss–Seidel method as the coarse grid solver. We also set $\varepsilon = 0$ as before. However, we can also use the CG method as we did in Sections 5.1 and 5.2 for similar results. We see in Table 5.7 that this method leads

to an extremely efficient method with a V-cycle convergence factor of between 10^{-7} and 10^{-8} in the first iteration. Hence convergence to the required tolerance of 10^{-8} can occur in one iteration, which is the case for the smaller problem sizes. Even for the larger problem sizes, two iterations are sufficient, though the convergence factor reduces to about 10^{-2} after the first iteration.

Problem size	# Coarse grids	Setup time (s)	Solve time (s)	N_{its}	μ_{avg}
32x32x32	2	6.30E-2	4.34E-2	1	–
64x64x64	3	4.69E-1	4.63E-1	1	–
128x128x128	4	4.53	8.64	2	0.007
256x256x256	5	35.4	70.32	2	0.010

Table 5.7: Three dimensional model problem (5.3.1) with $L_z = 10^4$ solved using NUMG. CPU time in seconds.

Note that, as discussed in Chapter 4, z-line relaxation on its own will be also very effective as a solver if the direction of the strongest coupling does not change throughout the domain. As $y \to \varepsilon$ and $y \to 1-\varepsilon$, the off-diagonal entry in the y-direction, i.e. $-\sin^2(\pi y_{j\pm\frac{1}{2}})\frac{h_x h_z}{h_y}$, will decrease but the off-diagonal entry in the x-direction, i.e. $-\frac{h_y h_z}{h_x}$, will be unchanged. The size of the off-diagonal entry in the z-direction, i.e. $-L_z \sin(\pi y_j)\frac{h_x h_y}{h_z}$, will also decrease near the y-boundaries. However, it must remain larger than the coupling in the x-direction for z-line relaxation to be effective, and as long as

$$L_z \sin^2(\pi y_j) > 1 \tag{5.3.2}$$

holds at each node in the domain, the coupling in the z-direction will remain largest. With $L_z = 10^4$, (5.3.2) will hold even at nodes closest to the y boundaries unless there are over 150 grid points in the y-direction. Hence line relaxation as a stand alone solver can also be expected to perform well for this problem for moderate problem sizes, and the results given in Table 5.8 confirm that this is the case. For larger problem sizes, however, it is clear that the performance of line relaxation deteriorates and NUMG becomes far superior.

Recall though, that $\nu_1 = 3$ and $\nu_2 = 2$ and so one multigrid V-cycle uses five sweeps of line relaxation. Comparing the times for problem size $64 \times 64 \times 64$ in Tables 5.7 and 5.8, we see that the extra work on the coarse grids is insignificant and essentially comes for free, and it is only the additional setup cost we have to take into account. Therefore, we believe that it is always worth using multigrid.

Problem size	Total time (s)	N_{its}
32x32x32	3.85E-2	4
64x64x64	8.63E-1	10
128x128x128	42.84	70
256x256x256	> 1000	621

Table 5.8: Three dimensional model problem (5.3.1) with $L_z = 10^4$ solved using z-line relaxation. CPU time in seconds.

5.3.2 Reducing the Anisotropy in the z-Direction

Now consider (5.3.1) with $0 < L_z \ll 10^4$. This is effectively reducing the strong coupling in the z-direction, which results in anisotropies that are similar to those in the Quasi-Geostrophic Omega equation from Section 2.2. We investigate the effect of this reduced anisotropy on the efficiency of NUMG, and these tests are of importance because it is necessary for the method to be robust $\forall L_z \in (0, 10^4]$. We have seen in Chapter 2 that problems of these types are of great importance at the Met Office.

Table 5.9 shows that the method becomes less robust to problem size the smaller the value of L_z. As L_z is decreased, the number of iterations rises with respect to problem size, and for $L_z = 1$, more than 300 iterations are required. Hence NUMG is not fully robust in this form and we need to further adapt it.

	$L_z = 10^4$		$L_z = 10^2$		$L_z = 1$	
Problem size	N_{its}	Total time	N_{its}	Total time	N_{its}	Total time
32x32x32	1	8.64E-2	2	1.64E-1	11	5.45E-1
64x64x64	1	9.32E-1	3	1.91	32	12.1
128x128x128	2	13.2	5	23.7	97	386
256x256x256	2	106	7	227	302	> 1000

Table 5.9: NUMG used to solve (5.3.1) for varying strengths of the vertical anisotropy. CPU time in seconds.

The reason why the performance degrades for smaller values of L_z is because as we decrease L_z, the off-diagonal entries in the z-direction also decrease, so they are no longer dominated by L_z but by $\sin(\pi y)$:

$$\mathcal{O}\left(\sin^2(\pi y_j)h\right) \begin{bmatrix} & \mathcal{O}(\sin^2(\pi y_{j+\frac{1}{2}})h) & \\ \mathcal{O}(h) & -\sum & \mathcal{O}(h) \\ & \mathcal{O}(\sin^2(\pi y_{j+\frac{1}{2}})h) & \end{bmatrix} \mathcal{O}\left(\sin^2(\pi y_j)h\right).$$

As $L_z \to 1$, (5.3.2) will hold at fewer and fewer grid points, particularly near the y boundaries, and so at these points the z off-diagonal entry will be smaller than the x

off-diagonal entry, making z-line relaxation alone unable to cope with the anisotropy. By reducing L_z we are effectively creating a problem where the direction of anisotropy changes as one moves along the y-axis. When $L_z = 1$, (5.3.2) doesn't hold anywhere in the domain, causing line relaxation to be ineffective. Therefore, NUMG needs to be suitably adapted to be able to deal with any uncertainties in the direction of the strong coupling caused by the size of L_z.

5.3.3 Line Relaxation and No Coarsening in z

We established in Section 5.1.1 for the 2D model problem (5.1.2) that line relaxation with full coarsening was very efficient provided that the direction of strong coupling didn't change. However, if the direction of strong anisotropy was unknown, then it was necessary to combine line relaxation in one direction with semi coarsening in the other. This is exactly the situation we find ourselves in for problem (5.3.1), where the direction of the strong coupling switches between x and z. Hence the best remedy to improve the performance of NUMG for this problem is to construct the coarse grids without coarsening in the z-direction, ie. we impose line relaxation with no coarsening in the z-direction. In the $x - y$ plane we use again conditional semi-coarsening to deal with the additional anisotropy caused by the coefficient variation, i.e. by $\sin^2(\pi y)$. This would then be expected to deal with the anisotropy that switches between the x- and z-directions to produce an optimal solver. In fact the theory in Section 5.4.2 shows that this is indeed the case.

Intuitively this strategy makes sense, because by not coarsening in z, the z off-diagonal entries on the coarse grids will become relatively larger than the x and y off diagonal entries. Hence (5.3.2) will hold at a larger percentage of grid points on each successively coarser grid which will make z-line relaxation more effective.

The obvious drawback to this method is that the coarsening factor will be reduced to approximately 3, so more work is required on the coarse grids. However, this is hardly a compromise if robustness with respect to problem size is achieved for each of the values of L_z and thus the number of iterations is largely reduced. The results from Table 5.10 – the key results from this section – confirm that the method is robust with respect to problem size for each of the tested values of L_z, unlike the method tested in Table 5.9. Thus by not coarsening in the z-direction, the NUMG method with z-line relaxation and conditional semi-coarsening on the $x - y$ plane optimally solves (5.3.1) for any value of $L_z > 0$.

	$L_z = 10^4$		$L_z = 10^2$		$L_z = 1$	
Problem size	N_{its}	Total time	N_{its}	Total time	N_{its}	Total time
32x32x32	1	1.13E-1	3	0.26	8	4.23E-1
64x64x64	2	1.41	4	2.89	9	4.37
128x128x128	3	15.19	5	23.6	10	39.9
256x256x256	3	126	5	189	10	326

Table 5.10: NUMG with conditional semi-coarsening on the $x-y$ plane, z-line relaxation and no vertical coarsening applied to (5.3.1). CPU time in seconds. The method is optimal for all values of $L_z > 0$.

5.3.4 Comparison with Full Coarsening and Semi Coarsening

To confirm the importance of conditional semi-coarsening in the $x-y$ plane, we conclude this section with tests for solving (5.3.1) using multigrid with line relaxation and no coarsening in the z-direction in conjunction with *full* or *semi* coarsening on the x–y plane. We use all the same settings as for the tests in Section 5.3.3 but without using conditional semi-coarsening. The purpose of this test is to determine whether conditional semi-coarsening is really necessary for optimally solving (5.3.1) for all $L_z > 0$. The results, when using full coarsening, are given in Table 5.11 for the same values of L_z as before, so that the performance of the method is measured for different anisotropies. We compare the results with those in Table 5.10.

	$L_z = 10^4$		$L_z = 10^2$		$L_z = 1$	
Problem size	N_{its}	Total time	N_{its}	Total time	N_{its}	Total time
32x32x32	1	9.88E-2	8	4.90E-1	27	1.25
64x64x64	2	1.23	30	15.18	80	36.4
128x128x128	11	48.81	92	376	–	–
256x256x256	48	> 1000	–	–	–	–

Table 5.11: Solving (5.3.1) using NUMG with z-line relaxation, no vertical coarsening and *full* coarsening on the x–y plane. CPU time in seconds.

The results can be interpreted as follows. When $L_z = 10^4$, we have seen in Section 5.3.1 that z-line relaxation is effective even as a stand alone solver. When combined with full coarsening the results are improved further, but the method is clearly not optimal since the number of iterations grows with problem size. Therefore it is clear that the additional anisotropy on the x-y plane is not handled effectively by full coarsening, which we have already demonstrated when using FCPR to solve (5.1.2) in Section 5.1.

For smaller values of L_z, the method becomes less optimal, and a large number of iterations are required to solve even the smaller problem sizes. This is because z-line

relaxation is less effective as a result of the direction of the strongest coupling changing as one moves up the y-axis, confirming the need for semi-coarsening.

However, when using semi-coarsening everywhere on the x–y plane (i.e. coarsening only in the x-direction) similar results are produced, see Table 5.12. Compared with Table 5.11, the performances are not quite as poor, because SCPR is more efficient than FCPR when solving (5.1.2) on the x-y plane, as demonstrated in Section 5.1. However, the method is nevertheless suboptimal for $L_z \ll 10^4$, and is clearly inferior to the conditional semi-coarsening method from Section 5.3.3. Hence it is evident from the results in this section that z-line relaxation must be combined with conditional semi-coarsening on the x–y plane in order to optimally solve (5.3.1) for all $L_z > 0$.

	$L_z = 10^4$		$L_z = 10^2$		$L_z = 1$	
Problem size	N_{its}	Total time	N_{its}	Total time	N_{its}	Total time
32x32x32	1	1.15E-1	3	3.11E-1	13	6.55E-1
64x64x64	2	1.51	5	3.40	33	17.88
128x128x128	3	15.40	8	38.92	101	416
256x256x256	3	135	11	381	–	–

Table 5.12: Solving (5.3.1) using NUMG with z-line relaxation, no vertical coarsening and coarsening only in the x-direction. CPU time in seconds.

5.4 Convergence Theory Using a Tensor Product Approach

Let us start again by looking at a general 2D problem. In this section we prove that, when combining line relaxation in one direction with semi coarsening in the other (i.e. SCLR), the convergence rates of the multigrid V-cycle applied to 2D elliptic problems on tensor product grids are bounded independently of the problem size and any existing grid-aligned anisotropies in the coefficients. This includes the degenerate cases where the operator features a singular perturbation in one coordinate direction. Given the convenient feature of the anisotropy being aligned with the coordinate directions, the construction of a robust multigrid method is greatly facilitated. Such a multigrid method that relies on the grid alignment of the anisotropy will be referred to in this section as *tensor product multigrid*. The analysis in this section follows the work of Börm and Hiptmair [14] (see also [13, §2]). We extend their theory to prove robustness of the method also in 3D when combining z-line relaxation with no coarsening in the z-direction.

Problems (5.1.2) and (5.3.1) are exactly the type of problems amenable to tensor product multigrid analysis. For theoretical purposes they will be discretised here using bilinear and trilinear finite elements, respectively (cf. Section 3.3.1). As we have seen

in Section 3.3.2, these are directly linked also to the finite volume scheme by using particular quadrature formulas. Hence the convergence theory covered in this section will also be applicable to a finite volume discretisation of (5.1.2) and (5.3.1).

Taking the idea from classical Fourier analysis [44] of multigrid, eigenspace techniques are employed to separate the coordinate directions, such that the 2D problem is reduced to a family of problems in subspaces involving only one coordinate direction. Thus, when employing line relaxation in one coordinate direction with semi coarsening in the other, the comparatively simple one-dimensional (1D) multigrid analysis of Braess and Hackbusch [16] translates directly into results for the 2D problem. In our extension of the theory to 3D, we use similar techniques to reduce the 3D problem to a family of problems in subspaces involving two coordinate directions only.

In Section 5.4.1 we recall the theory from [14] to prove the robustness of the multigrid method using SCLR applied to the 2D problem (5.1.2). Then in Section 5.4.2, we extend the theory to 3D and prove that the multigrid V-cycle is robust for solving (5.3.1) when combining z-line relaxation with no coarsening in the z-direction and conditional semi-coarsening on the $x-y$ plane.

5.4.1 Convergence Theory for NUMG in Two Dimensions

Recall problem (5.1.1) which is

$$-\nabla \cdot (K(x,y)\nabla u) = g \quad \text{on} \quad \Omega = \Omega_x \times \Omega_y, \tag{5.4.1}$$

with

$$K(x,y) = \begin{bmatrix} \alpha_1(x,y) & 0 \\ 0 & \alpha_2(x,y) \end{bmatrix},$$

for separable functions α_1 and α_2 and where $\Omega_x = (c^x, d^x)$ and $\Omega_y = (c^y, d^y)$ for some constants $c^x, d^x, c^y, d^y \in \mathbb{R}$. As in Section 3.3.1, we write (5.4.1) in weak form as follows: Find $u \in H_0^1(\Omega)$ such that

$$a(u,v) = (g,v)_{L_2(\Omega)} \quad \forall v \in H_0^1(\Omega), \tag{5.4.2}$$

where

$$a(u,v) = \int_\Omega \nabla v \cdot (K\nabla u)\, dxdy \quad \text{and} \quad (g,v)_{L_2(\Omega)} = \int_\Omega gv\ dxdy\,.$$

Thanks to the grid aligned anisotropy we have $K_{12} \equiv K_{21} \equiv 0$, and so $a(\cdot,\cdot)$ can be decomposed into two bilinear forms $a_1(\cdot,\cdot)$ and $a_2(\cdot,\cdot)$ as follows

$$a(u,v) = a_1(u,v) + a_2(u,v),$$

where $a_1(\cdot,\cdot)$ and $a_2(\cdot,\cdot)$ are given by

$$a_1(u,v) = \int_\Omega \alpha_1(x,y)\frac{\partial}{\partial x}u(x,y)\frac{\partial}{\partial x}v(x,y)\ dxdy,$$
$$a_2(u,v) = \int_\Omega \alpha_2(x,y)\frac{\partial}{\partial y}u(x,y)\frac{\partial}{\partial y}v(x,y)\ dxdy.$$

Now we formulate an approximate weak form by firstly decomposing Ω into a mesh of rectangular elements, and then choosing a finite dimensional space $V_h \subset H_0^1(\Omega)$ on this mesh. Noting that the bilinear forms $a_1(\cdot,\cdot)$ and $a_2(\cdot,\cdot)$ correspond to the derivatives in the x- and y-directions respectively, we introduce continuous, piecewise linear finite element spaces $V^x \subset H_0^1(\Omega_x)$ and $V^y \subset H_0^1(\Omega_y)$, and suitable nodal basis functions for each finite element space. Then the finite element space V_h is defined as

$$V_h = V^x \otimes V^y,$$

where $V^x \otimes V^y = span\{u(x)v(y)\ :\ u \in V^x,\ v \in V^y\}$. The basis of V_h will be the set of products of any two basis functions of V^x and V^y.

Recalling that $\alpha_1(x,y) = \alpha_1^1(x)\alpha_1^2(y)$ and $\alpha_2(x,y) = \alpha_2^1(x)\alpha_2^2(y)$, for functions $u_h = u^x \otimes u^y \in V_h$ and $v_h = v^x \otimes v^y \in V_h$ with $u^x, v^x \in V^x$ and $u^y, v^y \in V^y$, we have

$$\begin{aligned} a(u^x \otimes u^y, v^x \otimes v^y) &= a_1(u^x \otimes u^y, v^x \otimes v^y) + a_2(u^x \otimes u^y, v^x \otimes v^y) \\ &= a_x(u^x, v^x)\ b_y(u^y, v^y) + b_x(u^x, v^x)\ a_y(u^y, v^y)\,, \end{aligned} \tag{5.4.3}$$

where

$$\begin{aligned} a_x(u_x, v_x) &= \int_{\Omega_x} \alpha_1^1(x)\frac{\partial}{\partial x}u^x(x)\frac{\partial}{\partial x}v^x(x)\ dx\,, \\ a_y(u_y, v_y) &= \int_{\Omega_y} \alpha_2^2(y)\frac{\partial}{\partial y}u^y(y)\frac{\partial}{\partial y}v^y(y)\ dy\,, \\ b_x(u_x, v_x) &= \int_{\Omega_x} \alpha_2^1(x)u^x(x)v^x(x)\ dx\,, \\ b_y(u_y, v_y) &= \int_{\Omega_y} \alpha_1^2(y)u^y(y)v^y(y)\ dy\,. \end{aligned}$$

Using the definition of the Kronecker product, $\otimes$, given in (3.3.22), and the bases for V^x and V^y, the bilinear form (5.4.3) can be written in matrix notation as

$$A_h = A^x \otimes B^y + B^x \otimes A^y\,, \tag{5.4.4}$$

where $A_h \in \mathbb{R}^{n\times n}$ is the symmetric positive definite (SPD) stiffness matrix correspond-

ing to $a(\cdot,\cdot)$ with respect to V_h, with $n_x = \dim V^x$, $n_y = \dim V^y$ and $n = n_x \times n_y = \dim V_h$. Similarly, $A^x \in \mathbb{R}^{n_x \times n_x}$ and $A^y \in \mathbb{R}^{n_y \times n_y}$ are the stiffness matrices corresponding to a_x and a_y, respectively, and $B^x \in \mathbb{R}^{n_x \times n_x}$ and $B^y \in \mathbb{R}^{n_y \times n_y}$ are the mass matrices corresponding to b_x and b_y, respectively. These are also SPD matrices. Finally $\mathbf{b}_h \in \mathbb{R}^n$ is the load vector of the right-hand-side of (5.4.2) and the resulting system of linear equations is written as

$$A_h \mathbf{u}_h = \mathbf{b}_h \,, \tag{5.4.5}$$

which corresponds to the Galerkin discretisation of (5.1.2) by means of bilinear finite elements.

Tensor Product Multigrid

The key idea of tensor product multigrid is to split V_h into a number of A^y-orthogonal subspaces on which it is possible to reduce the two-dimensional problem to problems involving only the bilinear forms a_x and b_x. The A^y-orthogonality of the subspaces relies on two basic properties:

Semi coarsening: We only coarsen in the x-direction, thus picking a nested sequence of finite element spaces

$$V_1^x \subset V_2^x \subset \cdots \subset V_F^x = V^x$$

and the space $V_\ell := V_\ell^x \otimes V^y$ is used to discretise problem (5.1.2) on level ℓ, $\ell = 1, \ldots, F$, where V^y does not change from level to level.

Line relaxation: All the degrees of freedom located at grid points with the same x-coordinate are relaxed together, i.e. y-line relaxation is used. The relaxation occurs in the subspaces $span\{\phi_i^\ell \otimes u^y : u^y \in V^y\}$, $i = 1, \ldots, n_x^\ell$ where $\{\phi_i^\ell\}_{i=1}^{n_x^\ell}$ is the set of nodal basis functions of V_ℓ^x, $\ell = 1, \ldots, F$ and $n_x^\ell = \dim V_\ell^x$.

The complete proof of the convergence rate of the multigrid method using line relaxation in one direction and semi coarsening in the other will be given later in this section.

Let us first define the operators required for multigrid methods in a tensor product setting. It is known that the vector space $\mathbb{R}^n$ and the finite element space V_h are isomorphic. Thus, as in the proof of the approximation property (given in Appendix C), we define a bijective mapping from a vector $\mathbf{u}_\ell \in \mathbb{R}^{n^\ell}$ (where $n^\ell = n_x^\ell \times n_y = dim\ V_\ell$) to a function $u_\ell \in V_\ell$ via the operator $p_\ell : \mathbb{R}^{n^\ell} \to V_\ell$ such that

$$u_\ell = p_\ell \mathbf{u}_\ell \,,$$

with the adjoint mapping from u_ℓ to $\mathbf{u}_\ell$ as $r_\ell = p_\ell^T$ such that

$$< r_\ell u_\ell, \mathbf{u}_\ell >= (u_\ell, p_\ell \mathbf{u}_\ell)_{L_2(\Omega)} ,$$

where $< \cdot, \cdot >$ is the scalar product of vectors, i.e.

$$< \mathbf{a}, \mathbf{b} >= \sum_{i=1}^{n^\ell} a(i)b(i) .$$

Using p_ℓ, we define the canonical interpolation operator $P_\ell : \mathbb{R}^{n^{\ell-1}} \to \mathbb{R}^{n^\ell}$, by

$$P_\ell = p_\ell^{-1} p_{\ell-1}$$

for $\ell = 2, \ldots, F$. The canonical restriction operator, $R_\ell : \mathbb{R}^{n^\ell} \to \mathbb{R}^{n^{\ell-1}}$, is defined as the adjoint of P_ℓ, i.e. $R_\ell = P_\ell^T$ such that

$$< P_\ell \mathbf{u}_{\ell-1}, \mathbf{w}_\ell >=< \mathbf{u}_{\ell-1}, R_\ell \mathbf{w}_\ell > .$$

Now, thanks to semi-coarsening, the global transfer matrices R_ℓ and P_ℓ between $\mathbb{R}^{n^\ell}$ and $\mathbb{R}^{n^{\ell-1}}$ can be written as

$$\begin{aligned} P_\ell &= P_\ell^x \otimes P^y = P_\ell^x \otimes I \ \in \mathbb{R}^{n_x^\ell n_y \times n_x^{\ell-1} n_y} , \\ R_\ell &= R_\ell^x \otimes R^y = R_\ell^x \otimes I \ \in \mathbb{R}^{n_x^{\ell-1} n_y \times n_x^\ell n_y} , \end{aligned}$$

where the matrices P_ℓ^x and R_ℓ^x are transfer operators between $\mathbb{R}^{n_x^\ell}$ and $\mathbb{R}^{n_x^{\ell-1}}$, and we have $P^y = R^y = I$, thanks to not coarsening in the y-direction.

In addition to semi-coarsening, we use y-line relaxation, where we recall from Section 4.2 that relaxation methods are written as iterations of the following general form

$$\mathbf{u}_\ell^{(new)} = \mathbf{u}_\ell^{(old)} - W_\ell^{-1} \left(A_\ell \mathbf{u}_\ell^{(old)} - \mathbf{b}_\ell \right) , \tag{5.4.6}$$

where $W_\ell \in \mathbb{R}^{n^\ell \times n^\ell}$ approximates A_ℓ. For a block-Jacobi smoother, damped by $\theta \in \mathbb{R}^+$, it is shown in [14] that

$$W_\ell = W_\ell^{A^x} \otimes B^y + W_\ell^{B^x} \otimes A^y , \tag{5.4.7}$$

where $W_\ell^{B^x}$ and $W_\ell^{A^x}$ are the diagonals of B_ℓ^x and A_ℓ^x, multiplied by θ, respectively. Similarly, a block Gauss–Seidel smoother can be cast in the form (5.4.7).

Using the following tensor product rules

$$(C \otimes D)(\mathbf{x} \otimes \mathbf{y}) = (C\mathbf{x}) \otimes (D\mathbf{y}) \quad \text{for matrices } C,\ D \text{ and vectors } \mathbf{x},\ \mathbf{y}, \tag{5.4.8}$$

$$< \boldsymbol{\alpha} \otimes \boldsymbol{\beta}, \boldsymbol{\gamma} \otimes \boldsymbol{\delta} > = < \boldsymbol{\alpha}, \boldsymbol{\gamma} >< \boldsymbol{\beta}, \boldsymbol{\delta} > \quad \text{for vectors } \boldsymbol{\alpha},\ \boldsymbol{\beta},\ \boldsymbol{\gamma} \text{ and } \boldsymbol{\delta}, \tag{5.4.9}$$

it can easily be seen, e.g. in [14], that for any $\mathbf{u}_\ell^x, \mathbf{v}_\ell^x \in \mathbb{R}^{n_x^\ell}$ and $\mathbf{u}^y, \mathbf{v}^y \in \mathbb{R}^{n_y}$ we have

$$\begin{aligned} &< W_\ell(\mathbf{u}_\ell^x \otimes \mathbf{u}^y), \mathbf{v}_\ell^x \otimes \mathbf{v}^y > \\ &\qquad = < W_\ell^{A^x} \mathbf{u}_\ell^x, \mathbf{v}_\ell^x >< B^y \mathbf{u}^y, \mathbf{v}^y > + < W_\ell^{B^x} \mathbf{u}_\ell^x, \mathbf{v}_\ell^x >< A^y \mathbf{u}^y, \mathbf{v}^y >, \end{aligned}$$

which shows (5.4.7), as required. Note that, for a block-Jacobi smoother, W_ℓ corresponds to the diagonal blocks of A_ℓ multiplied by θ.

Now we aim to use a *block-diagonalisation* procedure to accomplish a separation of coordinate directions, which will reduce the 2D problem into a problem involving only the matrices A_ℓ^x, $W_\ell^{B^x}$ and $W_\ell^{A^x}$. This is done by considering A^y-orthogonal subspaces induced by tensor products of

- eigenvectors $\{\mathbf{e}_j^y\}_{j=1}^{n_y}$ of A^y, and
- arbitrary x-components.

Let the vectors $\{\mathbf{e}_j^y\}_{j=1}^{n_y}$ be a basis of $\mathbb{R}^{n_y}$ which form a B^y-orthonormal generalized eigenbasis of (A^y, B^y), i.e.

$$\left.\begin{aligned} A^y \mathbf{e}_j^y &= \omega_j B^y \mathbf{e}_j^y \\ < B^y \mathbf{e}_j^y, \mathbf{e}_k^y > &= \delta_{jk} \end{aligned}\right\}. \tag{5.4.10}$$

Such a basis exists since A^y and B^y are both SPD. Here, $\{\omega_j\}_{j=1}^{n_y}$ are the strictly positive eigenvalues for the pair (A^y, B^y), a consequence of A^y and B^y being positive definite. If (5.4.10) holds, then we also have

$$< A^y \mathbf{e}_j^y, \mathbf{e}_k^y > = \omega_j \delta_{jk}\,. \tag{5.4.11}$$

so the vectors $\{\mathbf{e}_j^y\}_{j=1}^{n_y}$ are also A^y-orthogonal. Using this A^y-orthogonality of the eigenvectors, and the two tensor product rules (5.4.8) and (5.4.9), we deduce that for all $j, k = 1, \ldots, n_y$ and for all $\mathbf{u}_\ell^x, \mathbf{v}_\ell^x \in \mathbb{R}^{n_x^\ell}$,

$$\begin{aligned} &< \mathbf{u}_\ell^x \otimes \mathbf{e}_k^y, A_\ell(\mathbf{v}_\ell^x \otimes \mathbf{e}_j^y) > \\ &= < \mathbf{u}_\ell^x \otimes \mathbf{e}_k^y, (A_\ell^x \otimes B^y)(\mathbf{v}^x \otimes \mathbf{e}_j^y) + (B_\ell^x \otimes A^y)(\mathbf{v}_\ell^x \otimes \mathbf{e}_j^y) > \ \text{(by definition of } A_\ell) \\ &= < \mathbf{u}_\ell^x, A_\ell^x \mathbf{v}_\ell^x >< \mathbf{e}_k^y, B^y \mathbf{e}_j^y > \ + \ < \mathbf{u}_\ell^x, B_\ell^x \mathbf{v}_\ell^x >< \mathbf{e}_k^y, A^y \mathbf{e}_j^y > \ \text{(by (5.4.8), (5.4.9))} \\ &= \delta_{j,k} < \mathbf{u}_\ell^x, (A_\ell^x + \omega_j B_\ell^x)\mathbf{v}_\ell^x > \ \text{(by (5.4.10), (5.4.11))}\,, \end{aligned} \tag{5.4.12}$$

for $\mathbf{u}_\ell^x,\ \mathbf{v}_\ell^x \in \mathbb{R}^{n_x^\ell}$, and similarly

$$< \mathbf{u}_\ell^x \otimes \mathbf{e}_k^y, W_\ell(\mathbf{v}_\ell^x \otimes \mathbf{e}_j^y) >= \delta_{jk} < \mathbf{u}_\ell^x, (W_\ell^{A^x} + \omega_j W_\ell^{B^x})\mathbf{v}_\ell^x > \ .$$

Now we define

$$\begin{aligned} A_{\ell,\omega}^x &= \omega B_\ell^x + A_\ell^x, \\ W_{\ell,\omega}^x &= \omega W_\ell^{B^x} + W_\ell^{A^x}. \end{aligned}$$

and for $j = 1, \ldots, n_y$, we define matrices $Q_\ell^{(j)} \in \mathbb{R}^{(n_x^\ell n_y)\times n_x^\ell}$ by

$$Q_\ell^{(j)} \mathbf{u}_\ell^x = \mathbf{u}_\ell^x \otimes \mathbf{e}_j^y.$$

for all $\mathbf{u}_\ell^x \in \mathbb{R}^{n_x^\ell}$. Then the matrix $Q_\ell = (Q_\ell^{(j)})_{j=1}^{n_y} \in \mathbb{R}^{(n_x^\ell n_y)\times(n_x^\ell n_y)}$, ie.

$$Q_\ell = \begin{bmatrix} \vdots & \vdots & & \vdots \\ Q_\ell^{(1)} & Q_\ell^{(2)} & \cdots & Q_\ell^{(n_y)} \\ \vdots & \vdots & & \vdots \end{bmatrix},$$

is an orthogonal matrix and the following is a similarity transformation that transforms A_ℓ into a *blockdiagonal* matrix $\hat{A}_\ell$, i.e.

$$\hat{A}_\ell = Q_\ell^T A_\ell Q_\ell = \sum_{j=1}^{n_y} \sum_{k=1}^{n_y} Q_\ell^{(k)T} A_\ell Q_\ell^{(j)} \,.$$

This block diagonalisation procedure is the key step in the analysis, explained as follows:

Let $\mathbf{u}_\ell^x$ and $\mathbf{u}^y$ be random vectors in $\mathbb{R}^{n_x^\ell}$ and $\mathbb{R}^{n_y}$ respectively. Then because $\{\mathbf{e}_j^y\}_{j=1}^{n_y}$ is an eigenbasis of $\mathbb{R}^{n_y}$ we can write

$$\mathbf{u}_\ell^x \otimes \mathbf{u}^y = \sum_{j=1}^{n_y} \alpha_j \mathbf{u}_\ell^x \otimes \mathbf{e}_j^y = \sum_{j=1}^{n_y} \alpha_j Q_\ell^{(j)} \mathbf{u}_\ell^x \,,$$

for some $\alpha_j \in \mathbb{R}$, $j = 1, \ldots, n_y$. Now, for all $j, k = 1, \ldots, n_y$ we have

$$\begin{aligned} \alpha_j \alpha_k < \mathbf{u}_\ell^x, Q_\ell^{(k)T} A_\ell Q_\ell^{(j)} \mathbf{u}_\ell^x > &= \alpha_j \alpha_k < \mathbf{u}_\ell^x \otimes \mathbf{e}_k^y, A_\ell(\mathbf{u}_\ell^x \otimes \mathbf{e}_j^y) > \\ &= \alpha_j \alpha_k \delta_{j,k} < \mathbf{u}_\ell^x, A_{\ell,\omega_j}^x \mathbf{u}_\ell^x > \end{aligned}$$

by the calculations in (5.4.12). Hence $Q_\ell^{(k)T} A_\ell Q_\ell^{(j)} = \delta_{j,k} A^x_{\ell,\omega_j}$ and so $\hat{A}_\ell$ is defined by

$$\hat{A}_\ell = Q_\ell^T A_\ell Q_\ell = blockdiag\left\{ A^x_{\ell,\omega_j} : j = 1, \ldots, n_y \right\}.$$

Similarly Q_ℓ transforms W_ℓ into a blockdiagonal matrix $\hat{W}_\ell$:

$$\hat{W}_\ell = Q_\ell^T W_\ell Q_\ell = blockdiag\left\{ W^x_{\ell,\omega_j} : j = 1, \ldots, n_y \right\}.$$

For the transfer operator P_ℓ, we deduce that:

$$\begin{aligned}
&< \mathbf{u}^x_\ell \otimes \mathbf{e}^y_k, P_\ell(\mathbf{v}^x_{\ell-1} \otimes \mathbf{e}^y_j) > \\
&\quad =< \mathbf{u}^x_\ell \otimes \mathbf{e}^y_k, (P^x_\ell \otimes I)(\mathbf{v}^x_{\ell-1} \otimes \mathbf{e}^y_j) > \quad \text{(by definition of } P_\ell) \\
&\quad =< \mathbf{u}^x_\ell \otimes \mathbf{e}^y_k, P_\ell \mathbf{v}^x_{\ell-1} \otimes I\mathbf{e}^y_j > \qquad \text{(by (5.4.8))} \\
&\quad =< \mathbf{u}^x_\ell, P^x_\ell \mathbf{v}^x_{\ell-1} > \underbrace{< \mathbf{e}^y_k, \mathbf{e}^y_j >}_{\delta_{j,k}} \qquad \text{(by (5.4.9))}
\end{aligned}$$

for $\mathbf{u}^x_\ell \in \mathbb{R}^{n^\ell_x}$ and $\mathbf{v}^x_{\ell-1} \in \mathbb{R}^{n^{\ell-1}_x}$. Thus

$$\begin{aligned}
< \mathbf{u}^x_\ell, Q^T_{\ell,k} P_\ell Q_{\ell-1,j} \mathbf{v}^x_{\ell-1} > &=< \mathbf{u}^x_\ell \otimes \mathbf{e}^y_k, P_\ell(\mathbf{v}^x_{\ell-1} \otimes \mathbf{e}^y_j) > \\
&= \delta_{j,k} < \mathbf{u}^x_\ell, P^x_\ell \mathbf{v}^x_{\ell-1} >,
\end{aligned}$$

and we obtain the following block-diagonalisation using the matrix Q_ℓ:

$$\hat{P}_\ell = Q_\ell^T P_\ell Q_{\ell-1} = blockdiag\left\{ P^x_\ell : j = 1, \ldots, n_y \right\},$$

Note that without semi-coarsening, it would not have been possible to transform P_ℓ into a blockdiagonal matrix.

Similarly,

$$\hat{R}_\ell = Q_{\ell-1}^T R_\ell Q_\ell = blockdiag\left\{ R^x_\ell : j = 1, \ldots, n_y \right\}.$$

Thus we observe that the similarity transformations using Q_ℓ have reduced the operators in the 2D problem to a family of operators for a simpler 1D problem on each of the $a_y(\cdot,\cdot)$-orthogonal subspaces.

Convergence Theory

Thanks to the Galerkin discretisation we have used, problem (5.4.5) is sparse and symmetric positive definite (SPD). For such problems it is possible to resort to the classical multigrid convergence theory of Hackbusch [44, Chapter 7]. We recall Theorem

4.6.10, the convergence of the multigrid V-cycle (c.f. [44, Theorem 7.2.5]):

Theorem 5.4.1. *Let A_F be SPD and assume that*

1. $W_\ell = W_\ell^T > 0 \quad$ *for all* $\ell = 1, \dots, F \quad$ *. (ie. the smoother is symmetric)*
2. $A_\ell \leq W_\ell \quad$ *for all* $\ell = 1, \dots, F \quad$ *. (ie. the smoothing property holds)*
3. *There exists a constant* $C_A > 0$ *independent of* $\ell \in \{2, \dots, F\}$ *such that*

$$0 \leq A_\ell^{-1} - P_\ell A_{\ell-1}^{-1} R_\ell \leq C_A W_\ell^{-1} .$$

(ie. the approximation property holds)

Then the iteration matrix of the multigrid V-cycle, M_F^V (see Lemma 4.6.9), satisfies

$$\|M_F^V\|_{A_F} \leq \frac{C_A}{C_A + \nu} ,$$

where ν is the total number of smoothing steps.

Assumption 1 is automatically satisfied for both the line Jacobi and symmetric line Gauss–Seidel smoothers. Now in order to prove the smoothing property (SP) and the approximation property (AP), we use the block-diagonalisation procedure described above to reduce the analysis of the 2D problem to the analysis of simpler 1D problems.

We do this by applying the matrices Q_ℓ from the right and Q_ℓ^T from the left to the inequalities in the SP and AP. This similarity transformation using Q_ℓ yields equivalent inequalities since Q_ℓ is orthogonal and therefore non-singular. The transformed inequalities feature block-diagonal matrices, and so the SP and AP from Theorem 5.4.1 can be investigated for each diagonal block separately.

In detail, for the SP, applying the similarity transformation gives

$$\begin{aligned} A_\ell \leq W_\ell &\Leftrightarrow Q_\ell^T A_\ell Q_\ell \leq Q_\ell^T W_\ell Q_\ell \qquad \text{(since } Q_\ell \text{ is non-singular)} \\ &\Leftrightarrow \hat{A}_\ell \leq \hat{W}_\ell \\ &\Leftrightarrow A_{\ell,\omega_j} \leq W_{\ell,\omega_j} \quad \forall j = 1, \cdots, n_y , \end{aligned} \tag{5.4.13}$$

and for the AP, we have

$$\begin{aligned} & 0 \leq A_\ell^{-1} - P_\ell A_{\ell-1}^{-1} R_\ell \leq C_A W_\ell^{-1} \\ \Leftrightarrow\ & Q_\ell^T A_\ell^{-1} Q_\ell^{-T} - Q_\ell^T P_\ell Q_{\ell-1} Q_{\ell-1}^{-1} A_{\ell-1}^{-1} Q_{\ell-1}^{-T} Q_{\ell-1}^T R_\ell Q_\ell \leq C_A Q_\ell^T W_\ell^{-1} Q_\ell^{-T} \\ \Leftrightarrow\ & \hat{A}_\ell^{-1} - \hat{P}_\ell \hat{A}_{\ell-1}^{-1} \hat{R}_\ell \leq C_A \hat{W}_\ell^{-1} \\ \Leftrightarrow\ & A_{\ell,\omega_j}^{-1} - P_{\ell,\omega_j} A_{\ell-1,\omega_j}^{-1} R_{\ell,\omega_j} \leq C_A W_{\ell,\omega_j}^{-1} \quad \forall j = 1, \cdots, n_y. \end{aligned} \tag{5.4.14}$$

Thus, thanks to the block-diagonalisation procedure accomplished by y-line relaxation, coarsening only in the x-direction and the use of invariant subspaces induced by a tensor product of the eigenvectors of A^y and arbitrary x-components, the SP and AP need only be investigated for diagonal blocks corresponding to each of the subspaces. Using these diagonal blocks, the main theorem for the convergence of the multigrid V-cycle using tensor product multigrid is given as follows:

Theorem 5.4.2. *For all $j = 1, \ldots, n_y$, suppose that there exists a constant $C_A > 0$ such that*

$$A^x_{\ell,\omega_j} \leq W^x_{\ell,\omega_j} \quad \ell = 1, \ldots, F \tag{5.4.15}$$

$$0 \leq (A^x_{\ell,\omega_j})^{-1} - P^x_\ell (A^x_{\ell-1,\omega_j})^{-1} R^x_\ell \leq C_A (W^x_{\ell,\omega_j})^{-1} \quad \ell = 2, \ldots, F \tag{5.4.16}$$

hold. Then the iteration matrix of the multigrid V-cycle, based on a y-line smoother and coarsening in only the x-direction satisfies

$$\|M^V_F\|_{A_F} \leq \frac{C_A}{C_A + \nu},$$

where ν is the total number of smoothing steps.

The proof of this theorem relies on the fact that (5.4.15) and (5.4.16) are equivalent to the inequalities of the SP and AP from Theorem 5.4.1, when using y-line relaxation and coarsening in x only. Therefore, showing that (5.4.15) and (5.4.16) are satisfied means the SP and AP from Theorem 5.4.1 are also satisfied, and since A_F is SPD we can deduce from Theorem 5.4.1 that the 2D V-cycle will converge independently of problem size.

We can prove the assumptions (5.4.15) and (5.4.16) of Theorem 5.4.2 by resorting to the theory given in Section 4.6. Recall that $\omega_j \geq 0$, which implies A^x_{ℓ,ω_j} and W^x_{ℓ,ω_j} are SPD matrices. Therefore Lemmas 4.6.4 and 4.6.6 show that (5.4.15) and (5.4.16) hold because (4.6.13) is satisfied for any symmetric smoothing iteration and the regularity assumption is satisfied for the family of 1D problems used in Theorem 5.4.2.

Finally we can apply Theorem 5.4.2 to the particular case we have been studying in this chapter, i.e. $\alpha_1 = 1$, $\alpha_2 = \sin^2(\pi y)$ and $\Omega = (0,1) \times (\varepsilon, 1-\varepsilon)$ for $0 < \varepsilon \ll 1$. The coefficients degenerate towards the boundary in the y-direction, but by using y-line relaxation and x-semi coarsening, the problem is reduced to a family of simple 1D Poisson-type problems in the x-direction which have no anisotropy. Hence the SP (5.4.15) and AP (5.4.16) will hold for these problems and Theorem 5.4.2 can be applied.

5.4.2 Convergence Theory for NUMG in Three Dimensions

In this section, we extend the theory from Section 5.4.1 (cf. [14]) to prove the robustness of the NUMG method in 3D for solving (5.3.1) when combining z-line relaxation with no z-coarsening. We resort to the eigenspace techniques used in Section 5.4.1 to separate the z-coordinate from the x- and y-coordinates and to reduce the 3D problem to a family of 2D problems on the $x-y$ plane only. We have already shown heuristically in Section 5.2 that the convergence rate of NUMG when solving these 2D problems is bounded independently of the problem size. As in Section 5.4.1, we resort to a finite element setting and use suitable quadrature rules to infer the results for the finite volume settings used otherwise in this thesis.

Let us first recall from problem (5.3.1) the type of equation we wish to solve using NUMG, namely

$$-\nabla \cdot (K(x,y,z)\nabla u) = g \quad \text{on} \quad \Omega = \Omega_{xy} \times \Omega_z \,,$$

with

$$K(x,y,z) = \begin{bmatrix} \alpha_1(x,y,z) & 0 & 0 \\ 0 & \alpha_2(x,y,z) & 0 \\ 0 & 0 & \alpha_3(x,y,z) \end{bmatrix},$$

for separable functions α_1, α_2 and α_3, i.e. $\alpha_1(x,y,z) = \alpha_1^1(x)\alpha_1^2(y)\alpha_1^3(z)$ etc, and where $\Omega_{xy} = (c^x, d^x) \times (c^y, d^y)$ and $\Omega_z = (c^z, d^z)$. The only additional condition that this general PDE must satisfy is that $\alpha_1^3(z) = \alpha_2^3(z) =: \alpha^3(z)$, i.e. the coefficients α_1 and α_2 have the same z-dependency. In order to solve (5.3.1) we write the PDE in weak form and then approximate the weak form, as done in Sections 3.3.1 and 5.4.1, by trilinear finite elements. The weak form of (5.3.1) is:
Find $u \in H_0^1(\Omega)$ such that

$$a(u,v) = (g,v)_{L^2(\Omega)} \quad \forall v \in H_0^1(\Omega), \tag{5.4.17}$$

where

$$a(u,v) = \int_\Omega \nabla v \cdot (K\nabla u)\, dxdydz \quad \text{and} \quad (g,v)_{L^2(\Omega)} = \int_\Omega gv \; dxdydz \,.$$

Matrix K is diagonal thanks to the grid aligned anisotropy, meaning that $a(\cdot,\cdot)$ can be decomposed into two bilinear forms $a_1(\cdot,\cdot)$ and $a_2(\cdot,\cdot)$ as in Section 5.4.1 as follows

$$a(u,v) = a_1(u,v) + a_2(u,v),$$

where $a_1(\cdot,\cdot)$ and $a_2(\cdot,\cdot)$ are given by

$$a_1(u,v) = \int_\Omega \hat{K}(x,y,z)\nabla_{xy}u(x,y,z)\cdot\nabla_{xy}v(x,y,z)\ dxdydz,$$
$$a_2(u,v) = \int_\Omega \alpha_3(x,y,z)\frac{\partial}{\partial z}u(x,y,z)\frac{\partial}{\partial z}v(x,y,z)\ dxdydz,$$

where $\hat{K} = K(1{:}2,1{:}2)$ and ∇_{xy} is the 2D gradient $\frac{\partial}{\partial x}\mathbf{e}_1 + \frac{\partial}{\partial y}\mathbf{e}_2$ on the $x-y$ plane (with $\mathbf{e}_1$ and $\mathbf{e}_2$ denoting the unit vectors in the x- and y-directions respectively). Now we approximate the weak form by firstly decomposing Ω into a mesh of cubes and choosing a finite element space $V_h \subset H_0^1(\Omega)$ on this mesh. We introduce a piecewise bilinear finite element space $V^{xy} \subset H_0^1(\Omega_{xy})$ as in Section 5.4.1, as well as a continuous piecewise linear finite element space $V^z \subset H_0^1(\Omega_z)$, with suitable piecewise (bi-)linear model basis functions for each of the spaces. Then we define V_h as

$$V_h = V^{xy} \otimes V^z,$$

where $V^{xy}\otimes V^z = span\{u(x,y)v(z)\ :\ u\in V^{xy},\ v\in V^z\}$. We define the nodal basis functions of V_h as the set of products of any two basis functions of V^{xy} and V^z.

For functions $u_h = u^{xy}\otimes u^z \in V_h$ and $v_h = v^{xy}\otimes v^z \in V_h$ with $u^{xy}, v^{xy} \in V^{xy}$ and $u^z, v^z \in V^z$, we have

$$\begin{aligned}
&a_1(u^{xy}\otimes u^z, v^{xy}\otimes v^z)\\
&\quad= \int_\Omega \left(\begin{bmatrix} \alpha_1^1(x)\alpha_1^2(y)\alpha_1^3(z) & 0 \\ 0 & \alpha_2^1(x)\alpha_2^2(y)\alpha_2^3(z) \end{bmatrix} \begin{bmatrix} \frac{\partial}{\partial x}u^{xy} \\ \frac{\partial}{\partial y}u^{xy} \end{bmatrix} \right) \cdot \begin{bmatrix} \frac{\partial}{\partial x}v^{xy} \\ \frac{\partial}{\partial y}v^{xy} \end{bmatrix} u^z v^z\ dxdydz\\
&\quad= \int_\Omega \left(\alpha_1^1(x)\alpha_1^2(y)\alpha_1^3(z)\frac{\partial}{\partial x}u^{xy}\frac{\partial}{\partial x}v^{xy} + \alpha_2^1(x)\alpha_2^2(y)\alpha_2^3(z)\frac{\partial}{\partial y}u^{xy}\frac{\partial}{\partial y}v^{xy} \right) u^z v^z\ dxdydz\\
&\quad= \underbrace{\int_{\Omega_{xy}} \begin{bmatrix} \alpha_1^1(x)\alpha_1^2(y) & 0 \\ 0 & \alpha_2^1(x)\alpha_2^2(y) \end{bmatrix} \nabla_{xy}u^{xy}\cdot\nabla_{xy}v^{xy}\ dxdy}_{a_{xy}(u^{xy},v^{xy})} \underbrace{\int_{\Omega_z} \alpha^3(z)u^z v^z\ dz}_{b_z(u^z,v^z)}.
\end{aligned}$$

where the separation of coordinate directions in the last line was possible because $\alpha_1^3(z) = \alpha_2^3(z)$. Similarly,

$$\begin{aligned}
&a_2(u^{xy}\otimes u^z, v^{xy}\otimes v^z)\\
&\quad= \underbrace{\int_{\Omega_{xy}} \alpha_3^1(x)\alpha_3^2(y)u^{xy}v^{xy}\ dxdy}_{b_{xy}(u^{xy},v^{xy})} \underbrace{\int_{\Omega_z} \alpha_3^3(z)\frac{\partial}{\partial z}u^z\frac{\partial}{\partial z}v^z\ dz}_{a_z(u^z,v^z)}.
\end{aligned}$$

Using the bases for V^{xy} and V^z, the approximate weak form can be written in matrix notation as

$$A_h = A^{xy} \otimes B^z + B^{xy} \otimes A^z \,,$$

where $A_h \in \mathbb{R}^{n\times n}$ is the SPD stiffness matrix for $a(\cdot,\cdot)$ with respect to V_h, with $n_{xy} = \dim V^{xy}$, $n_z = \dim V^z$ and $n = n_{xy} \times n_z = \dim V_h$. $A^{xy} \in \mathbb{R}^{n_{xy}\times n_{xy}}$ and $A^z \in \mathbb{R}^{n_z\times n_z}$ are the stiffness matrices for a_{xy} and a_z respectively, whilst $B^{xy} \in \mathbb{R}^{n_{xy}\times n_{xy}}$ and $B^z \in \mathbb{R}^{n_z\times n_z}$ are the mass matrices for b_{xy} and b_z respectively. Finally $\mathbf{b}_h \in \mathbb{R}^n$ is the Galerkin vector of the right-hand-side of (5.4.17) and the resulting system of linear equations is written as

$$A_h\mathbf{u}_h = \mathbf{b}_h \,,$$

which corresponds to a discretisation of (5.3.1) by means of trilinear finite elements.

Now, as in Section 5.4.1, we use a block diagonalisation procedure to split V_h into a number of subspaces that are A^z-orthogonal, and this relies on the two properties used by the 3D NUMG method:

No coarsening in the z-direction: Coarsening only occurs on the x–y plane, thus we pick a nested sequence of finite element spaces

$$V_1^{xy} \subset V_2^{xy} \subset \cdots \subset V_F^{xy} = V^{xy}$$

and the space $V_\ell := V_\ell^{xy} \otimes V^z$ is used to discretise problem (5.3.1) on level ℓ, $\ell = 1,\ldots,F$, where V^z does not change from level to level.

z-line relaxation: All the degrees of freedom located at grid points with the same x- and y-coordinate are relaxed together, i.e. z-line relaxation is used. The relaxation occurs in the subspaces $span\{\phi_i^\ell \otimes u^z : u^z \in V^z\}, i = 1,\ldots,n_{xy}^\ell$, where $\{\phi_i^\ell\}_{i=1}^{n_{xy}^\ell}$ is the set of nodal basis functions of V_ℓ^{xy}, $\ell = 1,\ldots,F$ and $n_{xy}^\ell = \dim V_\ell^{xy}$.

Before giving the main theorem of this section, let us define the remaining operators used in the multigrid method. The global transfer operators P_ℓ and R_ℓ between $\mathbb{R}^{n^{\ell-1}}$ and $\mathbb{R}^{n^\ell}$ (which are isomorphic to $V_{\ell-1}$ and V_ℓ respectively) are defined as

$$P_\ell = P_\ell^{xy} \otimes P^z = P_\ell^{xy} \otimes I \ \in \mathbb{R}^{n_{xy}^\ell n_z \times n_{xy}^{\ell-1} n_z} \,,$$
$$R_\ell = R_\ell^{xy} \otimes R^z = R_\ell^{xy} \otimes I \ \in \mathbb{R}^{n_{xy}^{\ell-1} n_z \times n_{xy}^\ell n_z} \,.$$

The matrices P_ℓ^{xy} and R_ℓ^{xy} are transfer operators between $\mathbb{R}^{n_{xy}^{\ell-1}}$ and $\mathbb{R}^{n_{xy}^\ell}$, and we have $P^z = R^z = I$ thanks to not coarsening in the z-direction.

The z-line block smoother has the form (5.4.6) with

$$W_\ell = W_\ell^{A^{xy}} \otimes B^z + W_\ell^{B^{xy}} \otimes A^z \,,$$

where $W_\ell^{A^{xy}}, W_\ell^{B^{xy}} \in \mathbb{R}^{n_{xy}^\ell}$. For a block Jacobi smoother, these correspond to the diagonals of A_ℓ^{xy} and B_ℓ^{xy} respectively. As done in Section 5.4.1, we can show that, for any $\mathbf{u}_\ell^{xy}, \mathbf{v}_\ell^{xy} \in \mathbb{R}^{n_{xy}^\ell}$ and $\mathbf{u}^z, \mathbf{v}^z \in \mathbb{R}^{n_z}$, we have

$$\begin{aligned} &< W_\ell(\mathbf{u}_\ell^{xy} \otimes \mathbf{u}^z), \mathbf{v}_\ell^{xy} \otimes \mathbf{v}^z > \\ &\qquad =< W_\ell^{A^{xy}} \mathbf{u}_\ell^{xy}, \mathbf{v}_\ell^{xy} >< B^z \mathbf{u}^z, \mathbf{v}^z > + < W_\ell^{B^{xy}} \mathbf{u}_\ell^{xy}, \mathbf{v}_\ell^{xy} >< A^z \mathbf{u}^z, \mathbf{v}^z > , \end{aligned}$$

as required. This is obtained by decomposing $\mathbf{u}_\ell^{xy}$ and $\mathbf{v}_\ell^{xy}$ into a linear combination of an orthonormal basis of $\mathbb{R}^{n_{xy}^\ell}$ and using tensor product rules (5.4.8) and (5.4.9), as done in [14] for a block Jacobi smoother applied to a 2D problem.

Now, as in Section 5.4.1 we aim to use a *block-diagonalisation* procedure to accomplish a separation of coordinate directions. This time, we separate the z-direction from the x- and y-directions and reduce the 3D problem into a problem involving only matrices A_ℓ^{xy}, $W_\ell^{B^{xy}}$ and $W_\ell^{A^{xy}}$. This is done by considering A^z-orthogonal subspaces induced by tensor products of

- eigenvectors $\{\mathbf{e}_j^z\}_{j=1}^{n_z}$ of A^z, and
- arbitrary components on the x–y plane.

Let the vectors $\{\mathbf{e}_j^z\}_{j=1}^{n_z}$ be a basis of V^z which form an B^z-orthonormal generalized eigenbasis of (A^z, B^z), i.e.

$$\left.\begin{aligned} A^z \mathbf{e}_j^z &= \omega_j B^z \mathbf{e}_j^z \\ < B^z \mathbf{e}_j^z, \mathbf{e}_k^z > &= \delta_{jk} \end{aligned}\right\} . \tag{5.4.18}$$

$\{\omega_j\}_{j=1}^{n_z}$ are eigenvalues of (A^z, B^z) that are strictly positive since A^z and B^z are positive definite. If (5.4.18) holds, then we also have

$$< A^z \mathbf{e}_j^z, \mathbf{e}_k^z > = \omega_j \delta_{jk} \,, \tag{5.4.19}$$

so A^z-orthogonality is obtained.

As in Section 5.4.1, we set $A_{\ell,\omega_j}^{xy} = \omega_j B_\ell^{xy} + A_\ell^{xy}$ and $W_{\ell,\omega_j}^{xy} = \omega_j W_\ell^{B^{xy}} + W_\ell^{A^{xy}}$, and we also set

$$S_{\ell,\omega_j}^{xy} = I - \left(W_{\ell,\omega_j}^{xy}\right)^{-1} A_{\ell,\omega_j}^{xy} \,,$$

$$M^{xy}_{\ell,\omega_j} = \left(S^{xy}_{\ell,\omega_j}\right)^{\frac{\nu}{2}} \left[I - P^{xy}_{\ell}\left(A^{xy}_{\ell-1,\omega_j}\right)^{-1} R^{xy}_{\ell} A^{xy}_{\ell,\omega_j}\right] \left(S^{xy}_{\ell,\omega_j}\right)^{\frac{\nu}{2}} + \left(S^{xy}_{\ell,\omega_j}\right)^{\frac{\nu}{2}} \left[P^{xy}_{\ell} M^{xy}_{\ell-1,\omega_j}\left(A^{xy}_{\ell-1,\omega_j}\right)^{-1} R^{xy}_{\ell} A^{xy}_{\ell,\omega_j}\right] \left(S^{xy}_{\ell,\omega_j}\right)^{\frac{\nu}{2}} ,$$

where S^{xy}_{ℓ,ω_j} and M^{xy}_{ℓ,ω_j} are the smoother and the V-cycle iteration matrix for the family of 2D problems. Using these matrices, we state and prove the main theorem of this chapter, which establishes the robustness of the NUMG method in 3D by reducing the problem to a family of 2D problems, for which the uniform convergence is assumed.

Theorem 5.4.3. *(Uniform convergence of the NUMG method in 3D) Suppose that for all $j = 1, \ldots, n_z$,*

$$\|M^{xy}_{F,\omega_j}\|_{A_F} < \delta(\nu) < 1 \tag{5.4.20}$$

holds, where δ is dependant only on the total number of smoothing steps, ν. Then the iteration matrix of the multigrid V-cycle (see Lemma 4.6.9) for the 3D problem, based on a z-line smoother and no coarsening in the z-direction, satisfies

$$\|M^{V}_{F}\|_{A_F} \leq \delta(\nu) < 1\,. \tag{5.4.21}$$

Proof. Using the A^z-orthogonality and B^z-orthonormality of the eigenvectors $\{e^z_j\}$, and the two tensor product rules (5.4.8) and (5.4.9), we deduce that for all $j, k = 1, \ldots, n_z$ and for $\mathbf{u}^{xy}_{\ell},\ \mathbf{v}^{xy}_{\ell} \in \mathbb{R}^{n^{\ell}_{xy}}$,

$$< \mathbf{u}^{xy}_{\ell} \otimes \mathbf{e}^z_k, A_{\ell}(\mathbf{v}^{xy}_{\ell} \otimes \mathbf{e}^z_j) >= \ \delta_{j,k} < \mathbf{u}^{xy}_{\ell}, (A^{xy}_{\ell} + \omega_j B^{xy}_{\ell})\mathbf{v}^{xy}_{\ell} > \ . \tag{5.4.22}$$

which follows the same computations as (5.4.12). Similarly

$$< \mathbf{u}^{xy}_{\ell} \otimes \mathbf{e}^z_k, W_{\ell}(\mathbf{v}^{xy}_{\ell} \otimes \mathbf{e}^z_j) >= \delta_{jk} < \mathbf{u}^{xy}_{\ell}, (W^{A^{xy}}_{\ell} + \omega_j W^{B^{xy}}_{\ell})\mathbf{v}^{xy}_{\ell} > \ .$$

Now, as in Section 5.4.1 for $j = 1, \ldots, n_z$, we define matrices $Q^{(j)}_{\ell} : \mathbb{R}^{n^{\ell}_{xy}} \to \mathbb{R}^{n^{\ell}_{xy} n_z}$ by

$$Q^{(j)}_{\ell}\mathbf{u}^{xy}_{\ell} = \mathbf{u}^{xy}_{\ell} \otimes \mathbf{e}^z_j \ .$$

for $\mathbf{u}^{xy}_{\ell} \in \mathbb{R}^{n^{\ell}_{xy}}$. We write vectors $\mathbf{u}^z \in \mathbb{R}^{n_z}$ as a linear combination of the eigenbases $\{\mathbf{e}^z_j\}_{j=1}^{n_z}$ of $\mathbb{R}^{n_z}$ and by the calculations of (5.4.22) we have for all $j, k = 1, \ldots, n_z$,

$$< \mathbf{u}^{xy}_{\ell}, Q^{(k)T}_{\ell} A_{\ell} Q^{(j)}_{\ell} \mathbf{v}^{xy}_{\ell} >=< \mathbf{u}^{xy}_{\ell} \otimes \mathbf{e}^z_k, A_{\ell}(\mathbf{v}^{xy}_{\ell} \otimes \mathbf{e}^z_j) >= \delta_{ij} < \mathbf{u}^{xy}_{\ell}, A^{xy}_{\ell,\omega_j} \mathbf{v}^{xy}_{\ell} > ,$$

and so we define the blockdiagonal matrix $\hat{A}_{\ell}$ by

$$\hat{A}_\ell = Q_\ell^T A_\ell Q_\ell = blockdiag\left\{A^{xy}_{\ell,\omega_j} : j = 1,\ldots,n_z\right\}.$$

Similarly Q_ℓ transforms W_ℓ into a blockdiagonal matrix:

$$\hat{W}_\ell = Q_\ell^T W_\ell Q_\ell = blockdiag\left\{W^{xy}_{\ell,\omega_j} : j = 1,\ldots,n_z\right\}.$$

For the transfer operator P_ℓ, we perform similar computations to (5.4.22) and obtain, for $j,k = 1,\ldots,n_z$,

$$\begin{aligned} < \mathbf{u}_\ell^{xy}, Q_\ell^{(k)T} P_\ell Q_{\ell-1}^{(j)} \mathbf{v}_{\ell-1}^{xy} > &=< \mathbf{u}_\ell^{xy} \otimes \mathbf{e}_k^z, (P_\ell^{xy} \otimes I)(\mathbf{v}_{\ell-1}^{xy} \otimes \mathbf{e}_j^z) > \\ &= \delta_{j,k} < \mathbf{u}_\ell^{xy}, P_\ell^{xy} \mathbf{v}_{\ell-1}^{xy} >, \end{aligned}$$

for $\mathbf{u}_\ell^{xy} \in \mathbb{R}^{n_{xy}^\ell}$ and $\mathbf{v}_{\ell-1}^{xy} \in \mathbb{R}^{n_{xy}^{\ell-1}}$, using (5.4.8) and (5.4.9). We have $P_\ell = P_\ell^{xy} \otimes I$ because we do not coarsen in the z-direction, and because of the identity matrix the similarity transformation of P_ℓ produces a block-diagonal matrix:

$$\hat{P}_\ell = Q_\ell^T P_\ell Q_{\ell-1} = blockdiag\left\{P_\ell^{xy} : j = 1,\ldots,n_z\right\}.$$

Similarly,

$$\hat{R}_\ell = Q_{\ell-1}^T R_\ell Q_\ell = blockdiag\left\{R_\ell^{xy} : j = 1,\ldots,n_z\right\}.$$

Now we use these similarity transformations to reduce the convergence analysis of the 3D problem to the analysis of problems on each of the invariant subspaces that correspond to a family of 2D problems.

Since Q_ℓ is non-singular and orthogonal for $\ell = 1,\ldots,F$, we have

$$\begin{aligned} Q_F^{-1} M_F^V Q_F &= Q_F^{-1} S_F^{\frac{\nu}{2}} (I - P_F A_{F-1}^{-1} R_F A_F) S_F^{\frac{\nu}{2}} Q_F + Q_F^{-1} S_F^{\frac{\nu}{2}} (P_F M_{F-1}^V A_{F-1}^{-1} R_F A_F) S_F^{\frac{\nu}{2}} Q_F \\ &= blockdiag\{M^{xy}_{F,\omega_j} : j = 1,\ldots,n_z\}. \end{aligned}$$

The matrices M^{xy}_{F,ω_j} satisfy

$$\|Q_F^{-1} M_F^V Q_F\|_2 = max\{\|M^{xy}_{F,\omega_j}\|_2 : j = 1,\ldots,n_z\},$$

and since the multiplication by orthogonal matrices Q_F or Q_F^{-1} does not change the spectral norm of a matrix (cf. [74, Chapter 2]), we have

$$\|M_F^V\|_2 = max\{\|M^{xy}_{F,\omega_j}\|_2 : j = 1,\ldots,n_z\} \le \delta(\nu) < 1,$$

by (5.4.20). Hence the result (5.4.21) is obtained. □

We now discuss the conditions in which (5.4.20) holds for a family of 2D problems on the x–y plane. We know from Theorem 5.4.2 that the 2D iteration matrix of the multigrid V-cycle, based on a line smoother in one coordinate direction and coarsening only in the other, will converge for 2D problems with grid-aligned anisotropies bounded by a contraction number $\delta(\nu) = C_A/(C_A + \nu)$. Hence, this is one possible condition for which (5.4.20) will hold.

In addition we would also like to show that the condition will hold when *conditional semi-coarsening* and a point smoother are used, since we have shown experimentally in Section 5.2.2 that this is the most efficient method for solving the 2D model problem (5.1.1). Unfortunately, a rigorous theoretical proof for the robustness of this method is missing. However, since we have shown this to be true both heuristically (cf. Section 5.2.1) and experimentally (cf. Section 5.2.2), there is strong evidence to support the statement that the 2D multigrid V-cycle based on conditional semi-coarsening and a point smoother is robust with respect to problem size and grid anisotropies.

Finally, we apply the theory to problem (5.3.1), i.e. with $\alpha_1 = 1$, $\alpha_2 = \sin^2(\pi y)$, $\alpha_3 = L_z \sin^2(\pi y)$ and $\Omega = (0,1) \times (\varepsilon, 1-\varepsilon) \times (0,1)$ for $\varepsilon \ll 1$. For L_z, we can use any of $0 < L_z$ as in the experiments in Section 5.3. Regardless of the value of L_z, the use of z-line relaxation and conditional semi coarsening reduces the analysis of the full problem to that of a family of 2D problems on the $x - y$ plane of the form (5.1.2). Experimental results for these 2D problems have already been carried out to show the robustness of the 2D NUMG method (see Table 5.5). Hence, assuming this robustness, we use Theorem 5.4.3 to deduce that the 3D NUMG method applied to (5.3.1) is optimal.

5.5 Non-Uniform Multigrid in Spherical Geometries

Having established the optimality of the NUMG method – both experimentally and theoretically – on model problems with inhomogeneous anisotropy, we now apply the method to problems of greater interest to the Met Office, i.e. anisotropic problems in spherical polar coordinates. We will begin by investigating the 2D Poisson-type equations in spherical geometries, which is similar to model problem (5.1.2), but with an added difficulty due to the singularity of the equation at the poles. We then investigate the 3D Poisson-type equations which are similar to model problem (5.3.1). In both cases the performance of NUMG will be compared against implementations of AMG as well as against existing solvers that are currently used at the Met Office to solve these type of problems.

5.5.1 Poisson-Type Equations on the Unit Sphere

In this section we solve the Poisson-type equation in spherical coordinates using the techniques developed in Section 5.1. The equation, in a nondimensionalized form (see Section 3.2.4) is:

$$-\frac{\partial}{\partial\phi}\left(\frac{L_\phi(\lambda)\sin(\pi\phi)}{\pi^2}\frac{\partial u}{\partial\phi}\right) - \frac{\partial}{\partial\lambda}\left(\frac{L_\lambda(\phi)}{4\pi^2\sin(\pi\phi)}\frac{\partial u}{\partial\lambda}\right) = g_{2D}\sin(\pi\phi)\,, \qquad (5.5.1)$$
$$\Omega = (0,1)^2.$$

Note that we do not have uniform ellipticity of the operator here, since we allow $\phi = 0, 1$ (i.e. equivalent to setting $\varepsilon = 0$ in (5.1.2)). This is solved with periodic boundary conditions at the boundary coinciding with $\lambda = 0$ and $\lambda = 1$, and a polar boundary condition at $\phi = 0$ and $\phi = 1$. The equation can be written in the form (5.1.1), with $\alpha_1(\phi,\lambda) = L_\phi(\lambda)\sin(\pi\phi)/\pi^2$ and $\alpha_2(\phi,\lambda) = L_\lambda(\phi)/(4\pi^2\sin(\phi))$. The finite volume discretisation of problem (5.5.1), given in Section 3.2.4, on the spherical polar grid introduced in Section 3.2.1 yields a matrix $A_\ell \in \mathbb{R}^{n^{(\ell)}\times n^{(\ell)}}$ with the following stencil

$$\begin{bmatrix} & -\frac{h_\lambda^{(\ell)}}{h_\phi^{(\ell)}}\frac{L_\phi(\lambda_k)}{\pi^2}\sin(\pi\phi_{j+\frac{1}{2}}) & \\ -\frac{h_{\phi,j}^{(\ell)}}{h_\lambda^{(\ell)}}\frac{L_\lambda(\phi_j)}{4\pi^2\sin(\pi\phi_j)} & -\sum & -\frac{h_{\phi,j}^{(\ell)}}{h_\lambda^{(\ell)}}\frac{L_\lambda(\phi_j)}{4\pi^2\sin(\pi\phi_j)} \\ & -\frac{h_\lambda^{(\ell)}}{h_\phi^{(\ell)}}\frac{L_\phi(\lambda_k)}{\pi^2}\sin(\pi\phi_{j-\frac{1}{2}}) & \end{bmatrix},$$

at all nodes expect the nodes at the poles, i.e. the nodes $\{(\phi_j,\lambda_k) : j = 1,\ldots,n_\phi^{(\ell)},\ k = 1,\ldots,n_\lambda^{(\ell)}\}$, where j and k represent the mesh lines of latitude and longitude respectively and $n_\phi^{(\ell)}$ and $n_\lambda^{(\ell)}$ are the nodes in the ϕ- and λ-directions respectively. The dimension of the problem is $n^{(\ell)} = (n_\phi^{(\ell)} \times n_\lambda^{(\ell)}) + 2$, with the two additional degrees of freedom corresponding to the pole nodes. $\mathcal{T}_\ell$ denotes the sequence of grid levels, where $\ell = 1,\ldots,F$, with $\mathcal{T}_F$ being the grid with the finest resolution of points. The system of equations we solve on $\mathcal{T}_F$ is

$$A_F\mathbf{u}_F = \mathbf{b}_F\,, \qquad (5.5.2)$$

where $\mathbf{b}_F$ is a vector containing the finite volume discretisation of $g_{2D}\sin(\pi\phi)$ at each grid point (including the poles). The coarse grid operators A_ℓ, $\ell = 1,\ldots F-1$ are all obtained via an identical finite volume discretisation of (5.5.1) on $\mathcal{T}_\ell$.

Now, let us firstly consider the case $L_\phi = L_\lambda = 1$, i.e. the Poisson equation. We assume that the fine grid is uniform, but in typical grid resolutions used a the Met Office, the number of grid points in the λ-direction is approximately double those in

the ϕ-direction. Thus, since the grid is defined on the unit square, we have $h_\phi \approx 2h_\lambda$. The stencil is comparable to that of (5.1.3) for model problem (5.1.2), with a variation in anisotropy only in the polar (ϕ) direction. We observe a stronger anisotropy near the poles caused by a degenerating ϕ-coefficient of the differential operator, hence the conditional semi-coarsening strategy from Section 5.2 can also be used here. Recall that the strategy relies on the ratio of the off-diagonal entries of the stencil, which is approximately $\left(\frac{2h_\lambda}{h_\phi}\right)^2 \sin^2(\pi\phi_j)$, and that for each j-line the type of coarsening depends on whether the ratio is greater than $\frac{1}{2}$. There is no variation of anisotropy in the λ-direction, so the grid is uniformly fully coarsened in this direction.

Algorithm 5.2 is the full algorithm for conditional semi-coarsening on problem (5.5.1), and is only a small modification to that of algorithm 5.1.

Algorithm 5.2 Conditional semi-coarsening for the Poisson-type equation on the sphere: `Cond_spherical`$(h_\lambda^{(\ell)}, h_\phi^{(\ell)}, n_\lambda^{(\ell)}, n_\phi^{(\ell)}, \ell, h_\lambda^{(\ell-1)}, h_\lambda^{(\ell-1)}, n_\lambda^{(\ell-1)}, n_\phi^{(\ell-1)})$

$incr = 1$ (incrementing up the ϕ-line)
for $j = 1,\ n_\phi^{(\ell)}$
  $ratio = (L_\phi/L_\lambda)(2h_\lambda^{(\ell)}/h_{\phi,incr}^{(\ell)})^2 \sin^2(\pi\phi_{incr})$ (ratio at line $incr$)
  if $ratio \geq 0.5$ **then**
    $h_{\phi,j}^{(\ell-1)} = h_{\phi,incr}^{(\ell)} + h_{\phi,incr+1}^{(\ell)}$ (mesh width doubled in
    $incr = incr + 2$ ϕ-direction at line $incr$)
  else
    $h_{\phi,j}^{(\ell-1)} = h_{\phi,incr}^{(\ell)}$ (mesh width unchanged in
    $incr = incr + 1$ ϕ-direction at line $incr$)
  end if
  if $incr > n_\phi^{(\ell)}$ **exit**
end for
$h_\lambda^{(\ell-1)} = 2h_\lambda^{(\ell)}$ (mesh width doubled in λ-direction on grid $\mathcal{T}_{\ell-1}$)
$n_\phi^{(\ell-1)} = j$ (no. of grid points in λ- and ϕ-
$n_\lambda^{(\ell-1)} = n_\lambda^{(\ell)}/2$ directions on grid $\mathcal{T}_{\ell-1}$)

We solve the Poisson Equation on the sphere using the machine used for the experiments in Sections 5.1 and 5.2 and the same multigrid components as outlined at the start of Section 5.1.1. However, we must also note that in 2D the finite volume discretisation of problem (5.5.1) on the grid introduced in Section 3.2.1 results in a singular system of linear equations with the nullspace of A_F spanned by the constant vector. Therefore, we require a compatibility condition on g_{2D} and we need to regularize the problem. We discuss the technique for doing this in the following section.

5.5.2 Dealing with the Singularity of the Problem

The matrices A_ℓ, $\ell = 1, \ldots, F$, are rank-deficient by one and their null space is spanned by the constant vector $\mathbf{1}_\ell$, which is a vector of ones of length $n^{(\ell)}$. Provided the right-hand-side vector on each grid level, denoted $\mathbf{b}_\ell$, is in the range of the operator (i.e. $\mathbf{b}_\ell \in \ range(A_\ell)$), an iterative solver such as the conjugate gradient method will find the solution to the problem despite the matrix being singular. Thus, we project $\mathbf{b}_\ell$ onto $range(A_\ell)$ on each grid level to ensure the existence of a solution.

In order to do this, we use the following relation from [74] which relates the range and null space of A:

$$range(A_\ell) = null(A_\ell^T)^\perp =: \mathbf{1}_\ell^\perp \,,$$

where $A_\ell = A_\ell^T$ so $null(A_\ell) = null(A_\ell^T)$. From this result we deduce that the projection onto $range(A)$ is accomplished by eliminating all components in the direction $\mathbf{1}_\ell$, and this is done by the following orthogonal projector

$$Q_\ell = I_\ell - \frac{\mathbf{1}_\ell \mathbf{1}_\ell^T}{< \mathbf{1}_\ell, \mathbf{1}_\ell >} = I_\ell - \frac{1}{n^{(\ell)}} \mathbf{1}_\ell \mathbf{1}_\ell^T, \quad \ell = 1, \ldots, F$$

where Q_ℓ and I_ℓ are the projection operator and identity matrix on $\mathcal{T}_\ell$, respectively. Q_ℓ is applied to the right-hand-side on $\mathcal{T}_\ell$ which ensures $Q_\ell \mathbf{b}_\ell \in \ range(A_\ell)$, $\forall \ell = 1, \ldots, F$.

In theory this projection is superfluous if the right hand side g_{2D} satisfies the following compatibility condition

$$\int_\Omega g_{2D} \ dV = 0 \,.$$

The solution is then determined up to a constant. However, even if this compatibility condition is satisfied for the continuous right hand side, g_{2D}, it is not necessarily the case that the discrete compatibility condition

$$< \mathbf{b}_F, \mathbf{1}_F >= 0 \,, \tag{5.5.3}$$

is automatically satisfied. The reason for this is because of inexact arithmetic within the discretisation. Therefore $\mathbf{b}_F$ must be projected onto the range of A_F if (5.5.3) doesn't hold. Similarly, (5.5.3) must hold on each of the coarse grids. In exact arithmetic, the right-hand-side $\mathbf{b}_\ell$ on the coarser grids would be in $range(A_\ell)$ by definition, thus satisfying (5.5.3), but due to the rounding errors this is not the case in practice so the projection is in fact necessary on each grid.

Remark 5.5.1 (Other methods of dealing with the singularity)**.** The projection of the right-hand-side is not the only approach that can be used to deal with the singularity

of the problem. Here we discuss two other approaches that have been tested, but they lead to exactly the same number of multigrid V-cycle iterations:

Block Elimination

By ordering the degrees of freedom such that the final two rows of matrix A_F correspond to the pole nodes, we can write the matrix as

$$A_F = \begin{bmatrix} \widetilde{A} & B \\ B^T & C \end{bmatrix},$$

with a full rank, sparse matrix $\widetilde{A} \in \mathbb{R}^{n^{(F)} \times n^{(F)}}$. $B \in \mathbb{R}^{n^{(F)} \times 2}$ contains only two columns irrespective of mesh refinement, thus we can solve (5.5.2) by block elimination by writing it in the following form:

$$A_F = \begin{bmatrix} \widetilde{A} & B \\ B^T & C \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{u}} \\ \tilde{\mathbf{v}} \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{f}} \\ \tilde{\mathbf{g}} \end{bmatrix}.$$

We first solve for a matrix $X \in \mathbb{R}^{n^{(F)} \times 2}$ and a vector $\mathbf{y} \in \mathbb{R}^{n^{(F)}}$ satisfying

$$\widetilde{A}X = B \quad \text{and} \quad \widetilde{A}\mathbf{y} = \tilde{\mathbf{f}}. \tag{5.5.4}$$

Then $\tilde{\mathbf{v}}$ is a solution to

$$(C - B^T X)\tilde{\mathbf{v}} = \tilde{\mathbf{g}} - B^T \mathbf{y},$$

which is a 2×2 system of rank 1 (which can easily be solved directly). Thus the solution $\tilde{\mathbf{v}}$ is unique up to a constant only. Finally we obtain $\tilde{\mathbf{u}}$ by calculating

$$\tilde{\mathbf{u}} = \mathbf{y} - X\tilde{\mathbf{v}}.$$

However, (5.5.4) consists of solving three systems with $\widetilde{A}$ using multigrid, as opposed to one system with A_F in the projection technique above.

Fixing the Solution at one of the Poles

Another method is to fix the solution at one of the poles, reducing the problem to solving only the following system

$$\hat{A}\hat{\mathbf{u}} = \hat{\mathbf{b}},$$

where $\hat{A} \in \mathbb{R}^{n^{(F)}-1 \times n^{(F)}-1}$ takes the first $n^{(F)} - 1$ rows and $n^{(F)} - 1$ columns of A_F. Similarly, $\hat{\mathbf{b}}$ takes the first $n^{(F)} - 1$ entries of $\mathbf{b}_F$. The singularity is eliminated by removing a degree of freedom at a pole, therefore the resulting solution is unique.

5.5.3 Numerical Results

The experiments carried out in this section are done using the same settings as in Section 5.2.2, in addition applying the projection of the right-hand-side on each grid. The results of this experiment are given in Table 5.13. The setup time and solve time increases linearly with the problem size and the number of iterations and the average convergence factor remains constant, indicating that the multigrid method is robust with respect to problem size, and performes optimally. Without the projection on each grid level, the multigrid method did not converge because the rounding errors caused the right-hand-side vectors on each coarse grid to lie outside the range of the operator, hence (5.5.3) was not satisfied.

Problem size	# Coarse grids	Setup time (s)	Solve time (s)	N_{its}	μ_{avg}
32x16	2	1.23E-3	2.27E-3	9	0.102
64x32	3	5.12E-3	8.63E-3	9	0.114
128x64	4	1.80E-2	3.61E-2	9	0.118
256x128	5	6.86E-2	1.70E-1	9	0.118
512x256	6	2.67E-1	7.86E-1	9	0.119

Table 5.13: Two dimensional Poisson's equation on the unit sphere solved using NUMG (with a projection onto the range of A_F in each iteration). CPU time in seconds.

Now, consider equation (5.5.1) with $L_\lambda \neq 1$ and $L_\phi \neq 1$. Let us look at what an additional anisotropy introduced by changing L_ϕ and L_λ does to the method. As long as $L_\lambda \geq L_\phi$ is satisfied, NUMG is expected to retain optimality because the direction of the strong coupling (i.e. the λ direction) will not change throughout the domain, which is an important factor in conditional semi-coarsening as discussed in Section 5.2. However, if $L_\lambda < L_\phi$, then the direction of the strong coupling will change direction near the equator and so additional coarsening strategies would need to be considered to retain optimality. We have already seen examples of this earlier in Sections 5.3.2 and 5.3.4 for the 3D model problem, where the changes in the direction of strong coupling resulted in a modification of the standard techniques in order to gain optimality of the method. The purpose of these tests is to show how the coarse grids adapt to the ratio $\frac{L_\phi}{L_\lambda}$. The smaller the ratio, the stronger the anisotropy becomes even at the equator, so the coarse grids will have to compensate for this by increasing the region in which semi coarsening occurs. The case $\frac{L_\phi}{L_\lambda} = 1$ has already been tested, and we compare this with cases $\frac{L_\phi}{L_\lambda} = 10^{-1}$, $\frac{L_\phi}{L_\lambda} = 10^{-2}$ and $\frac{L_\phi}{L_\lambda} = 10^{-4}$. We also test $\frac{L_\phi}{L_\lambda} = 10$ to demonstrate that the method requires modifications when $L_\lambda < L_\phi$.

For the case $\frac{L_\phi}{L_\lambda} = 1$, we know already that the problem is isotropic at the equator and strongly anisotropic at the poles, and so the first coarse grid is evenly partitioned

into fully coarsened and semi coarsened regions (cf. Figure 5-1). However, as the ratio of L_ϕ and L_λ is reduced, the anisotropy becomes stronger by a factor of $\frac{L_\phi}{L_\lambda}$ in all regions, and the problem will no longer be isotropic at the equator. Therefore the first coarse grid will have coarsening only in the λ-direction, i.e. uniform semi coarsening everywhere. If the value of $\frac{L_\phi}{L_\lambda}$ is sufficiently close to 1, successive coarse grids may have regions where the grid is fully coarsened, but these regions will not be as pronounced as when $\frac{L_\phi}{L_\lambda} = 1$.

We demonstrate this for a 512x256 grid. Recall that the ratio of the off-diagonal entries on line j for problem (5.5.1) is approximately $(2h_\lambda/h_{\phi,j})^2 \sin^2(\pi\phi_j)$. For (3.2.13), however, this is compensated by a factor L_ϕ/L_λ. At the equator, $\sin^2(\pi\phi_j) = 1$ so the ratio is

$$\frac{L_\phi}{L_\lambda}\left(\frac{2h_\lambda}{h_{\phi,j}}\right)^2.$$

Firstly, setting $\frac{L_\phi}{L_\lambda} = 10^{-1}$, we measure the ratio at the equator on each grid:

fine grid	0.1	(less than 0.5 so semi coarsening),
first coarse	0.4	(less than 0.5 so semi coarsening),
second coarse	1.6	(greater than 0.5 so full coarsening).

Therefore full coarsening occurs near the equator only from the third coarse grid onwards. Likewise for $\frac{L_\phi}{L_\lambda} = 10^{-2}$ we have

fine grid	0.01	(less than 0.5 so semi coarsening),
first coarse	0.04	(less than 0.5 so semi coarsening),
second coarse	0.16	(less than 0.5 so semi coarsening),
third coarse	0.64	(greater than 0.5 so full coarsening),

and so full coarsening only occurs at the equator from the fourth coarse grid onwards. For $\frac{L_\phi}{L_\lambda} = 10^{-4}$, however, full coarsening does not occur on any grid we used because the anisotropy is so strong even at the equator. Therefore non-uniform multigrid manages to adapt to the strength of anisotropy by changing the structure of the coarse grids, and Table 5.14 shows that it performs optimally regardless of the strength of anisotropy. However, this is only true as long as the direction of the strong coupling never changes. Note that for $\frac{L_\phi}{L_\lambda} = 10$, where the direction of strong coupling switches between the poles and the equator, the results in Table 5.14 show that the method is not optimal, although we still observe good convergence rates. However, luckily such a situation never occurs in any of the problems of interest to the Met Office.

	$L_\phi/L_\lambda = 10^{-1}$		$L_\phi/L_\lambda = 10^{-2}$	
Problem size	N_{its}	Total time	N_{its}	Total time
32x16	7	4.05E-3	7	4.68E-3
64x32	7	1.40E-2	7	1.66E-2
128x64	8	6.02E-2	7	6.06E-2
256x128	8	2.59E-1	8	2.75E-1
512x256	8	1.14	8	1.20
	$L_\phi/L_\lambda = 10^{-4}$		$L_\phi/L_\lambda = 10$	
Problem size	# Iterations	Total time	# Iterations	Total time
32x16	7	5.16E-3	12	4.28E-3
64x32	7	2.07E-2	13	1.58E-2
128x64	7	8.16E-2	14	6.87E-2
256x128	7	3.43E-1	14	3.03E-1
512x256	7	1.39	16	1.54

Table 5.14: Two dimensional Poisson's equation on the unit sphere with varying anisotropy solved using NUMG (CPU time in seconds).

5.5.4 The Galerkin Approach

Thus far, all the methods used have been set up with the sequence of matrices A_ℓ computed via a finite volume discretisation on each grid. Here we compute each coarse grid operator by the *Galerkin Product*, i.e. $A_{\ell-1} = R_\ell A_\ell P_\ell$ for $\ell = 1, \ldots, F$ (see Section 4.4.2 and [75, §2.3.2]). This is the usual situation in algebraic multigrid. The Poisson equation (5.5.1) on the unit sphere is solved using a standard multigrid V-cycle with pointwise Gauss−Seidel smoother combined with conditional semi-coarsening. We use linear interpolation and full weighting restriction and the stopping criterion is 10^{-8}.

We test this method because the majority of multigrid convergence theory relies on using the Galerkin product to form the coarse grid operators. The immediate issue that arises in practice is that, in the case of linear interpolation and full weighting restriction, the coarse grid operator created by the Galerkin product $A_{F-1} = R_F A_F P_F$ has extra fill-in, that is, it has more non-zero entries per row than a direct finite volume discretisation on the same grid would have. In fact, A_{F-1} computed via the Galerkin product has a 21-point stencil (see Figure 5-5).

The operator computed on each successively coarser grid (i.e. $\mathcal{T}_1, \ldots, \mathcal{T}_{F-1}$) will also have a 21-point stencil. This means more work is needed on the coarse grids when the matrix-vector multiplications and smoother are used. In addition, the setup cost will increase because the computation of each coarse grid operator via the multiplication of three matrices will be more expensive than by directly discretising on each grid. The results from Table 5.15 confirm this claim, with particularly the setup time going up considerably compared to the results in Table 5.13, by a factor of up to six. The extra

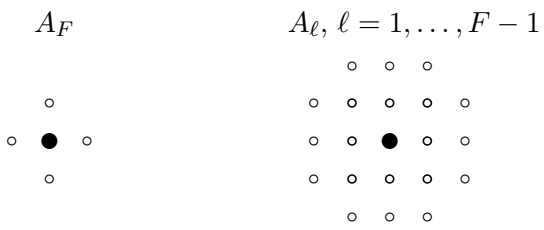

Figure 5-5: A_F has a 5-point stencil and A_ℓ, for $\ell = 1, \ldots, F-1$ have a 21-point stencil, the extra fill-in created as a result of the Galerkin Product

fill-in causes further problems in three dimensions which will be discussed in Section 5.6.1. However, one non-negligible benefit is that the denser stencils on the coarse grids lead to a significantly smaller number of iterations, five instead of nine, resulting in a faster solve time by a factor of approximately 1.5. The method is also fully robust to grid refinement.

Problem size	# Coarse grids	Setup time (s)	Solve time (s)	# Iterations	μ_{avg}
32x16	2	3.17E-3	1.53E-3	5	0.012
64x32	3	1.39E-2	5.91E-3	5	0.011
128x64	4	6.36E-2	2.54E-2	5	0.011
256x128	5	3.03E-1	1.21E-1	5	0.012
512x256	6	1.59	5.53E-1	5	0.012

Table 5.15: Two dimensional Poisson's equation on the unit sphere solved using NUMG, the Galerkin product and full-weighting restriction

The easiest way to reduce the fill-in is to simplify the restriction operator to the four-point average restriction (see Section 4.4.2 and [75, §2.8.4]). This is in fact the same order of accuracy as full weighting restriction (both second order accurate, as stated in [75] and Remark 4.4.1), so the required condition from Remark 4.4.1 is still satisfied. However, the symmetry of the multigrid iteration will be lost because $R_\ell \neq P_\ell^T$. This yields a more sparse 9-point stencil for the operators on all coarse levels, but the coarse grid matrices are non-symmetric so a different coarse grid solver is needed. We choose Bi-CGSTAB [76] as together with GMRES [65] it is one of the fastest Krylov methods for non-symmetric and sparse linear systems. The results of using a four point average restriction are in Table 5.16. We evidently observe a significant speedup from Table 5.15 in the setup, but it is still considerably slower than using finite volume discretisations on each grid, due the cost of the Galerkin product in the setup phase. Also the solve time is slower compared to the results in Table 5.15 due to the additional cost in using Bi-CGSTAB instead of CG and because of the larger number of iterations required.

Problem size	# Coarse grids	Setup time (s)	Solve time (s)	# Iterations	μ_{avg}
32x16	2	1.92E-3	1.86E-3	6	0.041
64x32	3	7.48E-3	6.68E-3	6	0.039
128x64	4	3.52E-2	3.12E-2	7	0.056
256x128	5	1.85E-1	1.45E-1	7	0.070
512x256	6	1.13	6.91E-1	7	0.071

Table 5.16: Two dimensional Poisson's equation on the unit sphere solved using NUMG, the Galerkin product and four-point averaging restriction

The experimental results have therefore shown that from a point of view of computational cost, it is not preferable to use the Galerkin product to set up the matrices A_ℓ, $\ell = 1, \ldots, F-1$ on each coarse grid level, despite the method still performing optimally. This indicates how efficient the NUMG solver given in Table 5.13 is.

5.5.5 Comparison with Algebraic Multigrid (AMG)

The success of non-uniform (conditional) coarsening strategies for anisotropic elliptic problems has already been demonstrated in the literature, such as [71, 63, 18], by the highly successful algebraic multigrid (AMG) methods. However, as discussed and demonstrated in Section 5.1.2, its setup cost is expected to be significantly larger than that of geometric methods as a consequence of the graph-based coarse grid selection and interpolation algorithms and the need to use the Galerkin product to compute the coarse grid matrices. Therefore, for problems that can be solved using geometric multigrid, it is typically less efficient to use AMG.

To test this assertion, problem (5.5.1) is solved using `BoomerAMG` as a preconditioner to the conjugate gradient method (CG + AMG) and as a stand alone solver (V-cycle only). As in Section 5.1.2, the default settings for `BoomerAMG` were used. However, in order to solve this problem using `BoomerAMG`, a degree of freedom had to be removed at the north pole to remove the singularity. Table 5.17 shows the results and clearly neither approach achieves a similar performance to the NUMG solver from Table 5.13. On the finest grid resolution 512×256, NUMG is faster in total by a factor 2.5.

		V-cycle only			CG + AMG		
Problem size	Setup time	Solve time	# Its	μ_{avg}	Solve time	# Its	μ_{avg}
32x16	4.52E-3	5.71E-3	8	0.088	6.90E-3	5	0.020
64x32	1.18E-2	2.49E-2	10	0.149	2.17E-2	6	0.028
128x64	3.85E-2	1.23E-1	12	0.209	9.55E-2	7	0.062
256x128	1.59E-1	5.50E-1	12	0.218	7.52E-1	7	0.060
512x256	6.62E-1	2.56	13	0.242	1.81	7	0.076

Table 5.17: `BoomerAMG` used to solve model problem (5.5.1). CPU times in seconds.

5.6 Extension to 3D Elliptic Problems in NWP

In this section we use the techniques developed in Section 5.3 to motivate the multigrid method for 3D Poisson-type equations on a spherical shell and then more specifically for the Quasi-Geostrophic Omega equation (2.2.16) and the Helmholtz problem (2.1.24).

Let us first recall from Section 3.1 the general nondimensionalized 3D problem (3.1.2) on a spherical shell, with a zeroth order term also included:

$$-\frac{\partial}{\partial r}\left(\alpha_1(\boldsymbol{\xi})\frac{\partial u}{\partial r}\right)-\frac{\partial}{\partial \phi}\left(\alpha_2(\boldsymbol{\xi})\frac{\partial u}{\partial \phi}\right)-\frac{\partial}{\partial \lambda}\left(\alpha_3(\boldsymbol{\xi})\frac{\partial u}{\partial \lambda}\right)+c(\boldsymbol{\xi})u=f(\boldsymbol{\xi}) \qquad (5.6.1)$$
$$\text{on} \quad \Omega=(0,1)^3\,,$$

with the stencil (assuming a uniform mesh in the ϕ- and λ-directions only):

$$-\alpha_1(\boldsymbol{\xi}_{i-\frac{1}{2},j,k})\frac{h_\lambda h_\phi}{h^-_{r,i}}\begin{bmatrix} & -\alpha_2(\boldsymbol{\xi}_{i,j+\frac{1}{2},k})\frac{h_\lambda h_{r,i}}{h_\phi} & \\ -\alpha_3(\boldsymbol{\xi}_{i,j,k-\frac{1}{2}})\frac{h_\phi h_{r,i}}{h_\lambda} & -\sum & -\alpha_3(\boldsymbol{\xi}_{i,j,k+\frac{1}{2}})\frac{h_\phi h_{r,i}}{h_\lambda} \\ & -\alpha_2(\boldsymbol{\xi}_{i,j+\frac{1}{2},k})\frac{h_\lambda h_{r,i}}{h_\phi} & \end{bmatrix}-\alpha_1(\boldsymbol{\xi}_{i+\frac{1}{2},j,k})\frac{h_\lambda h_\phi}{h^+_{r,i}}\,,$$

where $h^\pm_{r,i}=\frac{1}{2}(h_{r,i}+h_{r,i\pm1})$ and the coefficients are

$$\alpha_1(\boldsymbol{\xi}_{i\pm\frac{1}{2},j,k})=\alpha_1(r_{i\pm\frac{1}{2}},\phi_j,\lambda_k)=L_r(\phi_j,\lambda_k)(a+dr_{i\pm\frac{1}{2}})^2\sin(\pi\phi_j)/d^2\,, \qquad (5.6.2)$$

$$\alpha_2(\boldsymbol{\xi}_{i,j\pm\frac{1}{2},k})=\alpha_2(r_i,\phi_{j\pm\frac{1}{2}},\lambda_k)=L_\lambda(r_i,\lambda_k)\sin(\pi\phi_{j\pm\frac{1}{2}})/\pi^2\,, \qquad (5.6.3)$$

$$\alpha_3(\boldsymbol{\xi}_{i,j,k\pm\frac{1}{2}})=\alpha_3(r_i,\phi_j,\lambda_{k\pm\frac{1}{2}})=L_\phi(r_i,\phi_j)/(4\pi^2\sin(\pi\phi_j))\,. \qquad (5.6.4)$$

Recall that $a\approx 6371km$ and $d\approx 63km$ are the Earth's radius and depth of the atmosphere respectively. We enforce L_r, L_ϕ and L_λ to be separable, i.e. $L_r(\phi,\lambda)=L^\phi_r(\phi)L^\lambda_r(\lambda)$, which will ensure that α_i, $i\in\{1,2,3\}$ are all separable too, i.e. $\alpha_1=\alpha^1_1(r)\alpha^2_1(\phi)\alpha^3_1(\lambda)$. In addition, if we set $L^r_\phi(r)=L^r_\lambda(r)$, then the coefficients α_2 and α_3 will have the same r dependency, i.e. $\alpha^1_2(r)=\alpha^1_3(r)$, which means that problem (5.6.1) is amenable to the convergence theory from Section 5.4.2, based on a r-line smoother and no coarsening in r.

The boundary conditions of (5.6.1) are periodic at $\lambda=0$ and $\lambda=1$, polar at $\phi=0$ and $\phi=1$ and either Dirichlet or Neumann at $r=0$ and $r=1$. If $c=0$ and Neumann boundary conditions are imposed, then the discretisation of (5.6.1) will result in a singular system with the solution being unique only up to a constant, hence projection techniques will have to be enforced as described in Section 5.5.2. Firstly we consider the case $L_\lambda(\phi,r)=L_\phi(\lambda,r)=L_r(\lambda,\phi)=1$ and $c=0$, ie. the three dimensional Poisson equation in spherical coordinates. Note that with $c>0$ this becomes a Helmholtz-type problem, and the greater the value of c the better conditioned the problem becomes.

Hence we consider the only worst case with $c = 0$ until Section 5.6.5. With a and d as above, (5.6.1) is similar to problem (5.3.1) with $L_z = 10^4$, where the off diagonal entry the radial direction is much larger than those of the ϕ- and λ-directions. In addition the mesh is graded in the radial direction (but uniform in the ϕ- and λ-directions) with a smaller grid spacing near the lower boundary to obtain a higher resolution in the regions of most interest. In typical computations at the MET Office, the grid size is $216 \times 163 \times 70$ and so the mesh widths on the unit square are $h_\lambda = O(10^{-3})$ and $h_\phi = O(10^{-3})$, whereas in the radial direction we have a graded mesh with

$$h_{r,1} = \mathcal{O}(10^{-4}) \;\leq\; h_{r,i} \;\leq\; h_{r,n_r} = \mathcal{O}(10^{-1}),$$

where $h_{r,1}$ is the mesh width closest the earth's surface and h_{r,n_r} is the mesh width closest to the upper boundary. Thus we have that $h_{r,i} < h_\lambda$ and $h_{r,i} < h_\phi$ only near the surface of the Earth, i.e. for small i. This graded mesh width in fact causes the direction of strongest coupling to change at different entries. We have that $(a+dr)^2/d^2 = \mathcal{O}(10^4)$, and near the equator we have $\sin(\pi\phi) = \mathcal{O}(1)$. Thus at the equator, the relative sizes of the entries can be approximately calculated as:

Top of atmosphere Surface of Earth

$$-L_r10^1 \begin{bmatrix} & -L_\phi 10^{-2} & \\ -L_\lambda 10^{-2} & -\sum & -L_\lambda 10^{-2} \\ & -L_\phi 10^{-2} & \end{bmatrix} -L_r10^1 \quad -L_r10^4 \begin{bmatrix} & -L_\phi 10^{-5} & \\ -L_\lambda 10^{-5} & -\sum & -L_\lambda 10^{-5} \\ & -L_\phi 10^{-5} & \end{bmatrix} -L_r10^4 \tag{5.6.5}$$

so with $L_r = L_\phi = L_\lambda = 1$, the off-diagonal entry in the radial direction is always the largest. However, near the poles, where $\sin(\pi\phi) \approx \mathcal{O}(10^{-2})$, the entries are:

Top of atmosphere Surface of Earth

$$-L_r10^{-1} \begin{bmatrix} & -L_\phi 10^{-4} & \\ -L_\lambda 10^{0} & -\sum & -L_\lambda 10^{0} \\ & -L_\phi 10^{-4} & \end{bmatrix} -L_r10^{-1} \quad -L_r10^2 \begin{bmatrix} & -L_\phi 10^{-7} & \\ -L_\lambda 10^{-3} & -\sum & -L_\lambda 10^{-3} \\ & -L_\phi 10^{-7} & \end{bmatrix} -L_r10^2 \tag{5.6.6}$$

and so the off-diagonal entry in the λ-direction becomes the largest for entries that are both near the poles and towards the top of the atmosphere. However, for the majority of entries, the r off-diagonal entry is still in general the largest.

The solution of these Poisson- and Helmholtz-type problems on the sphere are of great importance in numerical weather forecasting, but the traditional solvers, such as the ones employed at the MET Office, are coping well with this problem, as we shall see below. We have seen that for most nodes in the domain, there is a large off-diagonal

entry in the radial direction in comparison to those of the λ and ϕ off-diagonal entries, meaning that the anisotropy is very large. Thus, as we did for solving (5.3.1), we use r–line relaxation, namely r–line Gauss–Seidel, and combine this with full-coarsening in the r-direction. We see in Table 5.18 that in combination with the conditional semi-coarsening strategy in the $\lambda - \phi$ plane this is an extremely efficient method, despite the r off-diagonal entry not being the largest in all parts of the domain, with similar results to those from Table 5.7.

Problem size	# Coarse grids	Setup time (s)	Solve time (s)	# Iterations	μ_{avg}
32x16x8	2	5.05E-3	4.37E-3	1	–
64x32x16	3	3.87E-2	3.80E-2	1	–
128x64x32	4	3.08E-1	7.25E-1	2	0.007
256x128x64	5	2.54	6.23	2	0.010

Table 5.18: Three dimensional Poisson's equation on the unit sphere solved using NUMG. CPU time in seconds.

Note that the anisotropy is in general so large that r-line Gauss–Seidel on its own (as we saw in Section 5.3) and conjugate gradients (CG) preconditioned with r-line Gauss–Seidel – which is essentially the same as the method currently employed by the MET Office [28] – work very well too at least for moderate grid sizes (cf. Table 5.19, columns 2 and 3). However, neither of these two other methods is robust to grid refinement and the number of iterations grows as the grid resolution is increased, such that for large (typical) problem sizes the non-uniform multigrid method outperforms both of them.

	NUMG with Galerkin product		r-line Gauss-Seidel		r-line preconditioned CG	
Problem size	# Iterations	Total time	# Iterations	Total time	# Iterations	Total time
32x16x8	5	5.16E-2	3	2.00E-3	4	6.22E-3
64x32x16	5	4.02E-1	6	3.11E-2	5	5.96E-2
128x64x32	5	3.65	26	1.08	13	9.68E-1
256x128x64	5	30.59	253	121.81	39	19.51

Table 5.19: Three dimensional Poisson's equation on the unit sphere solved using (column 1) NUMG with coarse grid operators created using the Galerkin product, (column 2) r-line Gauss-Seidel and (column 3) CG preconditioned with the r-line Gauss–Seidel smoother. CPU time in seconds.

5.6.1 The Galerkin Approach

In the non-uniform multigrid method above, all coarse grid operators A_ℓ, $\ell = 1, \ldots, F-1$ were computed via finite volume discretisations, as they were for the 2D problems. However, in this section we also experiment with using the Galerkin product

$A_{\ell-1} = R_\ell A_\ell P_\ell$. In 3D, when using linear interpolation and full weighting restriction, the coarse grid operators computed via the Galerkin product have an 81-point stencil. This is more than ten times the number of non-zeros per row than the 7-point stencil operator resulting from discretising on each grid and so the efficiency of the matrix/vector multiplications and the smoother will be severely affected.

More significantly, each node has *four* non-zero neighbours in each coordinate direction as opposed to the standard two, thus the line smoother used in the vertical direction no longer requires a tridiagonal solver but a *pentadiagonal* solver instead. The Thomas algorithm (a solver for tridiagonal systems) must therefore be replaced by a pentadiagonal solver.

Alternatively, replacing full weighting restriction with four-point averaging (as we did in section 5.5.4) greatly reduces the fill-in for the coarse grid operators to a 27-point stencil. In addition, there are now only two non-zero vertical neighbours at each node and so the Thomas algorithm can be used. Thus the settings we use for the experiments in this section are linear interpolation, four-point average restriction, the Galerkin product to construct coarse grid operators and the Thomas algorithm to solve tridiagonal systems in the r-line smoother. The results are listed in the first column of Table 5.19, which indicate that this method works well, with a constant number of iterations and a linear increase in CPU time. However, the number of iterations required is larger than what we observed in Table 5.18. Additionally, as in the two dimensional case, the cost of setting up the coarse grid operators is far larger than when using finite volume discretisations, and the application of the matrix/ vector multiplications and the smoother also takes longer on the coarse grids. In fact, even the CG method preconditioned with r-line Gauss–Seidel outperforms multigrid with the Galerkin approach for this problem, hence the Galerkin approach is clearly not the fastest solver for this problem and will no longer be considered.

5.6.2 Reducing the Anisotropy in the Radial Direction

Now consider (5.6.1) with $L_\lambda(\phi, r) = L_\phi(\lambda, r) = 1$, $c = 0$ and $L_r(\lambda, \phi) \ll 1$, ie. reducing the strength of coupling in the radial direction. We keep $L_\lambda(\phi, r)$ and $L_\phi(\lambda, r)$ fixed to 1 as we have already investigated the efficiency of the method when varying these values in section 5.5. These tests are significant because in addition to the Helmholtz problem, other elliptic problems on the sphere used in the Met Office are of the type where $L_r(\lambda, \phi) \ll L_\phi(\lambda, r), L_\lambda(\phi, r)$, e.g. the Quasi-Geostrophic Omega equation. We know from Section 5.3 that because of the theory from Section 5.4.2, NUMG will still be optimal (even though the number of iterations may be larger but still robust to grid refinement), but other traditional solvers described earlier do not cope as well.

CG preconditioned with r-line Gauss–Seidel performed strongly when $L_r(\lambda, \phi) = 1$ (cf. Table 5.19), but as Table 5.20 shows, its performance is substantially worse when we set $L_r = 10^{-4}$. This is because, looking at stencils (5.6.5) and (5.6.6), we see that the λ off-diagonal entries are larger than the r off-diagonal entries in many more regions of the domain when $L_r = 10^{-4}$, causing r-line smoothing to be far less effective.

Problem size	Solve time (s)	# Iterations
32x16x8	2.83E-2	57
64x32x16	0.70	132
128x64x32	20.03	370
256x128x64	397.24	879

Table 5.20: Preconditioned CG method (with r-line Jacobi preconditioner) used to solve (5.6.1) with $L_\lambda = L_\phi = 1$ and $L_r = 10^{-4}$.

As for NUMG, in order to ensure the method is optimal for $L_r \ll 1$ we resort to the technique from Section 5.3 of combining r-line relaxation and no vertical coarsening with conditional semi-coarsening on the $\phi - \lambda$ plane, which was shown to be a robust method for solving (5.3.1). Thus, despite the direction of strongest coupling changing throughout the domain as discussed above, the r-line smoother will take care of strong coupling in the r-direction and conditional semi-coarsening will take care of the strong coupling in the λ-direction. We test the method for $L_r = 1$, 10^{-2} and 10^{-4}, and it is crucial for NUMG to be able to deal with the varying strengths of vertical anisotropy described in this chapter so that it is able to optimally solve each type of elliptic problem in NWP that was described in Chapter 2. Table 5.21 confirms that the method is robust in each case and that it easily outperformes the CG method preconditioned with r-line Gauss–Seidel. Therefore this is the method that will be used to replace the preconditioned GCR method currently operational at the Met Office.

	$L_r = 1$		$L_r = 10^{-2}$		$L_r = 10^{-4}$	
Problem size	N_{its}	Total time	N_{its}	Total time	N_{its}	Total time
32x16x8	1	1.60E-2	3	2.95E-2	6	3.50E-2
64x32x16	2	1.82E-1	4	2.88E-1	7	4.09E-1
128x64x32	3	1.94	5	2.93	8	4.16
256x128x64	3	11.24	5	23.7	8	34.94

Table 5.21: NUMG with conditional semi-coarsening on the $\phi - \lambda$ plane, r-line relaxation and no vertical coarsening applied to (5.6.1), with $L_\lambda = L_\phi = 1$ and various values of L_r. CPU time in seconds.

Problem size	No. of iterations	Setup time, s	Solve time, s	Total time, s
32x16x8	5	4.16E-2	7.81E-2	1.20E-1
64x32x16	5	6.05E-1	1.10	1.71
128x64x32	6	6.69	12.1	18.8
256x128x64	7	57.5	106.9	164.4

Table 5.22: Three Dimensional Poisson-type Equation (5.6.1) with $L_\lambda = L_\phi = 1$ and $L_r = 10^{-4}$ solved using `BoomerAMG` as a preconditioner to CG.

5.6.3 Comparison with Algebraic Multigrid (AMG)

Unlike for the 2D model problems in this chapter, it was not possible to provide a simple non-uniform coarsening strategy that also considers the anisotropy in the radial direction too, because of the direction of strongest coupling not being consistent. Therefore, we resorted to using r-line relaxation instead. However, suppose we had resorted to the geometric multigrid method of SCLR on the $\phi - \lambda$ plane (which is a commonly used approach) instead of conditional semi-coarsening with a point smoother. This, in conjunction with an r-line smoother, would then result in a *plane* smoother with coarsening only in the one direction which the smoother wasn't applied. The plane smoother will be very costly, and moreover, the coarsening factor is only two which means there will be a significant amount of work done on the coarse grids too. Thus, our NUMG method will be comparatively cheaper than other such geometric approaches.

The only issue with using line relaxation is that it could potentially result in NUMG being less efficient than *algebraic* multigrid methods which will use more sophisticated coarsening strategies such that a simple point smoother can be used, even in 3D. However, we already know from the 2D tests that the setup cost for AMG methods is a particular bottleneck, and this only gets worse in 3D. Therefore NUMG can still be expected to outperform AMG methods.

As in Section 5.1.2, we use `BoomerAMG` implementation of AMG on problem (5.6.1) for the three cases $L_r(\lambda, \phi) = 1$, $L_r(\lambda, \phi) = 10^{-2}$ and $L_r(\lambda, \phi) = 10^{-4}$ ($L_\phi(\lambda, r) = L_\lambda(\phi, r) = 1$ for each case). Results are given in Table 5.22, and in fact they show that AMG methods are outperformed by NUMG more easily in 3D, with the total CPU time up to five times slower than NUMG.

5.6.4 The Quasi-Geostrophic Omega Equation

Let us now use the NUMG method described in Section 5.6 to solve the Quasi-Geostrophic Omega equation (2.2.16), which we recall is:

$$-\frac{N^2(R)}{\theta_{ref}(R)\rho_{ref}(R)}\nabla_h^2(\rho_{ref}w') - f_0^2\frac{\partial}{\partial R}\left(\frac{1}{\theta_{ref}(R)\rho_{ref}(R)}\frac{\partial}{\partial R}(\rho_{ref}w')\right) = -P_w$$
$$\text{on } \{(R,\Phi,\Lambda) : a \le R \le a+d,\ 0 \le \Phi \le \pi,\ 0 \le \Lambda \le 2\pi\},$$

where we recall from Section 2.2.4 that θ_{ref} and ρ_{ref} are the reference values of the potential temperature and density, respectively, N^2 is a stability parameter (approximated to be a function of R only), f_0^2 is the constant Coriolis parameter and P_w contains all the sources of quasi-geostrophic forcing. (R,Φ,Λ) are the usual spherical polar coordinates, and the equation is solved with Dirichlet boundary conditions $w' = 0$ at the lower and upper vertical boundaries (i.e. $R = a$ and $R = a+d$). It is solved for $\rho_{ref}w'$, where ρ_{ref} is already known. We nondimensionalize the equation as in Section 3.1 and write it in the general form (5.6.1), with coefficients

$$\alpha_1(r,\phi,\lambda) = \frac{a^2}{d^2}\frac{\sin(\pi\phi)f_0^2}{\hat{\theta}_{ref}(r)\hat{\rho}_{ref}(r)}, \tag{5.6.7}$$

$$\alpha_2(r,\phi,\lambda) = \frac{\hat{N}^2(r)\sin(\pi\phi)}{\pi^2\hat{\theta}_{ref}(r)\hat{\rho}_{ref}(r)}, \tag{5.6.8}$$

$$\alpha_3(r,\phi,\lambda) = \frac{\hat{N}^2(r)}{4\pi^2\sin(\pi\phi)\hat{\theta}_{ref}(r)\hat{\rho}_{ref}(r)}, \tag{5.6.9}$$

where $N^2(R) = \hat{N}^2(r)$, $\theta_{ref}(R) = \hat{\theta}_{ref}(r)$ and $\rho_{ref}(R) = \hat{\rho}_{ref}(r)$. Note that in (5.6.7), we have the term a^2 instead of $(a+dr)^2$ as in (5.6.2). This is an approximation made at the Met Office. The equation is discretised on the Arakawa C-grid and Charney Phillips grid, which we recall from Figures 2-2 and 2-3. A cell centred finite volume discretisation yields nodes at the ρ-levels on the Charney Phillips grid, but the w' points which we solve for are located at the θ-levels. Thus we make a small approximation in which we assume that the lower boundary is located at ρ-level 1 instead of θ-level 0, and the upper boundary at ρ-level Top-1 instead of θ-level Top. Then a finite volume discretisation will yield cell centres at approximately the θ-levels.

We use stencil (3.2.8) from Section 3.2.1 to deduce the cell centred finite volume discretisation of (2.2.16):

$$-X_{i-\frac{1}{2},j}\frac{h_\lambda h_\phi}{h_r}\begin{bmatrix} -Y_i\frac{1}{4\pi^2\sin(\pi\phi_j)}\frac{h_\phi h_r}{h_\lambda} & -Y_i\frac{\sin(\pi\phi_{j+\frac{1}{2}})}{\pi^2}\frac{h_\lambda h_r}{h_\phi} & \\ & -\sum & -Y_i\frac{1}{4\pi^2\sin(\pi\phi_j)}\frac{h_\phi h_r}{h_\lambda} \\ & -Y_i\frac{\sin(\pi\phi_{j-\frac{1}{2}})}{\pi^2}\frac{h_\lambda h_r}{h_\phi} & \end{bmatrix} - X_{i+\frac{1}{2},j}\frac{h_\lambda h_\phi}{h_r},$$

where

$$X_{i\pm\frac{1}{2},j} = f_0^2 \frac{\hat{\theta}_{ref}(r_i)\hat{\rho}_{ref}(r_i)}{\hat{\theta}_{ref}(r_{i\pm\frac{1}{2}})\hat{\rho}_{ref}(r_{i\pm\frac{1}{2}})} \frac{a^2}{d^2}\sin(\pi\phi_j), \quad Y_i = \hat{N}^2(r_i)\,.$$

Note that the problem has been scaled by $\hat{\theta}_{ref}(r_i)\hat{\rho}_{ref}(r_i)$ because $\frac{\hat{\theta}_{ref}(r_i)\hat{\rho}_{ref}(r_i)}{\hat{\theta}_{ref}(r_{i+\frac{1}{2}})\hat{\rho}_{ref}(r_{i+\frac{1}{2}})} \approx \frac{\hat{\theta}_{ref}(r_i)\hat{\rho}_{ref}(r_i)}{\hat{\theta}_{ref}(r_{i+\frac{1}{2}})\hat{\rho}_{ref}(r_{i+\frac{1}{2}})} = \mathcal{O}(1)$ for all i, hence the sizes of coefficients L_r, L_ϕ and L_λ are

$$\begin{aligned} L_r(\phi_j,\lambda_k) &\approx f_0^2 = \mathcal{O}(10^{-8})\,,\\ L_\lambda(r_i,\phi_j) &= \hat{N}^2(r_i) \approx \mathcal{O}(10^{-4})\,,\\ L_\phi(r_i,\lambda_k) &= \hat{N}^2(r_i) \approx \mathcal{O}(10^{-4})\,, \end{aligned}$$

Thus solving (2.2.16) is similar to solving (5.6.1) with $L_\lambda = L_\phi = 1$ and $L_r = 10^{-4}$, with each term scaled by approximately 10^{-4}. We solve this equation with one processor on the N72 analysis grid which is used for test problems at the Met Office. This grid is a smaller than the operational analysis grid and has a resolution of $144\times109\times70$. Figure 5-6 is a visualization of the vertical velocity profile at a roughly 8.8km above ground level, which corresponds to θ-level 20 on the Charney-Phillips grid. This snapshot was taken after one iteration of VAR and the data was produced using the NUMG solver of the quasi-geostrophic omega equation that was implemented into the VAR system.

5.6.5 The Helmholtz Problem

We conclude this chapter by solving the Helmholtz problem as given in (2.1.24). Note the presence of first and zeroth order terms which have thus far been neglected in this chapter. However, both these terms can be discretised using the finite volume method, as shown in Section 3.2.1. Writing the Helmholtz problem in a general tensor product form (3.1.1), we have

$$\begin{aligned} K(r,\phi,\lambda) &= \begin{bmatrix} \frac{(a+dr)^2}{d^2}\frac{\sin(\pi\phi)}{GA} & 0 & 0\\ 0 & \frac{\sin(\pi\phi)}{\pi^2} & 0\\ 0 & 0 & \frac{1}{4\pi^2\sin(\pi\phi)} \end{bmatrix},\\ \mathbf{a}(r,\phi,\lambda) &= \left[\frac{1}{\theta^n GA}\frac{\partial\theta^{(1)}}{\partial r}\frac{(a+dr)^2}{d}\sin(\pi\phi), 0, 0\right]^T,\\ c(r,\phi,\lambda) &= \frac{\left(\frac{p^n}{\kappa C_p \Pi^n} - \kappa\theta^n\rho^n\right)}{\kappa\Pi^n\theta^n\Delta t^2\alpha^2 C_p\rho^n\theta^{(1)}A}(a+dr)^2\sin(\pi\phi)\,. \end{aligned}$$

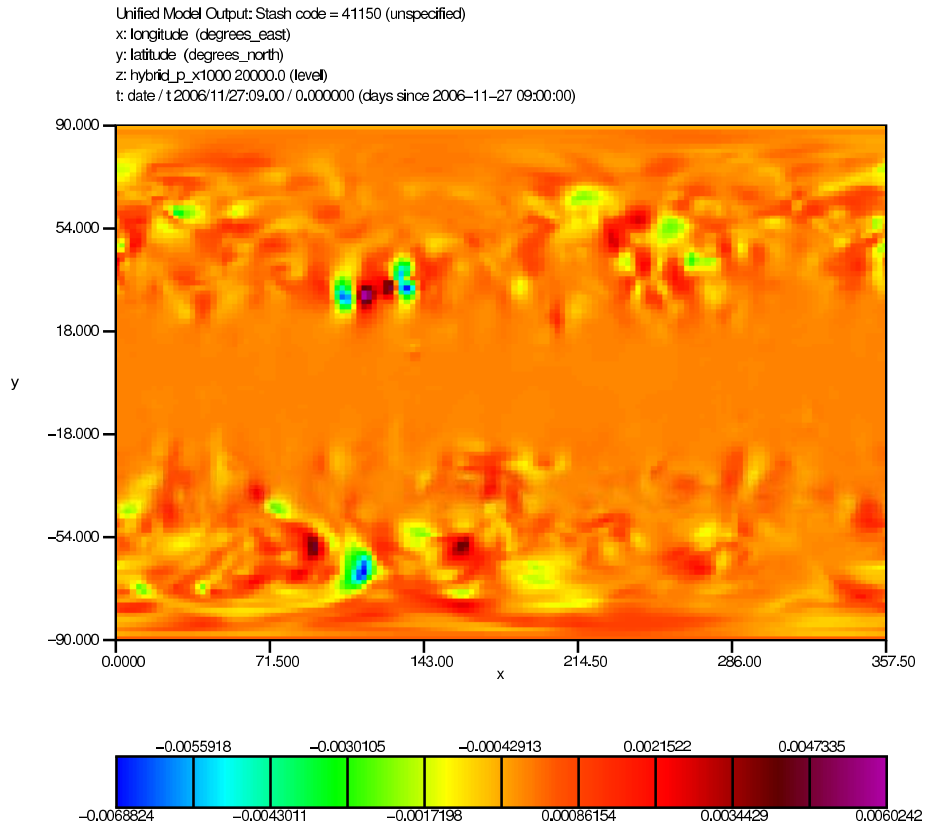

Figure 5-6: Snapshot of the vertical velocity, w, after one iteration of VAR at θ-level 20 (roughly 8.8km above ground level)

Recall from the derivation of the Helmholtz equation in Section 2.1 that $G = 1 - \alpha^2 \Delta t^2 C_p \frac{\partial \theta^{(1)}}{\partial z} \frac{\partial \Pi^n}{\partial z} > 0$ and $A = 1/(1 + \alpha^2 \Delta t^2 f^2) > 0$. Although these values are not known exactly, it is a good approximation that $A = \mathcal{O}(1)$ and $G = \mathcal{O}(1)$, though we have shown that NUMG can handle different magnitudes of coefficients (see Table 5.21). We solve the problem using the same range of problem sizes as done throughout this chapter for 3D problems, and the results are highlighted in Table 5.23.

The results indicate that the method is almost robust. We have already discussed that the presence of a positive zeroth order term (which we have here) improves the conditioning of a system. However, the presence of a first order term is usually detrimental to the conditioning of a system, because we recall from Section 3.2.1 that its discretisation yields a nonsymmetric operator. The lack of symmetry means that the CG method cannot be used as a coarse grid solver, but this is not an issue since we have been using an r-line smoother on the coarse grid for all the 3D experiments. Now, since we have a first order term only with respect to the radius, the r-line smoother

Problem size	# Coarse grids	Setup time (s)	Solve time (s)	# Iterations	μ_{avg}
32x16x8	2	1.04E-2	2.32E-2	5	0.039
64x32x16	3	7.94E-2	3.51E-1	7	0.074
128x64x32	4	6.33E-1	3.58	8	0.122
256x128x64	5	5.02	30.4	8	0.128

Table 5.23: Three dimensional Helmholtz problem (2.1.24), as used in NWP, solved on the unit sphere using NUMG. CPU time in seconds.

will in fact take care of this term on its own, thus creating no additional difficulties for the solver [3]. For this reason, we still achieve optimal convergence results despite the presence of the first-order term.

[3]Note that multigrid in the presence of strong advection terms is a major problem in scientific computing but is not covered in this thesis.

Chapter 6

Parallelization

In this chapter we describe the parallelization of algorithms for the solution of sparse linear systems, focusing primarily on the particular techniques used for the parallelization of the non-uniform multigrid (NUMG) method developed in Chapter 5. Section 6.1 gives a general introduction to parallelism, describing the main tools used and how the parallel performance of an algorithm is measured relative to its sequential performance (i.e. with one processor). In Section 6.2, we describe more specifically how each component of a multigrid algorithm is parallelized, addressing the main issues that affect the parallelism, such as how the grid is partitioned into subdomains for each processor and the necessary communication required between the processors. We then look at how to modify the parallelization for the NUMG method, where non-uniform coarse grids and the boundary conditions will have a particularly significant effect. Finally in Section 6.3, the performance of the parallel NUMG code is tested, with comparisons to parallel versions of existing solvers.

6.1 Introduction to Parallelism

In parallel computing, several calculations are carried out simultaneously, operating on the principle that large problems are split into smaller ones which are solved in parallel using multiple processors. The key to parallel computing is that the problems on each processor are run independently, that is, one processor may not modify a variable that another processor might be using or modifying. The rapidly developing parallel computer hardware and its characteristics must be understood and taken into account in program and algorithm design. In [32], parallel computers are distinguished into two main types:

Definition 6.1.1 (Shared memory)**.** For parallel computers with shared memory, the

main memory is shared between all processors, each of which can access all the memory at the same time.

The advantage of shared memory is that each processor has access to all the data, so porting the sequential code (i.e. code designed to run on one processor only) to parallel computers can be done with little difficulty. However, as the number of processors increases, good scalability, i.e. performance relative to the number of processors (see Section 6.1.2), cannot be guaranteed unless there is a lot of memory. Also, since all the processors have access to the same data, there needs to be *synchronization* between each processor to ensure exclusive access to any variable for one process at a time to prevent multiple processors modifying the same variable.

Definition 6.1.2 (Distributed memory)**.** For parallel computers with distributed memory, the main memory is distributed between each processor.

The advantages of distributed memory are that there are no access conflicts between processors since data is locally stored, and that well written code can be optimally scalable (see Section 6.1.2) for a large number of processors. However, there is no direct access to the data on other processors so if one processor requires data stored in the memory of another, *communication* between processors is necessary. This involves a significant rewrite of the sequential code.

In this chapter we focus on computers with distributed memory, since this is used for the multiprocessor computers at the University of Bath and at the Met Office.

6.1.1 Message Passing

For processors that do not share their memory, message passing platforms must be used when communication is required between processors. Message passing is a mechanism for directly transferring data from one processor to another, and the most common library in this field is the message passing interface (MPI) [42, 60]. In this section we focus on the global operations used in message passing for the MPI library.

Send and Receive

All parallel programs need routines to send and receive data from one processor to another. In the MPI library, these are implemented as

$$\texttt{MPI\_SEND}(\texttt{data}, \texttt{n}, \texttt{type}, \texttt{dest}, \texttt{tag}, \texttt{comm}, \texttt{ierr})$$

for sending `n` elements of data, where `data` is the index of the first element to be sent. `type` is the datatype of the elements (e.g. integer, real), and the message is sent from

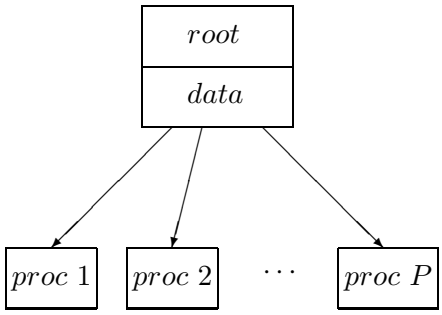

Figure 6-1: The `MPI_BCAST` operation

the calling processor to processor `dest`. The corresponding receiving routine is

$$\texttt{MPI\_RECV}(\texttt{data}, \texttt{n}, \texttt{type}, \texttt{srce}, \texttt{tag}, \texttt{comm}, \texttt{status}, \texttt{ierr})$$

which receives the message from processor `srce`. Thus the arguments `data, n, type` define the message, and the arguments `dest` (distribution processor) and `srce` (source processor) define the address. The remaining arguments are of no interest for the purposes of this chapter.

Broadcast

Often, information from one processor must be sent out to all processors. We denote the processor sending out the information as the root processor, and the MPI routine that achieves this is

$$\texttt{MPI\_BCAST}(\texttt{data}, \texttt{n}, \texttt{type}, \texttt{root}, \texttt{comm}, \texttt{ierr})\,,$$

where the root processor sends `n` elements of data with `data` being the index of the first element (see Figure 6-1).

All-Reduce

Here, values from each processor are combined and distributed back to all processors. This is useful for computing arithmetical/ logical operations using data stored across all the processors. The call

$$\texttt{MPI\_ALLREDUCE}(\texttt{senddata}, \texttt{recvdata}, \texttt{n}, \texttt{type}, \texttt{op}, \texttt{comm}, \texttt{ierr})$$

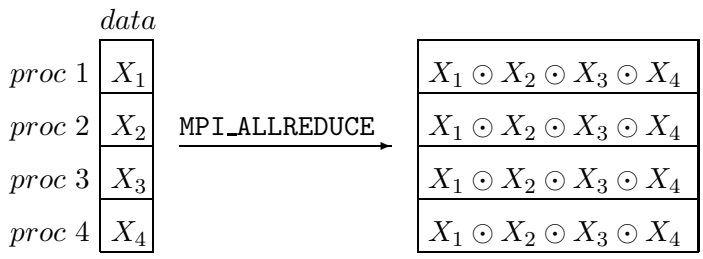

Figure 6-2: The `MPI_ALLREDUCE` operation, with an arithmetic/ logical operation $\odot$

is the appropriate function in the MPI library. Elements of length `n` are sent from each processor, with `senddata` being the index of the first element. `op` is the operation (e.g addition, multiplication) and `recvdata` is the index of the first element of the received data that has been distributed back to all processors. Figure 6-2 illustrates the All-reduce process. A slight variant, `MPI_REDUCE` is used if the received data is only required on one processor.

6.1.2 Performance of Parallel Algorithms

The suitability of parallel algorithms for a large number of processors can be measured in the following ways:

Definition 6.1.3 (Scalability/ weak scalability)**.** This is the property of a program that indicates its ability to handle growing amounts of work with a proportionally increasing number of processors. Let N_{loc} be the local problem size per processor. If the CPU time of running a problem of size N_{loc} with one processor is equal to that of solving a problem of size $P * N_{loc}$ with P processors, then the program is *optimally scalable* for P processors.

Definition 6.1.4 (Speedup/ strong scalability)**.** This is a measure of the gain in computing time by parallelism. The speedup, S_P, measures the performance gain of a parallel code running on P processors in comparison with the sequential version, with the global problem size N_{glob} fixed. The speedup is measured by

$$S_P = \frac{t_1}{t_P},$$

where t_1 and t_P are the CPU times of running the program on 1 and P processors respectively. The ideal speedup on P processors is $S_P = P$.

Measuring weak and strong scalability is important to determine the gains of using more processors. It is generally agreed that strong scalability is harder to achieve due

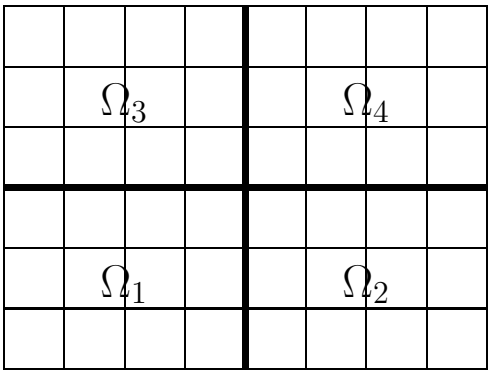

Figure 6-3: Partitioning Ω into four subdomains

to the memory requirements decreasing proportionally to the number of processors used. The amount of computation per processor decreases when more processors are used, and eventually this will lead to the inter-processor communication time becoming too significant a fraction of the total computation cost. It is therefore more desirable to measure parallel performance by weak scalability because a fixed local problem size ensures that the memory access and inter-processor communication costs per processor remain relatively constant.

6.2 Multigrid in Parallel

6.2.1 Grid Partitioning

The first consideration in the parallelization of a multigrid method is the partitioning of the grid. The physical domain Ω must be decomposed into several subdomains $\Omega_i, i = 1 \cdots, P$ corresponding to the number of processors, e.g. as in Figure 6-3 with $P = 4$. The boundaries of the subdomain must coincide with the boundaries of control cells, and this condition must similarly hold on all the coarser grids. Also, for a hierarchy of grids, the partitions on the coarse grids should be located on the same lines as those on the fine grid to avoid unnecessary additional communication when using the grid transfer operators. Note that this can always be done if the partitioning is chosen on the coarsest grid first, since each coarse grid is a subset of finer grids. It is preferable to choose a partitioning such that there is an even *load balance* between the processors, i.e. each subdomain will contain approximately the same number of grid points so that the work done by each processor is roughly equal. An even load balance can easily be achieved for uniform grids and when full or semi coarsening is employed.

6.2.2 Communication

All the components in the multigrid algorithm 4.5 (e.g. interpolation, restriction, smoothing and the coarse grid solve) can be fully described in terms of *vector dot products* and *matrix-vector multiplications*, in addition to some scalar operations. For many of these operations, data that is distributed across multiple processors is required, thus the need for communication is apparent. However, it is obvious that the whole multigrid method can be parallelized if we can parallelise each of the operations in it.

By a suitable reordering of rows and columns, a global matrix A (i.e. defined on the whole domain) and a global vector $\mathbf{x}$ can be partitioned into local components defined on the subdomains as follows:

$$A = \begin{pmatrix} A_{11} & A_{12}^{off} & \cdots & A_{1P}^{off} \\ A_{21}^{off} & A_{22} & \cdots & A_{2P}^{off} \\ & & \ddots & \\ A_{P1}^{off} & A_{P2}^{off} & \cdots & A_{PP} \end{pmatrix}, \qquad \text{and} \qquad \mathbf{x} = \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_P \end{pmatrix},$$

Let n denote the global problem size and n_i the number of local grid points in Ω_i. Then we have

$$A_{ii} \in \mathbb{R}^{n_i,n_i}, \qquad A_{ij}^{off} \in \mathbb{R}^{n_i,n_j} \qquad \text{and} \qquad \mathbf{x}_i \in \mathbb{R}^{n_i} \qquad \text{for} \qquad i,j = 1,\ldots,P\,.$$

Let us also define the following components of matrix A:

$$A_i = \begin{bmatrix} A_{i1}^{off} & A_{i2}^{off} & \cdots & A_{ii} & \cdots & A_{in}^{off} \end{bmatrix}.$$

It is common to store A_i in processor i, for $i = 1,\ldots,P$, and likewise $\mathbf{x}_i$ in processor i. This method of partitioning the matrices is in fact what is used for the parallel computations in this thesis.

Now, for some basic vector operations such as

$$\begin{aligned} &\text{Copying:} \quad \mathbf{y} = \mathbf{x} \quad \text{i.e. } \mathbf{y}_i = \mathbf{x}_i, \quad \forall i = 1,\ldots,P\,, \\ &\text{Scaling:} \quad \mathbf{x} = a\mathbf{x} \quad \text{i.e. } \mathbf{x}_i = a\mathbf{x}_i, \quad \forall i = 1,\ldots,P\,, \end{aligned}$$

there is no need for communication because the operations on each processor are independent. However, for parallel vector dot products or matrix vector multiplications, the operations are not independent.

Vector Dot Product

The dot product $\alpha = (\mathbf{a}, \mathbf{b})$ can be computed in parallel using `MPI_ALLREDUCE`. The local dot products

$$\alpha_i = (\mathbf{a}_i, \mathbf{b}_i)\,, \qquad \forall i = 1, \ldots, P$$

are computed independently for each subdomain Ω_i, and the `MPI_ALLREDUCE` call calculates

$$\alpha = \sum_{i=1}^{P} \alpha_i,$$

with the result available on all processors.

Matrix-Vector Multiplication

For a matrix-vector multiplication, the product $\mathbf{y} = A\mathbf{x}$ is written in terms of the local components as

$$\begin{pmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \\ \vdots \\ \mathbf{y}_P \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12}^{off} & \cdots & A_{1P}^{off} \\ A_{21}^{off} & A_{22} & \cdots & A_{2P}^{off} \\ & & \ddots & \\ A_{P1}^{off} & A_{P2}^{off} & \cdots & A_{PP} \end{pmatrix} \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_P \end{pmatrix}. \tag{6.2.1}$$

Here, the vectors $\mathbf{x}$, $\mathbf{y}$ and the matrix A are said to be *distributed* between processors (cf. [32]). Suppose the matrix is a finite volume discretisation of the Poisson equation on Ω, i.e. a 5-point stencil. Then the entries of A_{ij}^{off} correspond to a connection between a node in Ω_i and a node in Ω_j and will be zero almost everywhere. On processor i, the local matrix-vector multiplication will be

$$\mathbf{y}_i = A_i\mathbf{x} = A_{ii}\mathbf{x}_i + \sum_{i \neq j} A_{ij}^{off}\mathbf{x}_j\,. \tag{6.2.2}$$

Recall that we store A_{ij}^{off} locally on processor i. However, the need for communication arises because the entries of $\mathbf{x}_j$ are stored in processor j, thus the necessary components of $\mathbf{x}_j$ must be sent from processor j to processor i. These are the indices of the vector that correspond to the nodes in Ω_j that are adjacent to the boundary of Ω_i, as shown in Figure 6-4. This is because A_{ij}^{off} is in fact zero almost everywhere, with non-zero entries only in components that relate to those particular nodes.

Let $\boldsymbol{\beta}_{ji}$ denote the set of nodes in Ω_j that are adjacent to the boundary of Ω_i. Then the components of $\mathbf{x}_j$ that are sent to processor i are stored in a vector $\mathbf{x}_{i\leftarrow j} \in \mathbb{R}^{n_j}$

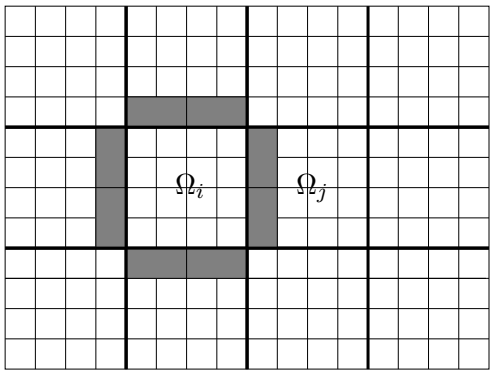

Figure 6-4: The communication needed for matrix-vector multiplications

belonging to processor i, as follows:

$$x_{i\leftarrow j}(k) = \begin{cases} x_j(k) & \text{if } k \in \boldsymbol{\beta}_{ji} \\ 0 & \text{otherwise} \end{cases} \tag{6.2.3}$$

for $k = 1, \ldots, n_j$. This procedure is known as the *accumulation* of $\mathbf{x}_i$, where additional components of the global vector $\mathbf{x}$ are accumulated from other processors whose subdomains share a boundary with Ω_i. Using the vectors $\mathbf{x}_{i\leftarrow j}$ from (6.2.3), the matrix-vector multiplication can be calculated on processor i as

$$\mathbf{y}_i = A_{ii}\mathbf{x}_i + \sum_{i\neq j} A_{ij}^{off}\mathbf{x}_{i\leftarrow j}\,, \qquad \forall i = 1, \ldots, P\,.$$

With matrix A being a 5-point stencil in 2D, for example, this means processor i will only ever require data from the four processors adjacent to it. This communication structure therefore resembles a 5-point stencil and will be known as a *5-point stencil communication topology*. Hence, the conversion of a distributed vector to an accumulated vector involves a 5-point stencil communication topology.

This communication strategy is also commonly referred to as a *ghost point* or *halo* strategy [32] since each processor requires data from nodes that are only directly outside its subdomain, as shown in Figure 6-4.

6.2.3 Parallel Components of Multigrid

We now discuss how each of the components in the multigrid algorithm can be implemented in parallel:

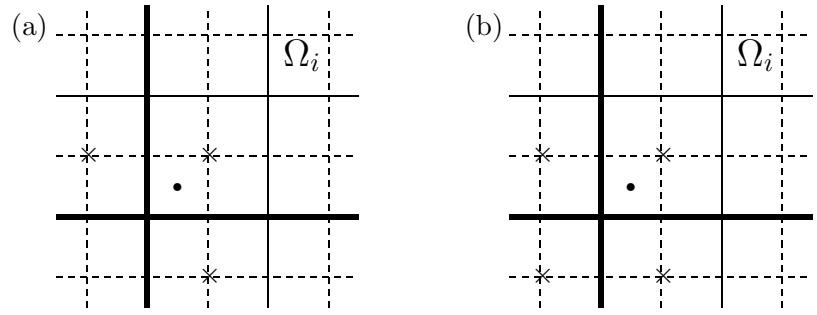

Figure 6-5: (a) Linear interpolation: some entries of the matrix depend on data from processors associated with adjacent subdomains only, and (b) bilinear interpolation: some entries of the matrix depend on data also from processors associated with subdomains that share a corner with Ω_i. Dotted lines denote the fine grid, the thin lines denote the coarse grid and the thick lines denote the subdomain boundaries

Interpolation and Restriction

Assuming all the grids are partitioned along the same lines, the interpolation and restriction matrices are partitioned like the matrix A from (6.2.1), i.e.

$$P_\ell = \begin{pmatrix} (P_\ell)_{11} & (P_\ell)_{12}^{off} & \cdots & (P_\ell)_{1P}^{off} \\ (P_\ell)_{21}^{off} & (P_\ell)_{22} & \cdots & (P_\ell)_{2P}^{off} \\ & & \ddots & \\ (P_\ell)_{P1}^{off} & (P_\ell)_{P2}^{off} & \cdots & (P_\ell)_{PP} \end{pmatrix}, \quad R_\ell = \begin{pmatrix} (R_\ell)_{11} & (R_\ell)_{12}^{off} & \cdots & (R_\ell)_{1P}^{off} \\ (R_\ell)_{21}^{off} & (R_\ell)_{22} & \cdots & (R_\ell)_{2P}^{off} \\ & & \ddots & \\ (R_\ell)_{P1}^{off} & (R_\ell)_{P2}^{off} & \cdots & (R_\ell)_{PP} \end{pmatrix}.$$

The matrix-vector multiplication $\mathbf{e}_\ell = P_\ell \mathbf{e}_{\ell-1}$ is calculated in parallel by accumulating the distributed components of $\mathbf{e}_{\ell-1}$ on each processor, and the components of $\mathbf{e}_\ell$ on processor i are found via the calculation (6.2.2), i.e.

$$(\mathbf{e}_\ell)_i = (P_\ell)_{ii}(\mathbf{e}_{\ell-1})_i + \sum_{i \neq j} (P_\ell)_{ij}^{off} (\mathbf{e}_{\ell-1})_j \,.$$

If linear interpolation is used (cf. Section 4.4.2), we have a 5-point communication topology because the entries of $(P_\ell)_{ij}^{off}$ will only be non-zero at indices corresponding to coarse grid nodes that are adjacent to the boundary of Ω_i, as illustrated in Figure 6-5(a). However, for bilinear interpolation or any higher order interpolations, then the entries of $(P_\ell)_{ij}^{off}$ can also depend on data from processors with subdomains that only share a corner with Ω_i, as illustrated in Figure 6-5(b), thus resulting in additional ghost points and a 9-point communication topology. In terms of the performance of the parallel algorithm it is not desirable to have to communicate with the four additional processors, and so linear interpolation is sufficient.

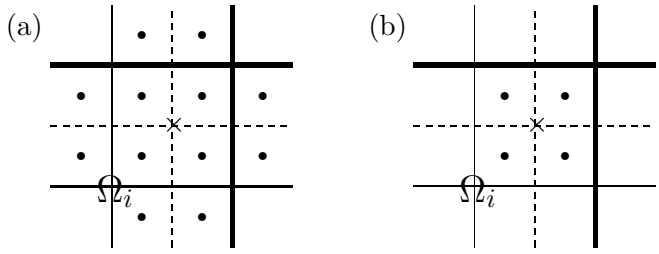

Figure 6-6: (a) Full weighting restriction: some entries of the matrix depend on data from processors associated with adjacent subdomains, and (b) Four point average: all the entries of the matrix depend on the data from the local processor. Dotted lines denote the fine grid, the thin lines denote the coarse grid and the thick lines denote the subdomain boundaries

The matrix-vector multiplication $\mathbf{r}_{\ell-1} = R_\ell \mathbf{r}_\ell$ is calculated in the same way:

$$(\mathbf{r}_{\ell-1})_i = (R_\ell)_{ii}(\mathbf{r}_\ell)_i + \sum_{i \neq j} (R_\ell)_{ij}^{off} (\mathbf{r}_\ell)_j \,.$$

The communication topology is a 5-point stencil if R_ℓ is the transpose of a linear interpolation operator because the entries of $(R_\ell)_{ij}^{off}$ will only be non-zero at entries corresponding to fine grid nodes that are adjacent to the boundary of Ω_i, as shown in Figure 6-6(a). If R_ℓ is the simpler four point average (cf. Section 4.4.2), however, no inter-processor communication is necessary because no data is needed from components that correspond to the fine grid nodes in a different subdomain (see Figure 6-6(b)). Clearly, the advantage of the four point average is that no communication is necessary, but since it is not the transpose of linear interpolation, the symmetry of the method will be lost.

Smoothing

Recall the Gauss–Seidel method from Section 4.2:

$$\mathbf{u}^{(k)} = \mathbf{u}^{(k-1)} + D^{-1}(\mathbf{b} - L\mathbf{u}^{(k)} - U\mathbf{u}^{(k-1)} - D\mathbf{u}^{(k-1)}) \,.$$

The updated solution at each node depends on the solution at previously updated nodes. If the previously updated nodes are on a different processor, then the parallelization of this method would mean processors have to wait for each other, thus increasing the amount of time some processors may remain idle.

However, for the Jacobi method

$$\mathbf{u}^{(k)} = \mathbf{u}^{(k-1)} + D^{-1}(\mathbf{b} - A\mathbf{u}^{(k-1)}),$$

the updated solution only depends on the solution at the previous iteration, and so the parallelization of the method does not require the processors to have to wait for each other. All the communication during the calculation of $\mathbf{u}^{(k)}$ will be to accumulate $\mathbf{u}^{(k-1)}$ at the start of each iteration.

By splitting the local matrix A_{ii} as $A_{ii} = D_{ii} + L_{ii} + U_{ii}$, the Jacobi method is parallelized as

$$\mathbf{u}_i^{(k)} = \mathbf{u}_i^{(k-1)} + D_{ii}^{-1}(\mathbf{b}_i - A_{ii}\mathbf{u}_i^{(k-1)} - \sum_{j\neq i} A_{ij}^{off}\mathbf{u}_{j\leftarrow i}^{(k-1)}), \qquad i = 1,\ldots,P,$$

where $\mathbf{u}^{(k-1)}$ is accumulated on each processor to obtain $\mathbf{u}_{j\leftarrow i}^{(k-1)}$, as defined in (6.2.3), and the same partitioning of A as (6.2.1) is used.

Now, as stated in 4.2, the Gauss–Seidel method is a better smoother than the Jacobi method, typically reducing the number of multigrid iterations required by approximately 25% (eg. 8 instead of 12). However, in parallel, the Gauss–Seidel requires more communication. Therefore it makes sense to modify the smoother to a hybrid Jacobi/Gauss−Seidel smoother, only making use of the most up-to-date values of the solution vector if the corresponding node is associated with the same processor (within one relaxation step). If the node is a ghost point and belongs to another processor then the value from the previous relaxation sweep is used (as in the Jacobi method). This may lead to a slight increase in the number of iterations on large numbers of processors or small local problem sizes, but it avoids unnecessary communication and data dependencies. The method is then parallelized, on processors $i = 1,\ldots,P$ as

$$\mathbf{u}_i^{(k)} = \mathbf{u}_i^{(k-1)} + D_{ii}^{-1}\left(\mathbf{b}_i - L_{ii}\widetilde{\mathbf{u}}_i^{(k)} - U_{ii}\mathbf{u}_i^{(k-1)} - D_{ii}\mathbf{u}_i^{(k-1)} - \sum_{j\neq i} A_{ij}^{off}\mathbf{u}_{i\leftarrow j}^{(k-1)}\right).$$

where

$$\widetilde{u}_i^{(k)}(\ell) = \begin{cases} u_i^{(k)}(\ell) & \text{if node } \mathbf{n}_\ell \in \Omega_i \\ u_{i\leftarrow j}^{(k-1)}(\ell) & \text{if node } \mathbf{n}_\ell \in \Omega_j, \quad j \neq i \end{cases}.$$

The algorithm for the parallel hybrid Jacobi/Gauss–Seidel smoother is given in Algorithm 6.1.

Note that for block smoothers, entire blocks are updated at each iteration rather than individual nodes, which requires tridiagonal solves typically using the Thomas

Algorithm 6.1 The hybrid Jacobi/Gauss–Seidel Method: `jac_gs(`$A_i, \mathtt{u}_i, \mathtt{b}_i$`)`

Choose $\mathbf{u}^{(0)}$
for $k = 1, 2, \ldots,$ until convergence $\ldots$ **do**
  Accumulate $\mathbf{u}_i^{(k-1)}$
  $\mathbf{u}_i^{(k)} = \mathbf{u}_i^{(k-1)} + D_{ii}^{-1}\left(\mathbf{b}_i - L_{ii}\widetilde{\mathbf{u}}_i^{(k)} - R_{ii}\mathbf{u}_i^{(k-1)} - D_{ii}\mathbf{u}_i^{(k-1)} - \sum_{j\neq i} A_{ij}^{off}\mathbf{u}_{i\leftarrow j}^{(k-1)}\right)$
end for

algorithm. This procedure can be parallelized, but the data dependencies are such that the processors will have to wait for each other, resulting in some processors being idle for long periods of time. Also, since the tridiagonal blocks are small compared to the main problem, particularly in 3D, it is preferable for each block to be contained on only one processor. This usually leads to a 2D partitioning strategy with no partitioning in the direction which the tridiagonal blocks are updated.

Coarse Grid Solve

The coarse grid solves discussed in thesis are either

- a Krylov subspace solver, or
- a line smoother.

Krylov subspace methods use a combination of matrix-vector multiplications (with the 5-point stencil operator A), vector dot products and scalar operations. These components, as discussed already, can all be parallelized. Line-smoothers can also be parallelized efficiently, provided that a hybrid Jacobi/ Gauss–Seidel smoother is used and each tridiagonal block is not split between multiple processors.

6.2.4 Parallelization Strategy of NUMG

We will see in this section that the particular strategies used for the NUMG method lead to necessary alterations of the sequential method in order to obtain an efficient parallelization. Let us first consider the radial direction in the 3D NUMG method. We have discussed already that the use of the r-line smoother means it would be detrimental to the efficiency of the method to partition the domain in the radial direction. The tridiagonal solves along each r-line would lead to too much unnecessary communication and data dependencies between the processors, therefore we partition only in the longitudinal and latitudinal directions. However, any number of partitions in these

directions should be admissible. Importantly, this partitioning strategy also conformes with the codes used at the Met Office.

Now for the longitudinal and latitudinal directions, the main issues that arise are the partitioning of the coarse grids and the communication at the boundaries.

The grid partitioning strategy is simple: choose a partitioning such that all grids are partitioned along the same lines (for the 2D method) or planes (for the 3D method) to avoid extra communication when using the grid transfer operators R_ℓ and P_ℓ. We have mentioned one way of achieving this, which is to partition the coarsest grid first, and then ensure that all finer grids are partitioned along the same lines/planes. However, this partitioning strategy leads to two major problems:

Load balance: Partitioning on the coarsest grid may have an effect on the load balance on the finer grids. On the finest grid, where the majority of the work is done, this becomes an issue as an uneven load balance will cause the processors to have an uneven distribution of work.

Grid partitioning at the Met Office: The multigrid solver is to be part of a much larger code (belonging to the Met Office) and the grid partitioning on the finest grid needs to conform with the partitioning used in the rest of the code. Since one of the ultimate aims of this project is to implement the multigrid method into the operational code used at the Met Office, it is vital that the grid partitions are identical in both codes, and this cannot be achieved if the partitioning is determined on the coarsest grid first. Typically, the Met Office chooses a partitioning such that there is an evenly distributed load balance on each processor.

With the above considerations in mind, we must choose the grid partitioning on the finest grid first. The immediate concern with doing this is that a line on the fine grid chosen as a partition may not exist on the coarse grid. Thus a small but necessary modification to the conditional semi-coarsening strategy introduced in Section 5.2 is required. We enforce an extra condition that coarsening cannot occur across a processor boundary, even if it is in a region where full coarsening is enforced. This ensures that the boundaries of each subdomain coincide with the boundaries of control cells and that the partitioning on each coarse grid is along the same lines/planes as that on the fine grid. This adjustment to the coarsening strategy results in some additional non-uniformity in the coarsening near processor boundaries, as shown in Figure 6-7 for two parallel coarse grids. Note that the mesh width in the θ-direction on grid $\mathcal{T}_\ell$, i.e. $h_\theta^{(\ell)}$, is no longer guaranteed to be constant thus it is generalized as $h_{\theta,k}^{(\ell)}$ for $k = 1, \ldots, n_\theta^{(\ell)}$,

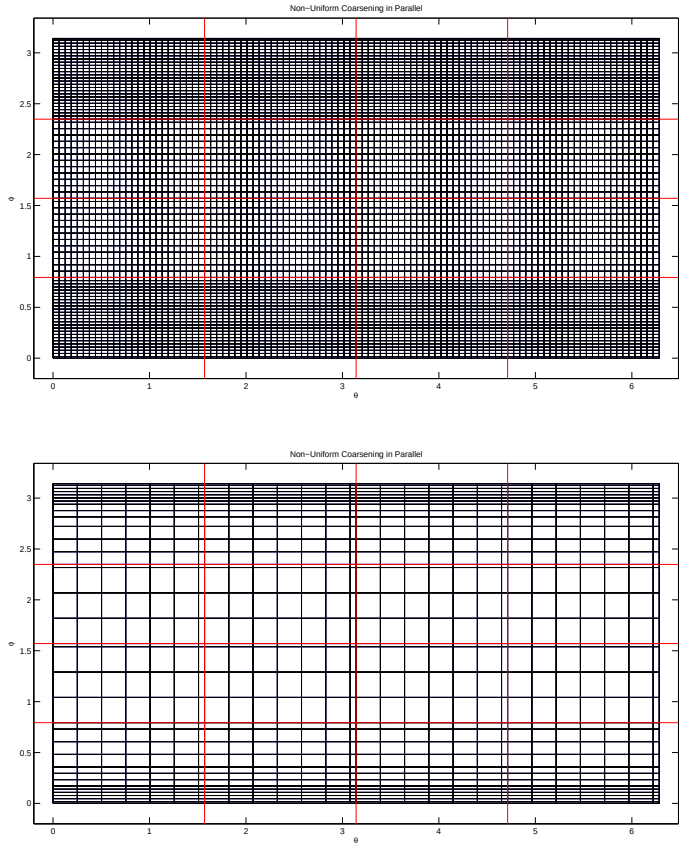

Figure 6-7: Parallel non-uniform coarsening keeping the location of the processor interface fixed. Grids for one refinement (top) and three refinements (bottom) with a uniform 4×4 partitioning onto 16 processors.

where $n_\theta^{(\ell)}$ is the number of grid points in the θ-direction on grid $\mathcal{T}_\ell$. Figure 6-7 also shows that the additional non-uniformity can actually become quite severe, which is detrimental particularly in the θ-direction where uniform full coarsening was imposed for the sequential code. Hence further adjustments are made to ensure that no cell is too fine or coarse relative to its adjacent cells. We do this by making sure that the mesh widths are at least quasi-uniform near the processor boundaries, i.e. $h_{\theta,k}^{(\ell)} \leq C_\theta h_{\theta,k+1}^{(\ell)}$ and $h_{\theta,k+1}^{(\ell)} \leq C_\theta h_{\theta,k}^{(\ell)}$ for some constant C_θ. The effect of this is that the same side of the domain is never semi-coarsened twice in succession, thus avoiding the possibility of thin cells at the boundaries. In our experiments, C_θ is set to 0.5.

Our partitioning strategy deals with both the above issues, despite introducing some additional non-uniformity on the coarse grids. However, we will see in Section 6.3 that the non-uniformity does not have a significant effect on the performance of the parallel method.

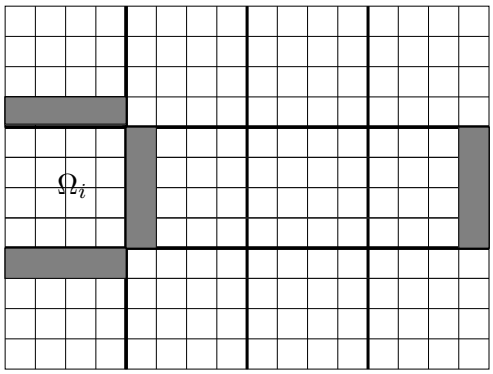

Figure 6-8: Communication for processor subdomains on the periodic boundary

As discussed already, a ghost point (or halo) strategy is used for communication between processors, which is implemented using MPI. However, the communication at the boundaries of the domain must be handled differently to account for the poles and the periodic boundary conditions. For a subdomain Ω_i which has a boundary coinciding with, say, the west boundary of Ω like in Figure 6-8, there is no adjacent subdomain on the left. However, because of the periodic boundary conditions, communication must occur with the east-most subdomain as demonstrated in Figure 6-8.

As for the treatment of the poles, each of the southern-most subdomains will require data from the south pole node, and likewise for the north pole. However, the poles are only stored on one processor each, so they need to be communicated to all the relevant processors. Also, the pole nodes themselves will need information from their adjacent nodes which may be distributed across several processors, so these all need to be communicated to the processors containing the poles.

In addition to the coarse grid partitioning and the treatment of the communication at the boundaries, the discretisation scheme is chosen such that it yields a 5-point stencil on each grid. This means discretising the problem on each grid rather than using the Galerkin product (cf. Section 4.4.2). Operators with a 5-point stencil mean that any matrix-vector multiplications or applications of the smoother using these operators require communication only with processors associated with adjacent subdomains, i.e. a 5-point communication topology. We also choose R_ℓ and P_ℓ to conform with the 5-point communication topology, i.e. linear interpolation and full weighting restriction.

6.2.5 Parallel Multigrid Algorithm

Using the parallel components of multigrid described in this chapter, Algorithm 6.2 describes the parallel V-cycle algorithm. Note that the accumulation of vectors takes place several times in the algorithm.

Algorithm 6.2 The parallel V-cycle on processor i: `PVcycle`$((A_\ell)_i, (\mathbf{u}_\ell)_i, (\mathbf{b}_\ell)_i, \ell)$

if $(\ell = 1)$ **then**
  Accumulate $(\mathbf{u}_\ell)_i$
  Solve $(A_\ell)_i(\mathbf{u}_\ell)_i = (\mathbf{b}_\ell)_i$ (parallel coarse grid solve)
else
  Accumulate $(\mathbf{u}_\ell)_i$
  $(\mathbf{u}_\ell)_i =$ `jac_gs`$((A_\ell)_i, (\mathbf{u}_\ell)_i, (\mathbf{b}_\ell)_i)$ (parallel pre-smoothing)
  $(\mathbf{r}_\ell)_i = (\mathbf{b}_\ell)_i - ((A_\ell)_{ii}(\mathbf{u}_\ell)_i + \sum_{j\neq i}(A_\ell)_{ij}^{off}(\mathbf{u}_\ell)_{ji})$
  Accumulate $(\mathbf{r}_\ell)_i$
  $(\mathbf{r}_{\ell-1})_i = (R_\ell)_{ii}(\mathbf{r}_\ell)_i + \sum_{j\neq i}(R_\ell)_{ij}^{off}(\mathbf{r}_\ell)_{ji}$ (parallel restriction)
  $(\mathbf{e}_{\ell-1})_i = \mathbf{0}$
  `PVcycle`$((A_{\ell-1})_i, (\mathbf{e}_{\ell-1})_i, (\mathbf{r}_{\ell-1})_i, \ell-1)$
  Accumulate $(\mathbf{e}_{\ell-1})_i$
  $(\mathbf{e}_\ell)_i = (P_\ell)_{ii}(\mathbf{e}_{\ell-1})_i + \sum_{j\neq i}(P_\ell)_{ij}^{off}(\mathbf{e}_{\ell-1})_{ji}$ (parallel interpolation)
  $(\mathbf{u}_\ell)_i = (\mathbf{u}_\ell)_i + (\mathbf{e}_\ell)_i$
  $(\mathbf{u}_\ell)_i =$ `jac_gs`$((A_\ell)_i, (\mathbf{u}_\ell)_i, (\mathbf{b}_\ell)_i)$ (parallel post-smoothing)
end if

The conditional semi-coarsening strategy is also altered in parallel, as discussed in Section 6.2.4. We enforce the additional constraint that coarsening cannot occur across subdomain boundaries whilst also maintaining the quasi-uniform mesh widths in the θ-direction. The parallel algorithm for conditional semi-coarsening is given in Algorithm 6.3.

6.3 Parallel Numerical Results – Speedup and Scaling

We test the parallel NUMG code for both the 2D and 3D elliptic problems. Firstly we test the code for the nondimensionalised 2D Poisson equation (5.5.1) on the unit sphere with $L_\phi = L_\theta = 1$, with results given in Section 6.3.1. Then in Section 6.3.2 the code is tested on the nondimensionalised 3D problem (5.6.1) on a spherical shell, with $L_\theta = L_\phi = 1$ and $L_r = 10^{-4}$, and using a graded mesh in the radial direction, i.e. the problem closely resembles the Quasi-Geostrophic Omega equation (2.2.16) solved on a typical grid used by the Met Office. We test the code on two different clusters, a 64-bit AMD Opteron 2210 cluster (`wolf`) with a total of 24 processors (the same machines as used in Chapter 5) and a 64-bit Intel Xeon E5462 cluster (`aquila`) with 2GB memory and 3MB Cache per processor. `aquila` has a total of over 800 processors, but it has only been possible to access a maximum of 256 processors at once. Both clusters use an Infinipath network.

Algorithm 6.3 Parallel conditional semi-coarsening for the Poisson-type equation on the sphere: `Cond_parallel`$(h_\theta^{(\ell)}, h_\phi^{(\ell)}, n_\theta^{(\ell)}, n_\phi^{(\ell)}, \ell, h_\theta^{(\ell-1)}, h_\theta^{(\ell-1)}, n_\theta^{(\ell-1)}, n_\phi^{(\ell-1)})$

$incr = 1$ (incrementing up the ϕ-line)
for $j = 1,\ n_\phi^{(\ell)}$
  $ratio = (L_\phi/L_\theta)(2h_\theta^{(\ell)}/h_{\phi,incr}^{(\ell)})^2 \sin^2(\pi\phi_{incr})$
  if $ratio \geq 0.5$ **and** line $incr \neq$ a processor boundary **then**
    $h_{\phi,j}^{(\ell-1)} = h_{\phi,incr}^{(\ell)} + h_{\phi,incr+1}^{(\ell)}$ (extra condition enforced for full
    $incr = incr + 2$ coarsening in ϕ-direction at line $incr$)
  else
    $h_{\phi,j}^{(\ell-1)} = h_{\phi,incr}^{(\ell)}$
    $incr = incr + 1$
  end if
  if $incr > n_\phi^{(\ell)}$ **exit**
end for
$incr = 1$ (incrementing up the θ-line)
for $k = 1,\ n_\theta^{(\ell)}$
  if $\mathrm{mod}(n_\theta, 2) == 0$ **or** line $incr \neq$ a processor boundary **or**
    $h_{\theta,incr}^{(\ell)} \leq 0.5 h_{\theta,incr+1}^{(\ell)}$ **then**
    $h_{\theta,k}^{(\ell-1)} = h_{\theta,incr}^{(\ell)} + h_{\theta,incr+1}^{(\ell)}$ (full coarsening will always occur unless
    $incr = incr + 2$ coarsening across a processor boundary)
  else
    $h_{\theta,k}^{(\ell-1)} = h_{\theta,incr}^{(\ell)}$ (semi coarsening used to prevent
    $incr = incr + 1$ coarsening across a processor boundary)
  end if
  if $incr > n_\theta^{(\ell)}$ **exit**
end for
$n_\phi^{(\ell-1)} = j,\ n_\theta^{(\ell-1)} = k$

6.3.1 2D results

Recall problem (5.5.1) with $L_\theta = L_\phi = 1$:

$$-\frac{\partial}{\partial\phi}\left(\frac{\sin(\pi\phi)}{\pi^2}\frac{\partial u}{\partial\phi}\right) - \frac{\partial}{\partial\theta}\left(\frac{1}{4\pi^2\sin(\pi\phi)}\frac{\partial u}{\partial\theta}\right) = g_{2D}\sin(\pi\phi)\,, \quad \Omega = (0,1)^2\,,$$

which is the Poisson equation in spherical polar coordinates. Recall that we use periodic conditions in the boundary coinciding with $\lambda = 0$ and $\lambda = 1$, and polar boundary conditions at $\phi = 0$ and $\phi = 1$. We firstly perform a speedup (strong scalability) test of the NUMG code, which we recall is a measure of the performance gain of a parallel

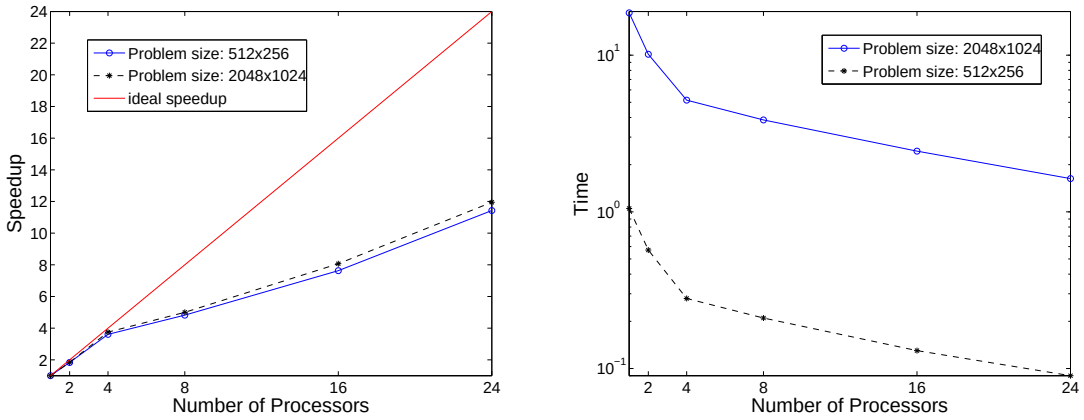

Figure 6-9: Speedup (strong scalability) test of the 2D NUMG method on `wolf` for two different problem sizes.

code running on N processors over the sequential code where the global problem size remains unchanged. We test this on problem sizes 512×256 and 2048×1024, with the results shown in Figure 6-9. The speedup is very good (almost optimal) on up to about 4 processors. However, for larger numbers of processors the amount of work that each of the processors has to do becomes smaller relative to the amount of communication required, and so the speedup drops off slightly from the optimal (linear) growth. The speedup is slightly better for the larger problem size, which is an expected result since the larger each subdomain the smaller the relative amount of communication.

A better test for how well the implementation scales on larger numbers of processors is a scaled efficiency (weak scalability) test, where the problem size per processor is fixed as the number of processors is increased. In this test the CPU time of a method that has optimum weak scalability should remain constant as the number of processors is increased. Figures 6-10 and 6-11 show the weak scalability of the method on the two different clusters with respect to CPU time and iterations. In Figure 6-10 the problem size per processor is 1920×960 on `wolf`, and observe that the method has almost optimal weak scalability, particularly beyond four processors. With 24 processors, it was possible to solve the problem on the entire globe with a horizontal grid resolution of approximately 3.5km at the equator and 4.5×10^7 unknowns in just over 30 seconds.

On `aquila` we fix the problem size per processor to 512×256 and tested the parallel performance of the code for much larger clusters of up to 256 processors. We observe very good parallel scaling with a slight increase of the CPU time for more than 96 processors. This is because the number of iterations jumps from about 9 to 12 when using more than 96 processors, potentially because of the hybrid smoother, where the

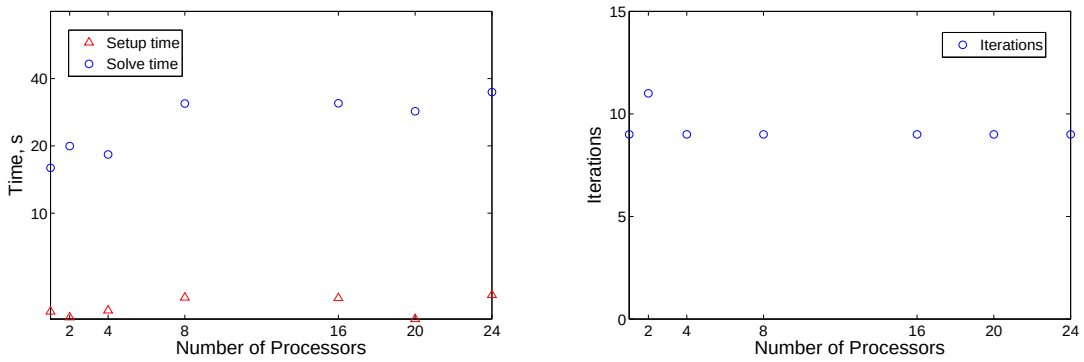

Figure 6-10: Scaled efficiency (weak scalability) test of 2D NUMG on `wolf`: Problem size 1920×960 on each processor.

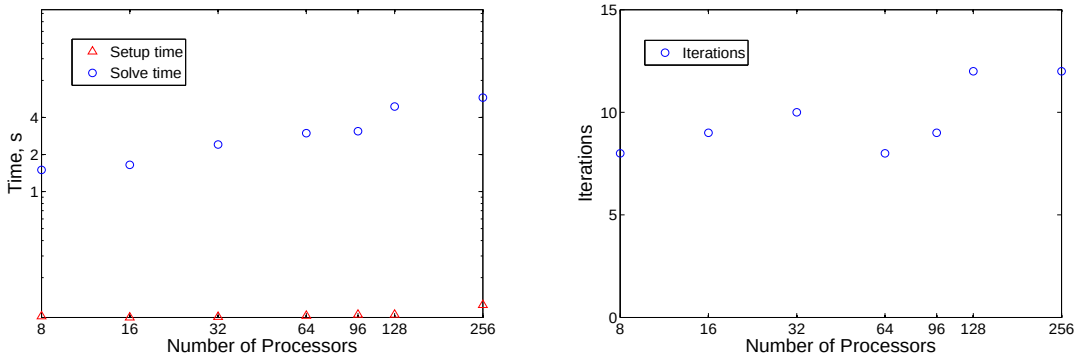

Figure 6-11: Scaled efficiency (weak scalability) test of 2D NUMG on `aquila`: Problem size 512×256 on each processor.

Jacobi method – which is less effective as a smoother than the Gauss-Seidel method (cf. 6.2.4) – has to be used at more nodes on the domain. However, this increase in iterations when using more than 96 processors is not significant, a factor of only 1.3, and so good scalability is observed even with a very large number of processors. With 256 processors, the problem was solved on the globe with a resolution of about 5km and 3.5×10^7 unknowns in as little as 6 seconds. With larger problem sizes per processor, the problem can be solved on even finer resolutions, though the main purpose of the tests on `aquila` was to demonstrate the scalability of the method on large clusters.

The results are very encouraging, particularly since the Met Office seeks to increase the grid resolution and number of processors used for solving their elliptic problems. The results have shown that both of these changes can clearly be handled effectively by the parallel NUMG method.

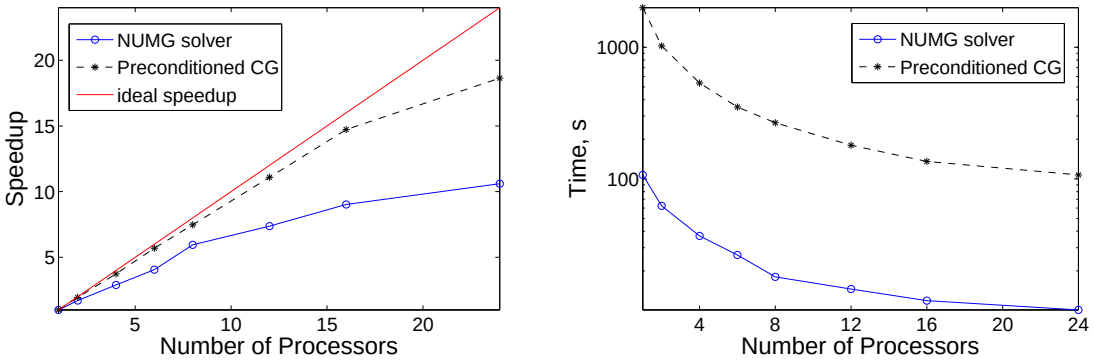

Figure 6-12: Speedup (strong scalability) test on `wolf` for 3D NUMG (solid blue line) and the r-line preconditioned CG method (dotted black line). Global problem size: $360 \times 180 \times 100$

6.3.2 3D results

We test the performance of NUMG on problem (5.6.1), i.e.

$$-\frac{\partial}{\partial r}\left(\alpha_1(\boldsymbol{\xi})\frac{\partial u}{\partial r}\right) - \frac{\partial}{\partial \phi}\left(\alpha_2(\boldsymbol{\xi})\frac{\partial u}{\partial \phi}\right) - \frac{\partial}{\partial \theta}\left(\alpha_3(\boldsymbol{\xi})\frac{\partial u}{\partial \theta}\right) = f\,,$$

on $\Omega = (0,1)^3$, where

$$\begin{aligned}
\alpha_1(\boldsymbol{\xi}) &= \alpha_1(r,\phi,\theta) = L_r(\phi,\theta)(a+dr)^2\sin(\pi\phi)\,,\\
\alpha_2(\boldsymbol{\xi}) &= \alpha_2(r,\phi,\theta) = L_\theta(r,\theta)\sin(\pi\phi)/\pi^2\,,\\
\alpha_3(\boldsymbol{\xi}) &= \alpha_3(r,\phi,\theta) = L_\phi(r,\phi)/(4\pi^2\sin(\pi\phi))\,.
\end{aligned}$$

Recall that a and d represent the radius of the Earth and the depth of the atmosphere, respectively. The boundary conditions we use are periodic at $\lambda = 0$ and $\lambda = 1$, polar at $\phi = 0$ and $\phi = 1$ and Dirichlet at $r = 0$ and $r = 1$. In addition, we set $L_\theta = L_\phi = 1$ and $L_r = 10^{-4}$. Recall from Section 5.6 that we are also interested in the performance of the Poisson-type equation which we obtain via setting $L_r = 1$. However, since solving the Poisson-type equation is essentially easier (see Table 5.21), we give the parallel results only for the more difficult case, i.e. for $L_r = 10^{-4}$.

We firstly test the speedup (strong scalability) of NUMG on a problem size of $360 \times 180 \times 100$, with the results shown in Figure 6-12. The speedup is very good (almost optimal) on up to about 8 processors. However, as with the 2D results, the speedup drops off slightly from the optimal growth for larger numbers of processors because the amount of work that each of the processors has to do becomes too small.

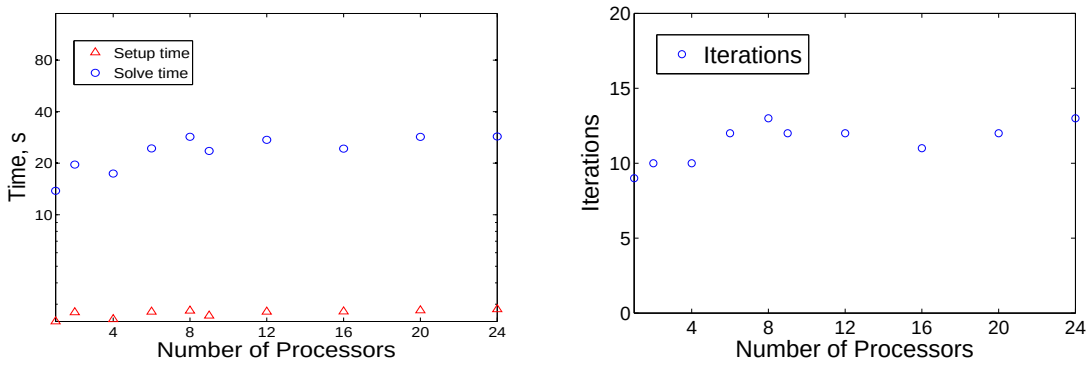

Figure 6-13: Scaled efficiency (weak scalability) test on `wolf` for 3D NUMG: Problem size $200 \times 100 \times 50$ on each processor.

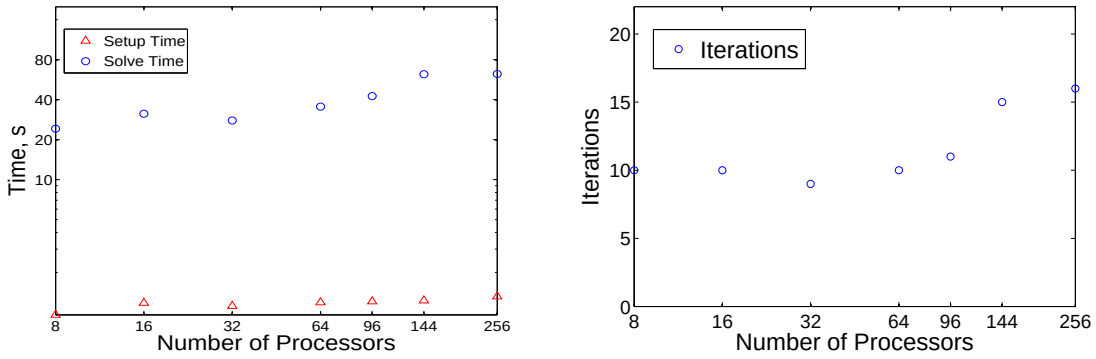

Figure 6-14: Scaled efficiency (weak scalability) test on `aquila` for 3D NUMG: Problem size $192 \times 120 \times 50$ on each processor.

In the scaled efficiency (weak scalability) test, Figures 6-13 and 6-14 show how the method scales on the two different clusters with respect to CPU time and the number of iterations. In Figure 6-13 we fix the problem size per processor to $200 \times 100 \times 50$ on `wolf`, and observe that the method scales almost optimally, particularly beyond four processors. On `aquila` we fix the problem size per processor to $192 \times 120 \times 50$ and observe also very good parallel scaling with a slight increase of the CPU time for more than 96 processors. This is because the number of iterations jumps from about 10 to about 15 when using more than 96 processors. This may be caused by the hybrid smoother or by the reduced anisotropy in the r-direction (as the grid is more refined in the horizontal direction than in the vertical). However, this trend was also observed in Figure 6-11 for the 2D scaling results on `aquila`, indicating that the hybrid smoother is likely to be the main cause of the increase in iterations. Despite the small increase

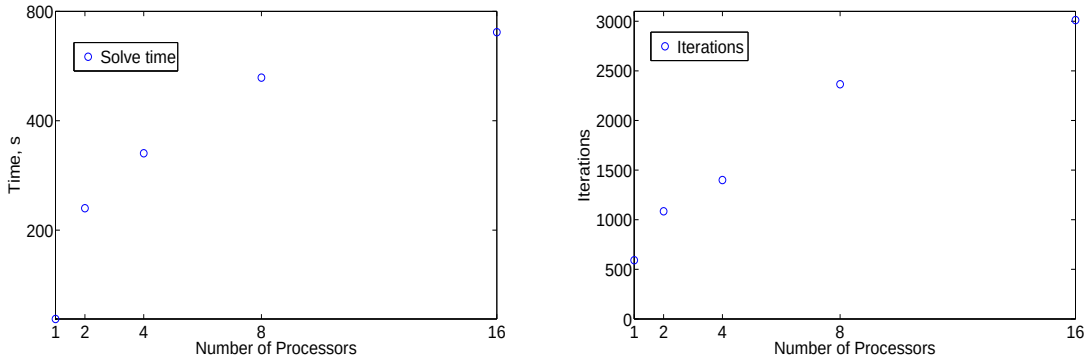

Figure 6-15: Scaled efficiency (weak scalability) test on `wolf` for the r-line preconditioned CG method: Problem size $200 \times 100 \times 50$ on each processor.

in iterations, we see that on 256 processors with the new parallel 3D NUMG method, the problem on the entire globe with a resolution of 10km at the equator and 3×10^9 unknowns was solved in about 60 seconds. This is extremely fast, particularly since the method has to deal with anisotropies present in two coordinate directions which change throughout the domain. On larger clusters, potentially even finer resolutions could be solved within the same time scale.

Now let us compare the parallel NUMG solver with parallelizations of the other methods. Krylov subspace methods with r-line Gauss–Seidel preconditioners (such as the ones used at the Met Office) are extremely well suited to an efficient parallelization, and the numerical results in Figure 6-12 show this clearly. For our experiments, we used the r-line preconditioned conjugate gradient (CG) method, and the speedup is almost optimal (linear) on any number of processors. However, as we saw in Section 5.6, the method is not robust, i.e. the number of iterations grows with the problem size, and for a grid resolution of $360 \times 180 \times 100$, the NUMG solver is over 10 times faster than r-line Jacobi preconditioned CG on one processor which we can see in Figure 6-12. Even when increasing the number of processors for the same problem size, the multigrid method is still about 10 times faster than preconditioned CG which is substantial.

For finer grid resolutions, such as the ones used in the weak scalability tests in Figures 6-13 and 6-14, the r-line preconditioned CG method becomes increasingly inferior to NUMG. This is demonstrated with a scaled efficiency (weak scalability) test for the preconditioned CG method. We see in Figure 6-15 that the method has poor weak scalability since the number of iterations using this method grows linearly with problem size, unlike NUMG. Hence the CPU time also grows at a similar rate, and with the problem size on 16 processors it takes over 20 times longer than NUMG. Therefore,

despite the slightly more efficient parallelization, it is never likely to outperform NUMG even with hundreds of processors, particularly as the problem size increases. This is a particularly important point to highlight since the results in this section have clearly shown that the NUMG method can handle an increase in grid resolution that is expected to occur within the VAR system.

Finally we demonstrate the practical use of the parallel NUMG code within the VAR system for solving the Quasi-Geostrophic Omega equation. Figure 6-16 is a visualization of the vertical velocity profile at θ-level 20 on the Charney-Phillips grid, using the N72 analysis grid which is used for test problems and has a resolution of $144 \times 109 \times 70$. This is exactly the same snapshot as Figure 5-6 but generated using eight processors instead of one. The two figures show exactly the same image, with no additional errors on the processor interfaces.

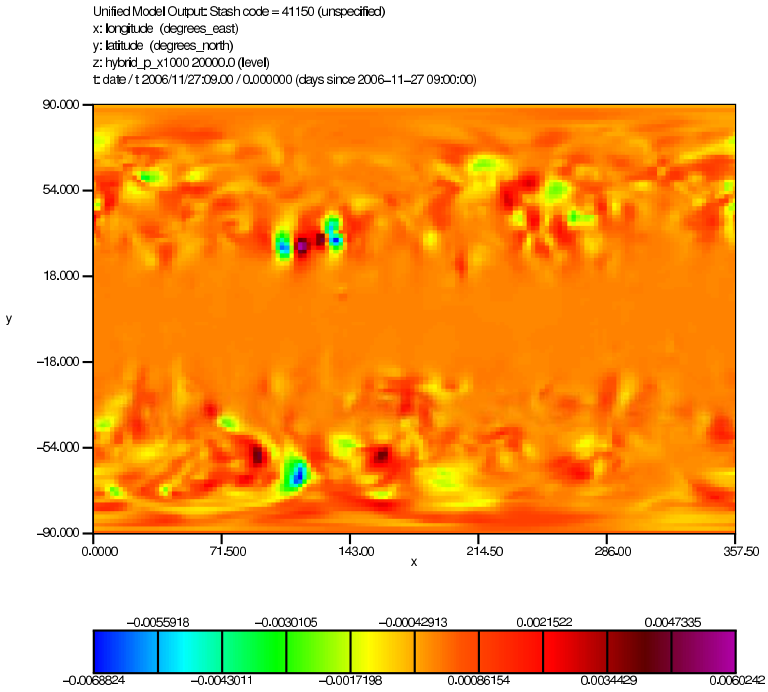

Figure 6-16: Snapshot of the vertical velocity, w, after one iteration of VAR at θ-level 20 (roughly 8.8km above ground level), using eight processors

Chapter 7

Application of Multigrid in the Potential Vorticity Based Control Variable Transform

In this final chapter we discuss a novel robust method for solving the balanced potential vorticity (PV) equation (2.3.7) and the unbalanced anti-PV equation (2.3.9) arising within the PV-based control variable transform (CVT), as described in Chapter 2.

The vorticity based CVT currently operational at the Met Office has various limitations as discussed in Section 2.3 (see also [53]). Hence a new CVT based on potential vorticity (PV) variables has been thoroughly investigated in, for example [5, 25]. This new scheme overcomes the limitations of the old CVT and should replace it in the operational system if it is feasible to implement and if its computational cost is not significantly greater.

The main issue arising from the PV-based CVT is the necessity to solve highly ill-conditioned three dimensional (3D) elliptic problems that are not present in the old CVT. The operators of these 3D problems are written abstractly in the form

$$-A\nabla_r^2 u \; - B\frac{\partial^2}{\partial r^2}\left(\nabla_r^{-2}\nabla_r \cdot f\nabla_r u\right) \; = \; g\,, \tag{7.0.1}$$

i.e. they have a two dimensional (2D) solve embedded in them and so it is not possible to discretise them straightforwardly. Moreover the problems are so ill-conditioned that the solvers used by the Met Office have been unsuccessful in approximating the solutions to them, even to a low residual tolerance.

Unfortunately the non-uniform multigrid method (NUMG) devised in Chapter 5 cannot directly be used to solve these problems due to the lack of a discrete operator.

However it is possible to use Krylov subspace methods since these only involve *applying* the operator, which can be done without an explicit matrix. Nevertheless, due to the ill-conditioned nature of the problem, the Krylov method will need to be accelerated by a preconditioner. A novel approach for doing this is to use NUMG to solve a simplified form of (7.0.1). The simplified problem will have no 2D solve embedded in it so that it can be discretised. Robust multilevel preconditioners accelerate Krylov methods to an extent that they perform optimally, so this new approach should not only solve the 3D problems in the PV-based CVT, but it should solve them very efficiently depending on how well the simplified form of (7.0.1) approximates the original operator and how efficiently NUMG can solve the simplified problem.

In Section 7.1 we recall the vorticity based CVT that is currently used in the operational data assimilation code at the Met Office and how the limitations of this method are overcome by introducing new control variables based on PV. The exact steps of the PV-based CVT are also given, with details on the equations that need to be solved at each step and where the variables are located on the grid. Then in Section 7.2, we investigate the methods for solving the main equations present in the new transformation, and how NUMG can be used to great effect. Numerical results are given in Section 7.3 for a simplified test problem, where we not only demonstrate the performance of the new method for solving the 3D problems, but also show that the full cycle of transformations yields accurate results. Finally in Section 7.4, we give the same numerical results for the full PV-based CVT.

7.1 PV-Based Control Variable Transform

Recall from Chapter 2 that the CVT allows the VAR cost function (2.2.2) to be expressed in terms of certain new variables that have fewer correlations thus reducing the density of the background error covariance matrix. The CVT is the transformation between the usual model variables, e.g. wind velocity, pressure and potential temperature, and a set of variables called control variables. The state of the system in terms of the model variables $\mathbf{x}$ is written as a sum of the background state $\mathbf{x}_b$ and an increment $\mathbf{x}'$. Minimizing the VAR cost function with respect to $\mathbf{x}'$ is problematic as the background error covariance matrix is large, not explicitly known and contains complicated correlations between errors in the model variables, as stated in [25]. In terms of the control variable increments, $\mathbf{v}'$, however, the cost function simplifies because the correlations between the variables are much smaller and so the background error covariance matrix can be approximated to a good degree of accuracy by a block diagonal matrix (cf. [53]). Special transformations are still necessary to reduce the background error

covariance matrix to the identity matrix, as described in Section 2.2.3 (cf. [49]).

We also recall from Chapter 2 that the control variables are chosen so that they represent normal modes of the system. In a linear shallow water system, one third of the normal modes are characterized as balanced, i.e. corresponding to Rossby waves, and two thirds as unbalanced, i.e. corresponding to inertia-gravity waves. As stated in [5], balanced components of flow are associated solely with PV whilst unbalanced components of flow have no PV and are associated with anti-PV. Thus PV only exists within balanced components of flow, where anti-PV = 0, and similarly, only the unbalanced variables can influence anti-PV, where PV = 0. The same concept can be extended to a 3D atmosphere, assuming that there are no significant correlations between the balanced and unbalanced normal modes. The choice of control variables is made based on this concept, choosing one variable that represents the balanced normal mode and two that represent the unbalanced modes, so that they have an uncorrelated property. The "vorticity-based" control variables which are currently used in the Met Office data assimilation system, described in [80], attempt to capture the balanced part of the flow entirely by the streamfunction. A limitation to this method is that the streamfunction is assumed to be a completely balanced variable with no allowance for an unbalanced component. However, in general, the streamfunction and pressure variables have both balanced and unbalanced components as follows

$$\psi = \psi_u + \psi_b \,,$$
$$p = p_u + p_b \,.$$

This suggests that the traditional vorticity based CVT is suboptimal, and so a new set of control variables based on PV [5, 53, 81] have been introduced. We recall from Section 2.3 that these are the velocity potential, χ, the balanced streamfunction, ψ_b and the unbalanced pressure, p_u. The new "PV-based" CVT, formulated in [25], recognizes the presence of an unbalanced streamfunction and exploits the association between PV and the balanced component of the flow, hence should not suffer the shortcomings of the vorticity-based CVT.

We now focus on implementing the PV-based CVT, where we only need to use the model variable increments u', v' and p' since the calculation of the remaining model variable increments (cf. (2.2.13) – (2.2.16)) are not affected by the new CVT. Recall that a staggered Arakawa C-grid is used in the horizontal (cf. Figure 2-2) and a Charney-Phillips grid in the vertical (cf. Figure 2-3), and each of the variables are located at specific points on the grid.

Firstly the transformation from the model to control variables, i.e. the **T**-transform,

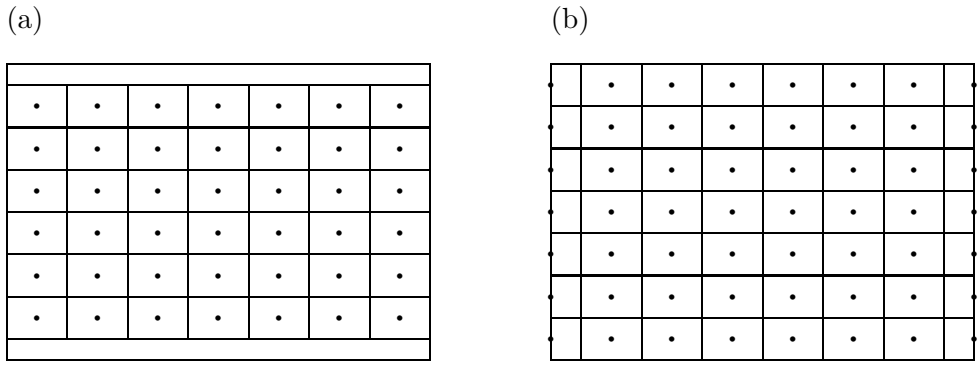

Figure 7-1: (a) The finite volume grid with (a) p'-points as cell centres and (b) ψ'-points as cell centres

is defined by

$$\mathbf{v}' = \mathbf{T}\mathbf{x}' \,,$$

with

$$\mathbf{x}' = (u', v', p') \qquad \text{and} \qquad \mathbf{v}' = (\chi', \psi'_b, p'_u) \,.$$

Now, given the model variables, the implementation of the **T**-transform is summarized via a succession of:

- finite difference calculations,
- interpolation of values between points on the grid,
- solving 2D and 3D elliptic problems,

and is done via an intermediate set of variables, namely the horizontal divergence, D, the PV, Q, and the anti-PV, $\overline{Q}$. The detailed sequence of steps in the **T**-transform is as follows:

Step 1 : $(u', v', p') \rightarrow (D', \psi', p')$

The model variables u' and v' are located on the edges of each cell in the Arakawa C-grid and on the ρ-levels in the vertical direction on the Charney-Phillips grid (see Figures 2-2 and 2-3). p' is found in the centre of each cell (including the pole nodes) and also on the ρ-levels, and we call these p'-points (see Figure 7-1(a)). The horizontal divergence increment, D', is calculated as the divergence of the horizontal velocity field, i.e.

$$D' = \nabla_r \cdot \mathbf{u}' = \frac{1}{a \sin\phi} \left(\frac{\partial u'}{\partial \lambda} + \frac{\partial (v' \sin\phi)}{\partial \phi} \right) \,,$$

where $\mathbf{u}' = (u', v')$. Recall that we solve all equations in a spherical geometry, thus $\phi \in [0, \pi]$ and $\lambda \in [0, 2\pi]$. A constant radius of a is used since the normal mode analysis is done with the shallow atmosphere approximation. The derivatives are calculated using finite differences. Since u' and v' are located on the edges of each cell, D' is found naturally at the p'-points. Since the poles are also located at cell centres, D' is derived at the poles using the divergence theorem:

$$A(k)D' = h_\lambda \sum_{i=1}^{n_\lambda} v'(i, h_\phi/2, k)\,, \qquad k = 1, \ldots, n_r$$

where $A(k) = \pi(r_k h_\phi/2)^2$ is the approximate area of the pole cell at level k in the r-direction, h_λ and h_ϕ are the mesh widths in the λ- and ϕ-directions respectively and n_λ and n_r are the number of grid points in the λ- and r-directions respectively.

The vorticity, ζ', is calculated as the curl of the velocity field, i.e. $\zeta' = \nabla_r \times \mathbf{u}'$, and is found naturally on the corners of each cell and on ρ-levels. We call these ψ'-points (see Figure 7-1(b)). By the Helmholtz decomposition, the horizontal velocities can be decomposed into rotational and divergent parts as in (2.2.10). Using the Helmholtz decomposition and the equation for the vorticity, the streamfunction, located at the ψ-points, is obtained by solving

$$\nabla_r^2 \psi' = \frac{1}{a \sin\phi} \left(\frac{\partial v'}{\partial \lambda} - \frac{\partial (u' \sin\phi)}{\partial \phi} \right), \tag{7.1.1}$$

where

$$\nabla_r^2 = \frac{1}{a^2 \sin\phi} \left(\frac{\partial}{\partial \phi} \left(\sin\phi \frac{\partial}{\partial \phi} \right) + \frac{1}{\sin\phi} \frac{\partial^2}{\partial \lambda^2} \right).$$

This 2D Poisson solve must be solved on each ρ-level. It is solved with periodic and polar boundary conditions defined in Section 3.2. As a result the system is rank deficient and is unique only up to a constant, as stated in Section 3.2.4, and must be solved using the techniques discussed in Section 5.5.2. For the experiments in this chapter, the constant is set to zero by projection (cf. Section 5.5.2). Variables located at ψ'-points are not defined at the poles.

Step 2 : $(D', \psi', p') \to (D', Q', \overline{Q}')$

We use (2.3.1) to calculate Q' from p' and ψ', i.e.

$$Q' = \alpha_0 \nabla_r^2 \psi' + \beta_0 p' + \gamma_0 \frac{\partial p'}{\partial r} + \varepsilon_0 \frac{\partial^2 p'}{\partial r^2}\,. \tag{7.1.2}$$

α_0, β_0, γ_0 and ε_0 are reference state values, defined in [4, Section 5.1]. Before calculating Q', however, we realize that not all variables are located at the same

place. We require Q' on the ψ'-points, but p' is located at the p'-points. Therefore, p' must be interpolated onto the ψ'-points before (7.1.2) can be applied.

To calculate $\overline{Q}'$ from p' and ψ', we use (2.3.5), i.e.

$$\overline{Q}' = \nabla_r \cdot (f\rho_0 \nabla_r \psi') - \nabla_r^2 p' \,, \tag{7.1.3}$$

where ρ_0 is a reference density value and $f = -2\Omega\cos(\phi)$ is the Coriolis force calculated using the Earth's angular velocity of $\Omega = 7.292 \times 10^{-5}\ rad/s$. $\overline{Q}'$ is required at the p'-points so ψ' is interpolated before (7.1.3) is applied.

Step 3 : $(D', Q', \overline{Q}') \to (\chi', \psi_b', p_u')$

Use (2.3.6) to substitute p_b' into (2.3.3), and since the unbalanced components have no PV, the following 'balanced equation' is solved using just for the balanced component of the streamfunction, i.e. ψ_b':

$$\boxed{\begin{aligned} \alpha_0 \nabla_r^2 \psi_b' + \beta_0 \left(\nabla_r^{-2} \nabla_r \cdot f\rho_0 \nabla_r \psi_b'\right) + \gamma_0 \frac{\partial}{\partial r}\left(\nabla_r^{-2} \nabla_r \cdot f\rho_0 \nabla_r \psi_b'\right) + \\ \varepsilon_0 \frac{\partial^2}{\partial r^2}\left(\nabla_r^{-2} \nabla_r \cdot f\rho_0 \nabla_r \psi_b'\right) \;=\; Q' \end{aligned}} \tag{7.1.4}$$

This equation is solved using periodic boundary and polar boundary conditions on the ϕ-λ plane, and Neumann boundary conditions at the r-boundaries. The solution to the 2D Poisson solves in (7.1.4) are unique only up to a constant. However, if this constant is fixed to zero, then (7.1.4) has a unique solution despite the Neumann boundary conditions, thanks to presence of the zeroth order term in the 3D problem. ψ_b' is found on the ψ'-points.

To obtain p_u', firstly we solve (2.3.4), which we recall is

$$\nabla_r^2 \xi' = \overline{Q}' \,, \tag{7.1.5}$$

on each ρ-level for ξ' – found at the p'-points – and then use (2.3.8) to substitute p_u' into (2.3.2). Since balanced components have no anti-PV, the following 'unbalanced equation' is solved just for the unbalanced component of the streamfunction, i.e. ψ_u':

$$\boxed{\begin{aligned} \alpha_0 \nabla_r^2 \psi_u' + \beta_0 \left(\nabla_r^{-2} \nabla_r \cdot f\rho_0 \nabla_r \psi_u'\right) + \gamma_0 \frac{\partial}{\partial r}\left(\nabla_r^{-2} \nabla_r \cdot f\rho_0 \nabla_r \psi_u'\right) + \\ \varepsilon_0 \frac{\partial^2}{\partial r^2}\left(\nabla_r^{-2} \nabla_r \cdot f\rho_0 \nabla_r \psi_u'\right) \;=\; \beta_0 \xi' + \gamma_0 \frac{\partial \xi'}{\partial r} + \varepsilon_0 \frac{\partial^2 \xi'}{\partial r^2} \end{aligned}} \tag{7.1.6}$$

which is found at the p'-points, with the same boundary conditions as for (7.1.4). Then p'_u can be found at the p'-points by solving

$$\nabla_r^2 \left(p'_u + \xi'\right) = \nabla_r \cdot \left(f\rho_0 \nabla_r \psi'_u\right), \tag{7.1.7}$$

i.e. (2.3.8) on each grid level.

Note that there is an alternative equation that can be solved for p'_u at the p'-points, as given in [5, Section 4.7.2]. However, it is simpler to solve (7.1.6) and (7.1.7) since (7.1.6) is very similar to (7.1.4), only with a different right-hand-side.

Finally, we deduce from the Helmholtz decomposition that

$$\nabla_r^2 \chi' = D', \tag{7.1.8}$$

which we solve on each ρ-level to find χ' at the p'-points.

This completes the **T**-transform. To recapitulate, we have to solve two 3D systems, (7.1.4) and (7.1.6), and four sets of 2D problems, namely (7.1.1), (7.1.5), (7.1.7) and (7.1.8), on each grid level. Note that in practice only one 3D solve, namely (7.1.4), is needed. Since we know ψ' from (7.1.1), we can easily find ψ'_u be method of subtraction, then use (7.1.7) to find p'_u. However, since the purpose of the experiments in this section is to show that the full cycle of the PV-based CVT works, we must perform each 3D solve independently to be certain that the 3D solver is accurate and to consequently ensure no errors are carried over between variables.

We now present the inverse problem, the **U**-transform, i.e.

$$\mathbf{x}' = \mathbf{U}\mathbf{v}'.$$

Given the PV-based control variable increments, the **U**-transform proceeds as follows:

Step 1 : $(\chi', \psi'_b, p'_u) \rightarrow (\chi', \psi', p')$

Given ψ'_b we can find p'_b by interpolating ψ'_b onto the p'-points and solving

$$\nabla_r^2 p'_b = \nabla_r \cdot \left(f\rho_0 \nabla_r \psi'_b\right). \tag{7.1.9}$$

on each ρ-level. This gives p'_b on the p'-points and since we already have p'_u on the p'-points, we find p', also on the p'-points, by

$$p' = p'_u + p'_b.$$

Now, given p'_u, we obtain ψ'_u by interpolating p'_u onto the ψ'-points and solving

$$\nabla_r^2 \psi'_u = (\alpha_0)^{-1} \left\{ -\beta_0 p'_u - \gamma_0 \frac{\partial p'_u}{\partial r} - \varepsilon_0 \frac{\partial^2 p'_u}{\partial r^2} \right\} \tag{7.1.10}$$

on each ρ-level. We now have ψ'_u and ψ'_b on the ψ'-points, so ψ' is found by

$$\psi' = \psi'_u + \psi'_b \,.$$

Step 2 : $(\chi', \psi', p') \rightarrow (u', v', p')$

The Helmholtz decomposition yields

$$u' = -\frac{1}{a}\frac{\partial \psi'}{\partial \phi} + \frac{1}{a \sin\phi}\frac{\partial \chi'}{\partial \lambda} \,,$$
$$v' = \frac{1}{a \sin\phi}\frac{\partial \psi'}{\partial \lambda} + \frac{1}{a}\frac{\partial \chi'}{\partial \phi} \,.$$

to obtain u' and v' at the edges of each cell. u' is also defined at the poles where it is simply set to $u' = 0$.

This completes the **U**-transform, where we had to solve two further sets of 2D problems, namely (7.1.9) and (7.1.10). Note that both the 3D problems are solved on the ψ'-points, but some of the 2D problems are also solved on the p'-points. Thus, in order to discretise these problems using the finite volume method, two grids are required, one whose cell centres are the p'-points (see Figure 7-1(a)) and another whose cell centres are the ψ'-points (see Figure 7-1(b)). Both p'- and ψ'-points are located on the ρ-levels, so interpolation between these points only requires averaging in the horizontal direction. The number of p'-points on the grid is $n_\lambda \times n_\phi + 2$ (see Section 3.2.4) and the number of ψ'-points is $n_\lambda \times (n_\phi + 1)$, noting that they are not located at the poles.

Clearly, the **T**-transform poses the biggest difficulties with the two 3D solves highlighted in step 3. The **U**-transform, in contrast, contains only two 2D Poisson solves on each ρ-level and some finite difference calculations. Thus, the majority of the cost of the PV-based CVT is the time taken for solving the 3D problems in the **T**-transform, and in the following section we investigate methods for solving these problems.

7.2 Strategy for Solving the Balanced PV-Equation

In this section we focus on the difficulties that arise in attempting to solve the balanced PV-equation (7.1.4) and possible methods that can be used to overcome these difficulties. The unbalanced equation (7.1.6) can be solved identically. Currently at the Met

Office, the generalized conjugate residual (GCR) method [34] is used to make an attempt at solving the equation, but even with a high number of iterations, convergence does not occur to a satisfactory tolerance.

We begin by tackling a simplified model problem, i.e. removing the zeroth and first order terms from (7.1.4), i.e.

$$\alpha_0 \nabla_r^2 \psi_b' + \varepsilon_0 \frac{\partial^2}{\partial r^2} \left(\nabla_r^{-2} \nabla_r \cdot f \rho_0 \nabla_r \psi_b' \right) \; = \; Q' \,, \tag{7.2.1}$$

where we set $\alpha_0 = N^2(r)$ and $\varepsilon_0 = f/\rho_0$, which are reasonably close to their true values. We also use the same boundary conditions as for (7.1.4) and (7.1.6). By removing the zeroth order term, the problem is now singular, thus the solution will be unique only up to a constant. The immediate issue we observe is that the operator ∇_r^{-2} is used only to indicate that a 2D Poisson solve is embedded within the 3D problem, and so it cannot be discretised. As a result, (7.2.1) cannot be solved using NUMG alone, but we incorporate NUMG within the development of an alternative strategy.

Let us introduce the following operator:

$$\mathcal{A} = \alpha_0 \nabla_r^2 + \varepsilon_0 \frac{\partial^2}{\partial r^2} \left(\nabla_r^{-2} \nabla_r \cdot f \rho_0 \nabla_r \right) \,.$$

Since $\mathcal{A}$ cannot be discretised, the main task is in finding a method of solving (7.2.1) without the use of a discrete operator.

The proposed strategy is to use the preconditioned conjugate gradient (PCG) method (cf. Algorithm 4.3) with the following preconditioner:

$$\mathcal{P} = N^2(r) \nabla_r^2 + f_0^2 \frac{\partial^2}{\partial r^2} \,. \tag{7.2.2}$$

This is a simplified version of $\mathcal{A}$ which assumes that f and ρ_0 are constant, where $f = f_0 \approx 10^{-4}$. The operator ∇_r^{-2} has disappeared, so $\mathcal{P}$ can be discretised to form a matrix P. Hence systems involving P can solved, and we do so using NUMG. In fact, $\mathcal{P}$ resembles the operator for the Quasi-Geostrophic omega equation from Section 2.2.4, which we recall can be solved optimally using NUMG (cf. Section 5.6.4). Note that with the boundary conditions as stated above, P is only semi-definite, since the constant vector is in the nullspace of P. Hence a projection step is required on each grid level to ensure that the right-hand-side in is the range of P (recall Section 5.5.2).

Now, the remaining issue with using the PCG method is how to implement the application of the operator $\mathcal{A}$. This will involve a 2D Poisson solve and finite volume calculations of second order terms.

Algorithm 7.1 `apply_operator`$(\mathcal{A}, \mathtt{p}, \mathtt{q})$

1. Given a vector $\mathbf{p}$, the discrete analogue of a function p, containing information at each node in the 3D domain, create a vector $\mathbf{p}_{2D}$ containing the same information as $\mathbf{p}$ but only on the first layer, i.e. on the first ρ-level.
2. Let A_f be a finite volume discretisation of $\nabla_r \cdot (f\rho_0 \nabla_r)$. Premultiply $\mathbf{p}_{2D}$ with A_f to obtain $\mathbf{c}$, the discrete analogue of $\nabla_r \cdot (f\rho_0 \nabla_r p_{2D})$.
3. Let A_{2D} be a finite volume discretisation of ∇_r^2. Solve $\nabla_r^2 p^*_{2D} = \nabla_r \cdot (f\rho_0 \nabla_r p_{2D})$ using a 2D NUMG solver to obtain $\mathbf{p}^*_{2D} = A_{2D}^{-1}\mathbf{c}$. The solution is unique only up to a constant, which is set to zero.
4. Repeat steps 1-3 on each ρ-level to obtain $\mathbf{p}^*$ which contains $\mathbf{p}^*_{2D}$ on every ρ-level.
5. Using a finite volume discretisation, A_2, of the second order term $\frac{\partial^2}{\partial r^2}$, we apply $\mathbf{q}_1 = \varepsilon_0 A_2 \mathbf{p}^*$
6. Obtain the discrete analogue of $\alpha_0 \nabla_r^2 p_{2D}$ by applying $\mathbf{q}_2^* = \alpha_0 A_{2D} \mathbf{p}_{2D}$ on each ρ-level. Then we have $\mathbf{q}_2$, a vector containing $\mathbf{q}_2^*$ on each layer.
7. Finally add $\mathbf{q}_1$ and $\mathbf{q}_2$ to obtain $\mathbf{q}$, the desired vector which gives the discrete values of $\alpha_0 \nabla_r^2 p + \varepsilon_0 \nabla_r^{-2} \nabla_r \cdot (f\rho_0 \nabla_r p)$ at each node in the 3D domain.

Thus the two key components of this new PCG method are:

Applying the operator $\mathcal{A}$: Replaces the traditional matrix-vector multiplication in the Krylov subspace method, since $\mathcal{A}$ cannot be discretised.

Preconditioning using the operator $\mathcal{P}$: A solve involving the discrete operator P can be done optimally using NUMG. It is a good choice for a preconditioner because $\mathcal{P}$ approximates $\mathcal{A}$ closely, particularly near the poles where $f^2 \approx f_0^2$.

The details of the sequence of steps used to apply the operator $\mathcal{A}$ are described in Algorithm 7.1.

As for the preconditioning step $\mathbf{z} = P^{-1}\mathbf{r}$, this is solved using NUMG V-cycle iterations until a specific tolerance is satisfied. A residual tolerance of 10^{-4} for the preconditioner proved to be sufficient in experiments, and stricter tolerances made no difference to the number of iterations required for the overall PCG method to converge.

The algorithm for solving (7.2.1) using PCG is given in Algorithm 7.2.

Algorithm 7.2 Preconditioned Conjugate Gradient method: `pcg(`$A, \mathtt{u}, \mathtt{b}$`)`

Choose $\mathbf{u}^{(0)}$ (initial solution) and P (preconditioner)
$\mathbf{q} = \mathtt{apply\_operator}(\mathcal{A}, \mathbf{u}^{(0)}, \mathbf{q})$ (apply $\mathcal{A}$ to $\mathbf{u}^{(0)}$)
$\mathbf{r}^{(0)} = \mathbf{b} - \mathbf{q}$
Solve $\mathbf{z}^{(0)} = P^{-1}\mathbf{r}^{(0)}$ (solve using NUMG)
$\mathbf{p}^{(0)} = \mathbf{z}^{(0)}$
for $k = 0, 1, \ldots,$ until convergence ...
  $\mathbf{q} = \mathtt{apply\_operator}(\mathcal{A}, \mathbf{p}^{(\mathtt{k})}, \mathbf{q})$ (apply $\mathcal{A}$ to $\mathbf{p}^{(k)}$)
  $\alpha_k = (\mathbf{z}^{(k)T}\mathbf{r}^{(k)})/(\mathbf{p}^{(k)T}\mathbf{q})$
  $\mathbf{u}^{(k+1)} = \mathbf{u}^{(k)} + \alpha_k \mathbf{p}^{(k)}$
  $\mathbf{r}^{(k+1)} = \mathbf{r}^{(k)} - \alpha_k A\mathbf{p}^{(k)}$
  if $\mathbf{r}^{(k+1)}$ sufficiently small **exit**
  Solve $\mathbf{z}^{(k+1)} = P^{-1}\mathbf{r}^{(k+1)}$ (solve using NUMG)
  $\beta_k = (\mathbf{z}^{(k+1)T}\mathbf{r}^{(k+1)})/(\mathbf{z}^{(k)T}\mathbf{r}^{(k)})$
  $\mathbf{p}^{(k+1)} = \mathbf{z}^{(k+1)} + \beta_k \mathbf{p}^{(k)}$
end for

7.3 Results for the Simplified Model Problem

We firstly test that the transformations do indeed work, i.e. that $\mathbf{I} = \mathbf{TU}$. We do this by using initial predetermined values for the control variable increments ψ_b', p_u' and χ'. We apply the full cycle of transformations, i.e. the **U**-transform followed by the **T**-transform, and then measure the resulting error between the initial and final values of the control variable increments. The values chosen for ψ_b', p_u' and χ' are:

$$\begin{aligned}
\psi_b' &= 0.5\cos(4\pi\phi)\sin(2\pi\lambda)\cos(\pi r)\,, \\
p_u' &= \cos(\pi\phi)\sin(4\pi\lambda)\sin(\pi r)\,, \\
\chi' &= 0.2\cos(\pi\lambda)\sin(2\pi\phi)\cos(\pi r)\,.
\end{aligned}$$

We perform the full cycle of transformations on different grid resolutions and determine the error with respect to the mesh width. Note that the mesh widths on the two grids in Figure 7-1 will always be equal.

We use the simplified model problem (7.2.1) to test the PV-based CVT, meaning that some other equations in the transformations must also be changed. These are:

(7.1.2) which becomes: $Q' = \alpha_0 \nabla_r^2 \psi' + \varepsilon_0 \dfrac{\partial^2 p'}{\partial r^2}\,,$ (7.3.1)

(7.1.4) which becomes: $\alpha_0 \nabla_r^2 \psi_b' + \varepsilon_0 \dfrac{\partial^2}{\partial r^2}\left(\nabla_r^{-2} \nabla_r \cdot f\rho_0 \nabla_r \psi_b'\right) = Q'\,,$ (7.3.2)

(7.1.6) which becomes: $$\alpha_0 \nabla_r^2 \psi'_u + \varepsilon_0 \frac{\partial^2}{\partial r^2} \left(\nabla_r^{-2} \nabla_r \cdot f \rho_0 \nabla_r \psi'_u \right) = \varepsilon_0 \frac{\partial^2 \xi'}{\partial r^2}, \quad (7.3.3)$$

(7.1.10) which becomes: $$\nabla_r^2 \psi'_u = (\alpha_0)^{-1} \left\{ -\varepsilon_0 \frac{\partial^2 p'_u}{\partial r^2} \right\}. \quad (7.3.4)$$

We use the preconditioned conjugate gradient (PCG) method preconditioned with an NUMG preconditioner at each iteration. Tables 7.1 and 7.2 show the results. In Table 7.1, $\text{error}_{\mathbf{X}}$ denotes the error between the initial values of control variable $\mathbf{X}$ and their values after the full cycle of transformations, measured using the Euclidean norm. We observe that the error is small and converges for all quantities relatively rapidly with respect to mesh width (at least quadratically for each variable). This fast convergence could be aided by the fact that smooth functions have been used for the control variables, which is likely to reduce the discretisation and interpolation errors. We will therefore test the full cycle of transformations in the next section with more oscillatory functions, which will simulate a more realistic representation of the true values of the control variables.

However, the results from this section confirm that the cycle of transformations is indeed accurate and that the accuracy can be improved using a finer resolution of grid points. Table 7.1 also shows the total number of PCG iterations required for solving (7.3.2) and (7.3.3) to a residual tolerance of 10^{-4}, and also the CPU time required per PCG solve. The number of iterations fluctuate slightly with problem size, but certainly do not show a trend of increasing with the problem size. Hence the total CPU time taken grows approximately linearly with problem size, and this optimality is even more apparent when measuring the CPU time per iteration on each problem size.

Problem Size	$\text{error}_{\psi'_b}$	$\text{error}_{p'_u}$	$\text{error}_{\chi'}$	# PCG its	Time taken for PCG solve
32x16x8	9.59E-2	0.11	6.68E-2	26/26	1.12/1.13
64x32x16	1.46E-2	1.60E-2	1.05E-2	39/45	14.6/15.6
128x64x32	3.34E-3	3.46E-3	1.03E-3	22/38	69.7/123.5
256x128x64	6.83E-4	5.11E-4	8.97E-6	16/29	463/865

Table 7.1: The magnitude of errors in the PV-based CVT. Column 5 denotes the number of PCG iterations required for solving (7.3.2) and (7.3.3), respectively. Similarly column 6 denotes the CPU time (in seconds) required for solving (7.3.2) and (7.3.3), respectively.

Table 7.2 gives the details of each of the components that is used in the process of solving the balanced PV-equation (7.3.2) using the PCG method. Since this equation and equation (7.3.3) are the main bottlenecks in the PV-based CVT, it is important that each component of these solves performs effectively. Column 2 shows the average number of iterations required for the 2D Poisson solve on each layer, solved using the

2D NUMG method from Section 5.5, which is clearly unaffected by the problem size. Likewise, Column 3 gives the average CPU time of the Poisson solves, which grows linearly with problem size. Column 4 gives the average CPU time for applying the operator $\mathcal{A}$ at each PCG iteration, which replaces the matrix-vector multiplication. Once again the CPU time grows linearly with problem size. Column 5 indicates the average number of V-cycle iterations required for the preconditioning step at each PCG iteration. Recall that the preconditioning system is solved using the 3D NUMG method to a residual tolerance of 10^{-4}, and stricter tolerances did not reduce the overall error of the full cycle of transformations any further. The number of V-cycle iterations required is robust with respect to the problem size, which is consistent with the experiments and theory from Chapter 5, and the time of the V-cycle solve grows linearly with problem size, as shown in Column 6. Hence we observe that all the components of the PCG method perform optimally, and as a result the PCG method performs almost optimally, with only a small fluctuation in the number of PCG iterations, as seen in Table 7.1.

Problem Size	# Its (2D)	2D solve	Apply $\mathcal{A}$	# Its (3D)	3D solve
32x16x8	8	2.6E-3	2.4E-2	3	1.8E-2
64x32x16	8	1.0E-2	0.17	4	0.18
128x64x32	9	3.9E-2	1.35	4	1.82
256x128x64	9	0.18	12.9	4	15.4

Table 7.2: Details of each component when solving the 3D balanced equation (7.3.2). CPU time in seconds. Identical results are found when solving the 3D unbalanced equation (7.3.3).

The methods just described are completely novel for these kind of problems and the results will be of great interest to the Met Office who are currently unable to carry out the PV-based CVT operationally due to the lack of a good enough solver.

7.4 Results for the Full Problem

This chapter is concluded with the results for the PV-based CVT as described in Section 7.1, using the balanced PV-equation (7.1.4) and the unbalanced anti-PV equation (7.1.6). No simplifications are made, and so the results here describe the transformations as they would be implemented at the Met Office.

We present the results in the same way as in Section 7.3, i.e. show whether the PV-based CVT yields accurate results, and then give the performance of the preconditioned Krylov subspace method on each of the 3D problems. The difference between the two 3D problems in this section and those in Section 7.3 is the inclusion of the first

and zeroth order terms. In addition, each of the coefficients α_0, β_0, γ_0 and ε_0 are now functions of radius *and latitude*, e.g. $\alpha_0 = \alpha_0(r, \phi)$, which imposes an additional difficulty. However, each of the functions varies smoothly in both directions. The operator for the full 3D problems are

$$\mathcal{A} = \alpha_0 \nabla_r^2 + \beta_0 \left(\nabla_r^{-2} \nabla_r \cdot f\rho_0 \nabla_r \right) + \gamma_0 \frac{\partial}{\partial r} \left(\nabla_r^{-2} \nabla_r \cdot f\rho_0 \nabla_r \right) + \varepsilon_0 \frac{\partial^2}{\partial r^2} \left(\nabla_r^{-2} \nabla_r \cdot f\rho_0 \nabla_r \right) ,$$

which is applied at each iteration.

Recall from Section 3.2.1 that the finite volume discretisation of first order terms yields a non-symmetric matrix. Hence the preconditioned conjugate gradient (PCG) method will be replaced with the preconditioned stabilized biconjugate gradient (Bi-CGSTAB) method [76], a fast iterative method for the solution of non-symmetric linear systems (note that GMRES [65] is also a suitably fast method for this situation). The preconditioning matrix, P, is a finite volume discretisation of

$$\mathcal{P} = \widetilde{\alpha_0} \nabla_r^2 + \widetilde{\beta_0} f\rho_0 + \widetilde{\gamma_0} f\rho_0 \frac{\partial}{\partial r} + \widetilde{\varepsilon_0} f\rho_0 \frac{\partial^2}{\partial r^2} ,$$

where $\widetilde{\alpha_0}$, $\widetilde{\beta_0}$, $\widetilde{\gamma_0}$ and $\widetilde{\varepsilon_0}$ are functions of r only and take the values of α_0, β_0, γ_0 and ε_0 at $\phi = \pi/4$, i.e. midway between the south pole and the equator. Solves involving the preconditioning matrix P will be done with NUMG using a tolerance of 10^{-4} as before. The inclusion of the zeroth order term ensures that the operator is positive definite, and a unique solution exists if the constant in the solution to the 2D solves is fixed. Note also that the first order term in the radial direction poses no additional difficulties for NUMG (as shown when solving the Helmholtz equation in Section 5.6.5) because of the r-line smoother we use.

The Bi-CGSTAB method requires *two* applications of the operator and *two* NUMG solves as opposed to one of each as in the case of the PCG method. The algorithm of the preconditioned Bi-CGSTAB method is given in Algorithm 7.3, as we use this to solve (7.1.4) and (7.1.6) to a residual tolerance of 10^{-4} as we did in the previous section.

The values chosen this time for ψ_b', p_u' and χ' are:

$$\begin{aligned}
\psi_b' &= 0.5 \cos(40\pi\phi) \sin(15\pi\lambda) \cos(8\pi r) + \cos(6\pi\phi) \sin(25\pi\lambda) \cos(12\pi r) , \\
p_u' &= 2 \cos(\pi\phi) \sin(4\pi\lambda) \sin(\pi r) - 0.1 \cos(30\pi\phi) \sin(40\pi\lambda) \sin(20\pi r) , \\
\chi' &= 3 \cos(20\pi\lambda) \sin(25\pi\phi) \cos(35\pi r) + \cos(5\pi\phi) \sin(10\pi\lambda) \sin(15\pi r) .
\end{aligned}$$

Algorithm 7.3 Bi-CGSTAB method with left preconditioning: `pcg(`$A, \mathtt{u}, \mathtt{b}$`)`

Choose $\mathbf{u}^{(0)}$ (initial solution) and P (preconditioner)
$\mathbf{t} = \mathtt{apply\_operator}(\mathcal{A}, \mathbf{u}^{(0)}, \mathbf{t})$ (apply $\mathcal{A}$ to $\mathbf{u}^{(0)}$)
$\mathbf{r}^{(0)} = \mathbf{b} - \mathbf{t}$
$\mathbf{p}^{(0)} = \mathbf{v}^{(0)} = \mathbf{0}$
$\hat{\mathbf{r}} = \mathbf{r}^{(0)}$
$\alpha = \omega = \rho_0 = 1$
for $k = 0, 1, \ldots,$ until convergence ...
  $\rho_{k+1} = \hat{\mathbf{r}}^T \mathbf{r}^{(k)}$
  $\beta = (\rho_{k+1}\alpha)/(\rho_k \omega)$
  $\mathbf{p}^{(k+1)} = \mathbf{r}^{(k)} + \beta(\mathbf{p}^{(k)} - \omega \mathbf{v}^{(k)})$
  Solve $\mathbf{y} = P^{-1}\mathbf{p}^{(k+1)}$ (solve using NUMG)
  $\mathbf{v}^{(k+1)} = \mathtt{apply\_operator}(\mathcal{A}, \mathbf{y}, \mathbf{v}^{(\mathtt{k+1})})$ (apply $\mathcal{A}$ to $\mathbf{y}$)
  $\alpha = \rho_{k+1}/(\hat{\mathbf{r}}^T \mathbf{v}^{(k+1)})$
  $\mathbf{s} = \mathbf{r}^{(k)} - \alpha \mathbf{v}^{(k+1)}$
  Solve $\mathbf{z} = P^{-1}\mathbf{s}$ (solve using NUMG)
  $\mathbf{t} = \mathtt{apply\_operator}(\mathcal{A}, \mathbf{z}, \mathbf{t})$ (apply $\mathcal{A}$ to $\mathbf{z}$)
  $\omega = (\mathbf{t}^T \mathbf{s})/(\mathbf{t}^T \mathbf{t})$
  $\mathbf{u}^{(k+1)} = \mathbf{u}^{(k)} + \alpha \mathbf{y} + \omega \mathbf{z}$
  $\mathbf{r}^{(k+1)} = \mathbf{s} - \omega \mathbf{t}$
  if $\mathbf{r}^{(k+1)}$ sufficiently small **exit**
end for

which have wavenumbers that are much higher than for the control variables used for the tests in Section 7.3, and so the functions are much harder to capture accurately, particularly with the coarse grids.

Table 7.3 shows that the CVT is implemented accurately, with the error between the initial and final values of each control variable converging between linearly and quadratically with respect to the mesh width. This is a more expected rate of convergence of the errors than those observed in Section 7.3, since the interpolation and discretisation operators used are second-order accurate with respect to mesh width. Note that there is a discrepancy in the number of iterations required for solving (7.1.4) and (7.1.6), caused by a different right-hand-side. We did not investigate the reason for the differences, but nevertheless it is clear that for both cases the number of Bi-CGSTAB iterations does not grow with problem size, hence the total CPU time grows roughly linearly with problem size. Table 7.4 demonstrates that each of the components in the preconditioned Krylov subspace method performes optimally, with the number of iterations independent of the problem size and the CPU times increasing linearly.

Problem Size	error$_{\psi'_b}$	error$_{p'_u}$	error$_{\chi'}$	# PCG its	Time taken for PCG solve
32x16x8	0.29	0.31	0.27	19/39	2.30/4.61
64x32x16	0.10	0.11	0.12	16/23	12.0/17.1
128x64x32	3.76E-2	4.77E-2	5.15E-2	6/15	48.3/119
256x128x64	1.55E-2	1.70E-2	1.84E-2	6/16	391/988

Table 7.3: The magnitude of errors in the PV-based CVT. Column 5 denotes the number of PCG iterations required for solving (7.1.4) and (7.1.6), respectively. Similarly column 6 denotes the CPU time (in seconds) required for solving (7.1.4) and (7.1.6), respectively.

Problem Size	# Its (2D)	2D solve	Apply $\mathcal{A}$	# Its (3D)	3D solve
32x16x8	8	2.9E-3	2.9E-2	4	2.9E-2
64x32x16	8	9.0E-3	0.16	4	0.21
128x64x32	9	3.8E-2	1.9	4	1.9
256x128x64	9	0.18	16.5	4	15.6

Table 7.4: Details of each component when solving the 3D balanced equation (7.1.4). The same results are found when solving the 3D unbalanced equation (7.1.6). CPU time in seconds.

Finally we demonstrate the performance of the method for solving the balanced PV-equation (7.1.4) within the developmental VAR code for the PV-based CVT. In addition to the coefficients α_0, β_0, γ_0 and ε_0 varying in two coordinate directions (latitude and radius), the mesh in the radial direction also varies in *all three* coordinate directions to take into account orography (relief of mountains) of the Earth's surface. (7.1.4) was solved in 38 iterations of the preconditioned Bi-CGSTAB method using the N108 data assimilation grid which has a resolution of $216 \times 163 \times 70$. Figure 7-2 is a snapshot of the balanced streamfunction plotted at ρ-level 40 which is approximately 29.8km above the surface of the Earth.

7.5 Summary

We have demonstrated in this chapter that NUMG can also be effectively used as a preconditioner to Krylov subspace methods in harder problems arising in the PV-based CVT. The new techniques outlined in the chapter successfully deal with the ill-conditioned nature of these problems and are able to solve them efficiently irrespective of the mesh resolution. Since the Met Office's GCR solver has not even been able to find a solution to these problems, the method described in this chapter is a big step towards getting the PV-based CVT operational in the Met Office's data assimilation code.

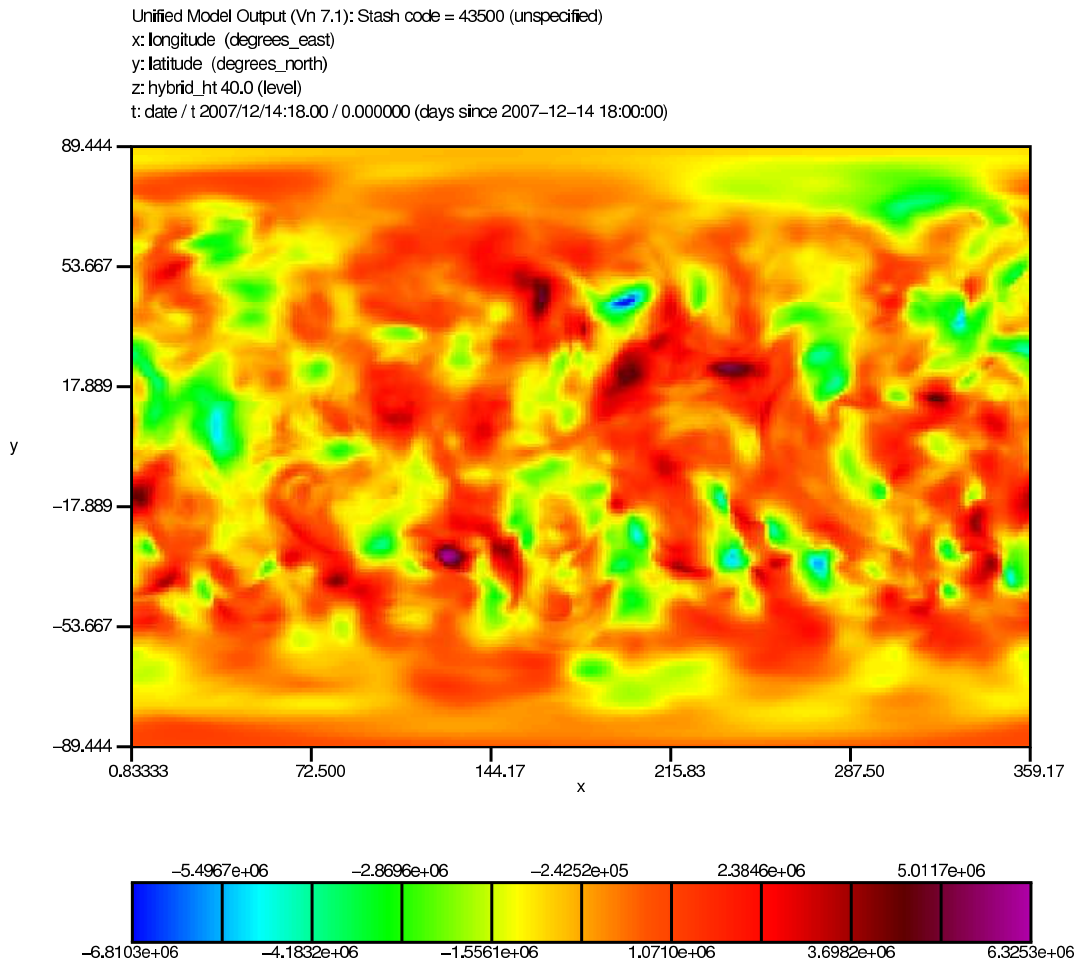

Figure 7-2: Snapshot of the balanced streamfunction, ψ_b, at ρ-level 40 (roughly 29.8km above ground level)

Chapter 8

Future Work and Extensions

The work of this thesis has answered many questions and has made a genuine contribution to the development of the VAR code at the Met Office. Nevertheless, there are several natural extensions to the thesis that ought to be investigated and would be of potential use to the Met Office. These are:

1. Solving the main elliptic problems on different grids, e.g. Yin–Yang or icosahedral grids [9]. These grids will avoid both the "pole problem" and the anisotropies introduced by the spherical polar grid used in this thesis and at the Met Office. Hence, the accuracy and speed of the multigrid method that can be achieved on these grids would be of great interest. Clearly the multigrid method will need to be modified for each particular grid.

2. Investigating multigrid methods for elliptic problems with a large convection term in all three coordinate directions. Solving equations with convection dominated flows will be useful from the point of view of the Met Office.

3. Solving the balanced equation as a coupled problem rather than a single nested problem. Techniques such as block elimination (cf. Section 5.5.2) could be used.

4. Investigating a theoretical justification for the robustness of the two-dimensional Poisson-type equation with grid-aligned anisotropy, when solved with the non-uniform multigrid method.

5. Investigation of multigrid methods for a more general class of problems, e.g. for matrices with positive off-diagonal entries. This could be linked with point 2, above.

Appendix A

Convergence Theory Using Fourier Analysis

Recall the one-dimensional model problem (4.1.1) from Section 4.1:

$$-u_{xx} = f(x) \quad \text{on} \quad \Omega = (0,1), \quad u(x) = 0 \quad \text{on} \quad \Gamma = \{0,1\}.$$

Let $n_\ell - 1$ and $n_{\ell-1} - 1$ be the number of interior grid points on levels ℓ (the fine grid) and $\ell - 1$ (the coarse grid) respectively. A finite difference discretisation of the above problem leads to a system of $n_\ell - 1$ equations denoted by

$$A_\ell \mathbf{u}_\ell = \mathbf{b}_\ell \,,$$

with

$$A_\ell = \frac{1}{h_\ell^2} \begin{bmatrix} 2 & -1 & & \\ -1 & 2 & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{bmatrix}.$$

We denote the mesh widths on each level by

$$h_\ell = \frac{1}{n_\ell}, \qquad h_{\ell-1} = \frac{1}{n_{\ell-1}} = 2h_\ell \,.$$

Recall from Section 4.2.3 that the eigenvectors $\{\mathbf{v}_\ell^{(i)} : 1 \le i \le n_\ell - 1\}$ of A_ℓ are

$$v_{j,\ell}^{(i)} = \sin(ij\pi h_\ell), \qquad i, j = 1, \ldots, n_\ell - 1$$

These eigenvectors form an orthonormal basis, so a matrix Q_ℓ consisting of $\mathbf{v}_\ell^{(i)}$ as its columns is unitary, i.e. $Q_\ell^T Q_\ell = I$. We form Q_ℓ with a special ordering of the eigenvectors (the reason will become apparent later):

$$Q_\ell = [\mathbf{v}_\ell^{(1)}, \mathbf{v}_\ell^{(n_\ell-1)}, \mathbf{v}_\ell^{(2)}, \mathbf{v}_\ell^{(n_\ell-2)}, \ldots, \mathbf{v}_\ell^{(\frac{n_\ell}{2}-1)}, \mathbf{v}_\ell^{(\frac{n_\ell}{2}+1)}, \mathbf{v}_\ell^{(\frac{n_\ell}{2})}] \,.$$

Multiplication by Q_ℓ or Q_ℓ^T does not change the spectral norm or spectral radius of a matrix, thus we have

$$\|\hat{M}_\ell\|_2 = \|M_\ell\|_2 \qquad \text{and} \qquad \rho(\hat{M}_\ell) = \rho(M_\ell) \,, \tag{A.0.1}$$

where M_ℓ is the iteration matrix for the two-grid method and $\hat{M}_\ell = Q_\ell^{-1} M_\ell Q_\ell$. Now, convergence of the two-grid method is obtained if $\rho(M_\ell) < 1$, but since the spectral radius and spectral norm of a symmetric matrix are equivalent, and by (A.0.1), we have that convergence is obtained if

$$\|\hat{M}_\ell\|_2 < 1 \,.$$

It will be apparent that the Fourier-transformed iteration matrix $\hat{M}_\ell$ of the two-grid method is a block diagonal matrix of the form

$$\hat{M}_\ell = blockdiag\{M_1, M_2, \ldots, M_{n_{\ell-1}-1}, M_{n_{\ell-1}}\} \,,$$

where $M_i \in \mathbb{R}^{2\times 2}$ for $1 \le i \le n_{\ell-1} - 1$ and $M_{n_{\ell-1}} \in \mathbb{R}$. The matrices M_i satisfy

$$\begin{cases} \|\hat{M}_\ell\|_2 = max\{\|M_i\|_2 \ : \ 1 \le i \le n_{\ell-1}\} \\ \rho(\hat{M}_\ell) = max\{\rho(M_i) \ : \ 1 \le i \le n_{\ell-1}\} \end{cases} ,$$

therefore it is sufficient to show that

$$max\{\|M_i\|_2 \ : \ 1 \le i \le n_{\ell-1}\} < 1$$

for convergence. The Fourier transformed two-grid iteration matrix is

$$\begin{aligned} \hat{M}_\ell &= Q_\ell^T M_\ell Q_\ell = Q_\ell^T (I - P A_{\ell-1}^{-1} R A_\ell) S_\ell^\nu Q_\ell \\ &= (I - \hat{P} \hat{A}_{\ell-1}^{-1} \hat{R} \hat{A}_\ell) \hat{S}_\ell^\nu \,, \end{aligned}$$

where

$$\hat{A}_\ell = Q_\ell^T A_\ell Q_\ell, \quad \hat{S}_\ell = Q_\ell^T S_\ell Q_\ell, \quad \hat{P} = Q_{\ell-1}^T P Q_\ell, \quad \hat{P} = Q_\ell^T P Q_{\ell-1} \,.$$

Note that we are only using pre-smoothing for simplicity, but the analysis is still valid when post-smoothing is included. The Fourier transformation matrix $Q_{\ell-1}$ is built using the eigenvectors of $A_{\ell-1}$, i.e. $\mathbf{v}^{(i)}_{j,\ell-1} = \sin(2ij\pi h_\ell)$, $i = 1, \ldots, n_{\ell-1} - 1$, thus

$$Q_{\ell-1} = [\mathbf{v}^{(1)}_{\ell-1}, \mathbf{v}^{(2)}_{\ell-1}, \ldots, \mathbf{v}^{(n_{\ell-1}-1)}_{\ell}] .$$

From Section 4.2.3, we know that the eigenvalues of A_ℓ are

$$\lambda_i = \frac{4}{h_\ell^2} \sin^2 \left(i\pi \frac{h_\ell}{2} \right) .$$

For convenience we introduce the notation

$$s_i = \sin \left(i\pi \frac{h_\ell}{2} \right) , \qquad c_i = \cos \left(i\pi \frac{h_\ell}{2} \right) .$$

Noting that $s^2_{n_\ell - i} = c_i^2$ and that $A_\ell \mathbf{v}^{(i)}_\ell = \lambda_i \mathbf{v}^{(i)}_\ell$ for $i = 1, \ldots, n_\ell - 1$, we have

$$\hat{A}_\ell = Q_\ell^T A_\ell Q_\ell = blockdiag\{A_1, \ldots, A_{n_{\ell-1}}\} , \tag{A.0.2}$$

where

$$A_i = 4h_\ell^{-2} \begin{bmatrix} s_i^2 & 0 \\ 0 & c_i^2 \end{bmatrix} , i = 1, \ldots, n_{\ell-1} - 1 \qquad \text{and} \qquad A_{n_{\ell-1}} = 2h_\ell^{-2} . \tag{A.0.3}$$

For the smother, suppose we choose the Jacobi method damped by $\omega = \frac{1}{2}$, i.e. $S_\ell = I - \frac{1}{4} h_\ell^2 A_\ell$. Then noting that $s_i^2 = 1 - c_i^2$, we use (A.0.2) and (A.0.3) to deduce that

$$\hat{S}_\ell = Q_\ell^T S_\ell Q_\ell = blockdiag\{S_1, \ldots, S_{n_{\ell-1}}\} , \tag{A.0.4}$$

where

$$S_i = \begin{bmatrix} c_i^2 & 0 \\ 0 & s_i^2 \end{bmatrix} , i = 1, \ldots, n_{\ell-1} - 1 \qquad S_{n_{\ell-1}} = \frac{1}{2}. \tag{A.0.5}$$

For the coarse grid operator, we have

$$\hat{A}_{\ell-1} = Q^T_{\ell-1} A_{\ell-1} Q_{\ell-1} = diag\{A'_1, \ldots, A'_{n_{\ell-1}}\} \quad \text{where} \quad A'_i = 4h_\ell^{-2} s_i^2 c_i^2 , \tag{A.0.6}$$

because of $A_{\ell-1} \mathbf{v}^{(i)}_{\ell-1} = \lambda'_i \mathbf{v}^{(i)}_{\ell-1}$ with $\lambda'_i = 4h_{\ell-1}^{-2} \sin^2 \left(i\pi \frac{h_{\ell-1}}{2} \right) = h_\ell^{-2} \sin^2(i\pi h_\ell)$. Finally for the transfer operators we obtain

$$\hat{R} = Q^T_{\ell-1} R Q_\ell = blockdiag\{R_1, \ldots, R_{n_{\ell-1}-1}, 0\} \quad \text{where} \quad R_i = \sqrt{\frac{1}{2}} [c_i^2, s_i^2] , \tag{A.0.7}$$

and

$$\hat{P} = Q_\ell^T R Q_{\ell-1} = blockdiag\{P_1, \ldots, P_{n_{\ell-1}-1}, 0\} \quad \text{with} \quad P_i = \sqrt{2} \begin{bmatrix} c_i^2 \\ s_i^2 \end{bmatrix}. \tag{A.0.8}$$

Now, using (A.0.2) – (A.0.8), we obtain

$$M_i = (I - P_i(A_i')^{-1} R_i A_i) S_i^\nu, \qquad i = 1, \ldots, n_{\ell-1} - 1 \qquad \text{and} \qquad M_{n_{\ell-1}} = 2^{-\nu}.$$

Inserting the matrices, we obtain for $i = 1, \ldots, n_{\ell-1} - 1$,

$$\begin{aligned} M_i &= \left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} c_i^2 \\ s_i^2 \end{bmatrix} \frac{h^2}{4c_i^2 s_i^2} [c_i^2, s_i^2] \begin{bmatrix} s_i^2 & 0 \\ 0 & c_i^2 \end{bmatrix} \right) \begin{bmatrix} c_i^2 & 0 \\ 0 & s_i^2 \end{bmatrix}^\nu \\ &= \begin{bmatrix} s_i^2 & c_i^2 \\ s_i^2 & c_i^2 \end{bmatrix} \begin{bmatrix} c_i^2 & 0 \\ 0 & s_i^2 \end{bmatrix}^\nu . \end{aligned} \tag{A.0.9}$$

The first matrix represents the coarse grid correction and the second represents the smoother. Now since $1 \le i \le n_{\ell-1} - 1 \le n_\ell - i \le n_\ell$, we have

$$0 < s_i^2 < \frac{1}{2} < c_i^2 < 1\,.$$

Hence in the coarse grid correction, the component of the smooth eigenvector (i.e. the first column containing the s_i terms) is better reduced, while the smoother better reduces the component of the oscillatory eigenvector (the second column). Now, using (A.0.9),

$$\begin{aligned} \|M_i\|_2 &= \sqrt{2[s_i^4(1 - s_i^2)^{2\nu} + (1 - s_i^2)^2 s_k^{4\nu}]} \\ \Rightarrow max\{\|M_i\|_2\} &= max\{f_\nu(s_i^2) : i = 1, \ldots, n_{\ell-1}\}\,, \end{aligned}$$

where

$$f_\nu(\xi) = \sqrt{2[\xi^2(1 - \xi)^{2\nu} + (1 - \xi)^2 \xi^{2\nu}]}\,.$$

Since $0 < s_i^2 < \frac{1}{2}$, we have

$$\|M\|_2 = \|\hat{M}\|_2 = max\{\|M_k\|_2\} \le max_{0 \le \xi \le \frac{1}{2}} |f_\nu(\xi)|\,.$$

Hence the convergence rate of the two-grid iteration depends only on the number of smoothing steps ν and is independent of h_ℓ.

Appendix B

The Multigrid Iteration Matrix

The proof of Lemma 4.6.9 is as follows:

Proof. For $\ell = 1$, there is no multigrid iteration and only the coarse grid solve. For $\ell = 2$, the multigrid algorithm is identical to the two-grid algorithm, since only two grids are used. Let $\ell > 2$, and $\mathbf{u}_\ell^j$ be the approximation to the solution at the j^{th} iteration of multigrid. Then the pre-smoothing step yields

$$\bar{\mathbf{u}}_\ell = S_\ell^{\nu_1} \mathbf{u}_\ell^j \,.$$

Then we apply the multigrid iteration at level $\ell-1$ to $A_{\ell-1}\mathbf{e}_{\ell-1} = \mathbf{d}_{\ell-1} := R(\mathbf{b}_\ell - A_\ell \bar{\mathbf{u}}_\ell)$ with starting guess $\mathbf{e}_{\ell-1}^0 = 0$:

$$\begin{aligned} \mathbf{e}_{\ell-1}^1 &= M_{\ell-1}^{MG}\mathbf{e}_{\ell-1}^0 + N_{\ell-1}^{MG}\mathbf{d}_{\ell-1} = N_{\ell-1}^{MG}\mathbf{d}_{\ell-1} \\ \Rightarrow\ \mathbf{e}_{\ell-1}^\gamma &= \sum_{\mu=0}^{\gamma-1} (M_{\ell-1}^{MG})^\mu N_{\ell-1}^{MG}\mathbf{d}_{\ell-1} \,. \end{aligned}$$

Interpolating the error onto the fine grid, we have

$$\mathbf{e}_\ell = P\mathbf{e}_{\ell-1}^\gamma = P\sum_{\mu=0}^{\gamma-1} (M_{\ell-1}^{MG})^\mu N_{\ell-1}^{MG}\mathbf{d}_{\ell-1},$$

and after updating the error and post-smoothing, we obtain

$$S_\ell^{\nu_2}\mathbf{u}_\ell^{j+1} = S_\ell^{\nu_2}(\bar{\mathbf{u}}_\ell + \mathbf{e}_\ell) = S_\ell^{\nu_2}S_\ell^{\nu_1}\mathbf{u}_\ell^j + P\sum_{\mu=0}^{\gamma-1} (M_{\ell-1}^{MG})^\mu N_{\ell-1}^{MG} R(\mathbf{b}_\ell - A_\ell S_\ell^{\nu_1}\mathbf{u}_\ell^j) \,.$$

Now, any solution of $A_\ell \mathbf{u}_\ell = \mathbf{b}_\ell$ is a stationary point of (4.6.32), thus

$$\begin{aligned}
&\mathbf{u}_\ell = M_\ell^{MG}\mathbf{u}_\ell + N_\ell^{MG}\mathbf{b}_\ell = M_\ell^{MG}\mathbf{u}_\ell + N_\ell^{MG}A_\ell\mathbf{u}_\ell \\
\Rightarrow &I = M_\ell^{MG} + N_\ell^{MG}A_\ell \\
\Rightarrow &N_\ell^{MG} = (I - M_\ell^{MG})A_\ell^{-1}\,.
\end{aligned}$$

Using this expression for N_ℓ^{MG} we obtain

$$\begin{aligned}
&P\sum_{\mu=0}^{\gamma-1}(M_{\ell-1}^{MG})^\mu N_{\ell-1}^{MG} R(\mathbf{b}_\ell - A_\ell S_\ell^{\nu_1}\mathbf{u}_\ell^j) \\
&\qquad = P\sum_{\mu=0}^{\gamma-1}(M_{\ell-1}^{MG})^\mu (I - M_{\ell-1}^{MG})A_{\ell-1}^{-1} R(\mathbf{b}_\ell - A_\ell S_\ell^{\nu_1}\mathbf{u}_\ell^j) \\
&\qquad = P\sum_{\mu=0}^{\gamma-1}[(M_{\ell-1}^{MG})^\mu - (M_{\ell-1}^{MG})^{\mu+1}]A_{\ell-1}^{-1} R(\mathbf{b}_\ell - A_\ell S_\ell^{\nu_1}\mathbf{u}_\ell^j) \\
&\qquad = P[I - (M_{\ell-1}^{MG})^\gamma]A_{\ell-1}^{-1} R(\mathbf{b}_\ell - A_\ell S_\ell^{\nu_1}\mathbf{u}_\ell^j)\,.
\end{aligned}$$

Inserting this into (B) gives

$$S_\ell^{\nu_2}\mathbf{u}_\ell^{j+1} = S_\ell^{\nu_2}(\bar{\mathbf{u}}_\ell + \mathbf{e}_\ell) = S_\ell^{\nu_2}S_\ell^{\nu_1}\mathbf{u}_\ell^j + P[I - (M_{\ell-1}^{MG})^\gamma]A_{\ell-1}^{-1} R(\mathbf{b}_\ell - A_\ell S_\ell^{\nu_1}\mathbf{u}_\ell^j),$$

and so the iteration matrix is

$$\begin{aligned}
M_\ell^{MG} &= S_\ell^{\nu_2}(I - P[I - (M_{\ell-1}^{MG})^\gamma]A_{\ell-1}^{-1}RA_\ell)S_\ell^{\nu_1} \\
&= S_\ell^{\nu_2}(I - PA_{\ell-1}^{-1}RA_\ell)S_\ell^{\nu_1} + S_\ell^{\nu_2}P(M_{\ell-1}^{MG})^\gamma A_{\ell-1}^{-1}RA_\ell S_\ell^{\nu_1} \\
&= M_\ell^{TG} + S_\ell^{\nu_2}P(M_{\ell-1}^{MG})^\gamma A_{\ell-1}^{-1}RA_\ell S_\ell^{\nu_1}\,,
\end{aligned}$$

by induction. □

Appendix C

Proof of the Approximation Property

Proof. (of Theorem 4.6.3) The proof will be given in a finite element setting. We consider the finite element discretisation of the 2D Poisson equation as it satisfies the regularity assumption (4.6.5). Let us first recall the problem to be solved (in the weak formulation):

$$\text{Find } u \in V \text{ s.t. } a(u,v) = (f,v)_{L_2(\Omega)} \quad \forall v \in V\,,$$

where $a : V \times V \to \mathbb{R}$ is a bilinear form, $(\cdot,\cdot)_{L_2(\Omega)}$ is the scalar product of $L_2(\Omega)$ and $V = H_0^1(\Omega)$. For the Poisson problem, the weak form is

$$a(u,v) = \int_\Omega \nabla u \cdot \nabla v \; dxdy\,, \qquad (f,v)_{L_2(\Omega)} = \int_\Omega f v dx\,.$$

The weak form may be recast as the problem of finding $u \in V$ satisfying the equation

$$\mathcal{A}u = f\,. \tag{C.0.1}$$

where $\mathcal{A} : V \to V'$ is a mapping from the Hilbert space V to a dual space V' of all bounded linear functionals on V.

We introduce a hierarchy of piecewise bilinear finite dimensional spaces

$$V_1 \subset V_2 \subset \cdots \subset V_{F-1} \subset V_F \subset V\,.$$

with suitable piecewise bilinear nodal basis functions for each of the spaces. We want to solve the approximate weak form in the space V_F, so we seek $u_F \in V_F$ s.t.

$$a(u_F, v_F) = (f, v_F)_{L_2(\Omega)} \quad \forall v_F \in V_F\,. \tag{C.0.2}$$

Then $\mathbf{u}_F \in \mathbb{R}^{n^F}$ is the vector containing the values of u_F at the nodal points, and we have $dim(V_\ell) = dim(\mathbb{R}^{n^\ell}) = n^\ell$, $\forall \ell = 1, \ldots, F$.

The finite element discretisation of (C.0.2) on level ℓ is:

$$A_\ell \mathbf{u}_\ell = \mathbf{b}_\ell \, .$$

We now define a bijective mapping from $\mathbb{R}^{n_\ell}$ to V_ℓ using the operator p_ℓ:

$$u_\ell = p_\ell \mathbf{u}_\ell \, ,$$

with the adjoint mapping from V_ℓ to $\mathbb{R}^{n_\ell}$ defined as $r_\ell = p_\ell^T$ such that

$$< r_\ell u_\ell, \mathbf{u}_\ell >= (u_\ell, p_\ell \mathbf{u}_\ell)_{L_2(\Omega)} \, ,$$

where $\mathbf{u}_\ell \in \mathbb{R}^{n_\ell}$, $u_\ell \in V_\ell$ and where $< \cdot, \cdot >$ is the scalar product defined as

$$< \mathbf{a}, \mathbf{b} >= h_\ell^2 \sum_{i=1}^{n^\ell} a(i) b(i) \, .$$

The scaling by h_ℓ^2 in the definition of $< \cdot, \cdot >$ is applied to ensure that the Euclidean norm of a vector $\mathbf{v}_\ell \in \mathbb{R}^{n_\ell}$ and the $L^2(\Omega)$-norm of the corresponding function $p_\ell \mathbf{v}_\ell \in V_\ell$ are uniformly equivalent, i.e.

$$\frac{1}{C} \parallel \mathbf{v}_\ell \parallel_2 \leq \parallel p_\ell \mathbf{v}_\ell \parallel_{L^2(\Omega)} \leq C \parallel \mathbf{v}_\ell \parallel_2 \, . \tag{C.0.3}$$

Using the bijective mapping, we define the canonical interpolation operator $P_\ell : \mathbb{R}^{n_{\ell-1}} \rightarrow \mathbb{R}^{n_\ell}$, by

$$p_{\ell-1} = p_\ell P_\ell \quad \Rightarrow \quad P_\ell = p_\ell^{-1} p_{\ell-1} \, .$$

The canonical restriction operator, $R_\ell : \mathbb{R}^{n_\ell} \rightarrow \mathbb{R}^{n_{\ell-1}}$, is defined as the adjoint of P_ℓ:

$$R_\ell = P_\ell^T \quad \text{s.t.} \quad < P_\ell \mathbf{u}_{\ell-1}, \mathbf{w}_\ell >=< \mathbf{u}_{\ell-1}, R_\ell \mathbf{w}_\ell > \, .$$

The coarse grid operator $A_{\ell-1}$ is then constructed using the Galerkin product (see Section 4.4.2), i.e. $A_{\ell-1} = R_\ell A_\ell P_\ell$.

Recall the regularity assumption from Finite Element theory [19] (see also (4.6.5)):

$$\parallel u_\ell - u \parallel_{L_2(\Omega)} \leq C h_\ell^2 \parallel f \parallel_{L_2(\Omega)} \, . \tag{C.0.4}$$

Now, (C.0.1) implies $u = \mathcal{A}^{-1} f$. Also, it can be shown that $A_\ell = r_\ell \mathcal{A} p_\ell$ and $\mathbf{b}_\ell = r_\ell f$

which gives $u_\ell = p_\ell A_\ell^{-1} r_\ell f$. Inserting the values for u and u_ℓ into (C.0.4), we get

$$\begin{aligned} &\| p_\ell A_\ell^{-1} r_\ell f - \mathcal{A}^{-1} f \|_{L_2(\Omega)} \leq C h_\ell^2 \| f \|_{L_2(\Omega)} \\ \Rightarrow &\| p_\ell A_\ell^{-1} r_\ell - \mathcal{A}^{-1} \|_{L_2(\Omega) \to L_2(\Omega)} \leq C h_\ell^2 \,. \end{aligned} \tag{C.0.5}$$

Similarly, on $V_{\ell-1}$ we have:

$$\| p_{\ell-1} A_{\ell-1}^{-1} r_{\ell-1} - \mathcal{A}^{-1} \|_{L_2(\Omega) \to L_2(\Omega)} \leq C h_{\ell-1}^2 \,. \tag{C.0.6}$$

So combining (C.0.5) and (C.0.6) gives

$$\begin{aligned} &\| p_\ell A_\ell^{-1} r_\ell - p_{\ell-1} A_{\ell-1}^{-1} r_{\ell-1} \|_{L_2(\Omega) \to L_2(\Omega)} \\ &\qquad \leq \| p_\ell A_\ell^{-1} r_\ell - \mathcal{A}^{-1} \|_{L_2(\Omega) \to L_2(\Omega)} + \| p_{\ell-1} A_{\ell-1}^{-1} r_{\ell-1} - \mathcal{A}^{-1} \|_{L_2(\Omega) \to L_2(\Omega)} \\ &\qquad \leq C(h_\ell^2 + h_{\ell-1}^2) \,. \end{aligned}$$

Then using $p_{\ell-1} = p_\ell P_\ell$ and $r_{\ell-1} = R_\ell r_\ell$, we obtain

$$\| p_\ell (A_\ell^{-1} - P A_{\ell-1}^{-1} R) r_\ell \|_{L_2(\Omega) \to L_2(\Omega)} \leq C(h_\ell^2 + h_{\ell-1}^2) \,.$$

Now inequality (C.0.3) implies $\| p_\ell X r_\ell \|_{L_2(\Omega) \to L_2(\Omega)} \leq C^2 \| X \|_2$, where $\| \cdot \|_2$ now denotes the spectral norm, so

$$\begin{aligned} \| A_\ell^{-1} - P A_{\ell-1}^{-1} R \|_2 &\leq \frac{1}{C^2} \| p_\ell (A_\ell^{-1} - P A_{\ell-1}^{-1} R) r_\ell \|_{L_2(\Omega) \to L_2(\Omega)} \\ &\leq C'(h_\ell^2 + h_{\ell-1}^2) \leq C_A h_\ell^2 \end{aligned}$$

under the assumption that $h_{\ell-1} = 2h_\ell$. This proves the approximation property. □

Bibliography

[1] A. Arakawa and C.S. Konor. Vertical differencing of the primitive equations based on the Charney-Phillips grid in hybrid σ-p vertical coordinates. *Monthly weather review*, 124(3):511–528, 1996.

[2] A. Arakawa and V.R. Lamb. Computational design of the basic dynamical processes of the UCLA general circulation model. *General circulation models of the atmosphere.(A 78-10662 01-47) New York, Academic Press, Inc.*, pages 173–265, 1977.

[3] C.F. Baillie, J.C. McWilliams, J.B. Weiss, and I. Yavneh. Implementation and performance of a grand challenge 3D quasi-geostrophic multi-grid code on the Cray T3D and IBM SP2. In *Proceedings of Supercomputing95*. Citeseer, 1995.

[4] R.N. Bannister and M.J.P. Cullen. New PV-based Control Variables for Met Office VAR. 2009.

[5] R.N. Bannister, M.J.P. Cullen, and M. Wlasak. A regime-dependant balanced control variable based on potential vorticity for variational data assimilation. *Q. J. R. Meteorol. Soc.*, Submitted.

[6] R.N. Bannister, D. Katz, M.J.P. Cullen, A.S. Lawless, and N.K. Nichols. Modelling of forecast errors in geophysical fluid flows. *Dynamics*, 2007.

[7] S.R.M. Barros. Multigrid methods for two-and three-dimensional Poisson-type equations on the sphere. *Journal of Computational Physics*, 92(2):313–348, 1991.

[8] G.K. Batchelor. *An introduction to fluid dynamics*. Cambridge Univ Pr, 2000.

[9] J.R. Baumgardner and P.O. Frederickson. Icosahedral discretization of the two-sphere. *SIAM J. Numer. Anal.*, 22(6):1107–1115, 1985.

[10] S. Beuchler. A preconditioner for solving the inner problem of the p-version of the FEM. *time: 5/2000 Please send comments concerning this metadata document to wwwadm@ mathematik. tu-chemnitz. de last update: 5. September 2000*, 2000.

[11] G. Birkhoff, R.S. Varga, and D. Young. Alternating direction implicit methods. *Advances in computers*, 3:189–273, 1962.

[12] H.B. Bluestein. Principles of Kinematics and Dynamics. Vol. I. Synoptic–Dynamic Meteorology in Midlatitudes, 1992.

[13] S. Börm. Tensor product multigrid for Maxwell's equation with aligned anisotropy. *Computing*, 66(4):321–342, 2001.

[14] S. Börm and R. Hiptmair. Analysis of tensor product multigrid. *Numerical Algorithms*, 26(3):219–234, 2001.

[15] DE Bourne and PC Kendall. *Vector analysis and Cartesian tensors*. CRC, 1992.

[16] D. Braess and W. Hackbusch. A new convergence proof for the multigrid method including the V-cycle. *SIAM Journal on Numerical Analysis*, 20(5):967–975, 1983.

[17] J.H. Bramble and X. Zhang. Uniform convergence of the multigrid V-cycle for an anisotropic problem. *Mathematics of computation*, 70(234):453–470, 2001.

[18] A. Brandt, S.F. McCormick, and J. Ruge. Algebraic multigrid (AMG) for sparse matrix equations. *Sparsity and its Applications*, pages 257–284, 1984.

[19] S.C. Brenner and L.R. Scott. *The mathematical theory of finite element methods*. Springer Verlag, 2008.

[20] W.L. Briggs and S.F. McCormick. *A multigrid tutorial*. Society for Industrial Mathematics, 2000.

[21] E.O. Brigham. *The fast Fourier transform and its applications*. Prentice Hall Englewood Cliffs, NJ, 1988.

[22] D.E. Brown, N.K. Nichols, and M.J. Bell. Preconditioners for inhomogeneous anisotropic problems in spherical geometry. *International Journal for Numerical Methods in Fluids*, 47(10-11):1213–1219, 2005.

[23] P.G. Ciarlet. *The finite element method for elliptic problems*. North-Holland, 1978.

[24] M.J.P. Cullen. A test of a semi-implicit integration technique for a fully compressible non-hydrostatic model. *Royal Meteorological Society, Quarterly Journal*, 116:1253–1258, 1990.

[25] M.J.P. Cullen. Four-dimensional variational data assimilation: A new formulation of the background-error covariance matrix, based on a potential-vorticity representation. *Quarterly Journal of the Royal Meteorological Society*, 129(593):2777–2796, 2003.

[26] M.J.P. Cullen. *A mathematical theory of large-scale atmosphere/ocean flow*. Imperial College Pr, 2006.

[27] A. Danilenko. Solution of the Neumann problem for Poisson's equation by the multigrid method in the three-dimensional case. *Computational mathematics and mathematical physics*, 31(10):74–80, 1991.

[28] T. Davies, MJP Cullen, AJ Malcolm, MH Mawson, A. Staniforth, AA White, and N. Wood. A new dynamical core for the Met Office's global and regional modelling of the atmosphere. *Quarterly Journal of the Royal Meteorological Society*, 131(608):1759–1782, 2005.

[29] T.A. Davis. *Direct methods for sparse linear systems*. Society for Industrial Mathematics, 2006.

[30] J. Derber and F. Bouttier. A reformation of the background error covariance in the ECMWF global data assimilation system. *Tellus*, 51A:195–221, 1999.

[31] M. Diamantakis, T. Davies, and N. Wood. An iterative time-stepping scheme for the Met Office's semi-implicit semi-Lagrangian non-hydrostatic model. *Quarterly Journal of the Royal Meteorological Society*, 133(625):997–1012, 2007.

[32] C.C. Douglas, G. Haase, and U. Langer. *A tutorial on elliptic PDE solvers and their parallelization*. Siam, 2003.

[33] I.S. Duff, A.M. Erisman, and J.K. Reid. *Direct methods for sparse matrices*. Oxford University Press, USA, 1989.

[34] S.C. Eisenstat, H.C. Elman, and M.H. Schultz. Variational iterative methods for nonsymmetric systems of linear equations. *SIAM Journal on Numerical Analysis*, 20(2):345–357, 1983.

[35] G. Evensen. *Data assimilation: The ensemble Kalman filter*. Springer Verlag, 2009.

[36] R. Eymard, T. Gallouët, and R. Herbin. Finite volume methods. *Handbook of numerical analysis*, 7:713–1018, 2000.

[37] R.D. Falgout and U.M. Yang. hypre: A Library of High Performance Preconditioners. *Lecture Notes in Computer Science*, 2331/2002:632–641, 2002.

[38] M. Fisher. Background error covariance modelling. In *Seminar on Recent Development in Data Assimilation for Atmosphere and Ocean*, pages 45–63, 2003.

[39] S.R. Fulton, P.E. Ciesielski, and W.H. Schubert. Multigrid methods for elliptic problems: A review. *Mon. Wea. Rev*, 114(5):943–959, 1986.

[40] M.W. Gee, J.J. Hu, and R.S. Tuminaro. A new smoothed aggregation multigrid method for anisotropic problems. *Numer. Lin. Alg. Appl*, 16:19–37, 2009.

[41] T. Gjesdal. A Cell-Centered Multigrid Algorithm for All Grid Sizes. In *NASA conference publication*, pages 327–338. Citeseer, 1996.

[42] W. Gropp, E. Lusk, and A. Skjellum. *Using MPI: portable parallel programming with the message-passing interface*. MIT press, 1999.

[43] W. Hackbusch. Multi-grid convergence theory. *Multigrid Methods*, pages 177–219, 1982.

[44] W. Hackbusch. *Multi-grid methods and applications*. Springer Verlag, 1985.

[45] W. Hackbusch. *Iterative solution of large sparse systems of equations*. Springer, 1994.

[46] V.E. Henson and U.M. Yang. BoomerAMG: a parallel algebraic multigrid solver and preconditioner. *Applied Numerical Mathematics*, 41(1):155–177, 2002.

[47] M.R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, (49):409–436, 1952.

[48] J.D. Hogg, J.K. Reid, and J.A. Scott. A DAG-based sparse Cholesky solver for multicore architectures. 2009.

[49] N.B. Ingleby. The statistical structure of forecast errors and its representation in the Met. Office global 3-D variational data assimilation scheme. *Quarterly Journal of the Royal Meteorological Society*, 127(571):209–232, 2001.

[50] A. Iserles. *A first course in the numerical analysis of differential equations*. Cambridge Univ Pr, 2008.

[51] M. Kameyama, A. Kageyama, and T. Sato. Multigrid-based simulation code for mantle convection in spherical shell using Yin-Yang grid. *Physics of the Earth and Planetary Interiors*, 171(1-4):19–32, 2008.

[52] D. Katz. *The application of PV-based control variable transformations in variational data assimilation.* PhD thesis, Reading University, Department of Mathematics, 2007.

[53] D. Katz, A.S. Lawless, N.K. Nichols, M.J.P. Cullen, and R.N. Bannister. A comparison of potential vorticity-based and vorticity-based control variables. *The University of Reading, Department of Mathematics, Numerical Analysis Report*, 8:1–34, 2006.

[54] M.C. Lai and W.C. Wang. Fast direct solvers for Poisson equation on 2D polar and spherical geometries. *Numerical Methods for Partial Differential Equations*, 18(1):56–68, 2002.

[55] J. Larsson, F.S. Lien, and E. Yee. Conditional semi-coarsening multigrid algorithm for the Poisson equation on anisotropic grids. *J. Comput. Phys.*, 208:368–383, 2005.

[56] AC Lorenc. Analysis methods for numerical weather prediction. *Quart. J. Roy. Meteor. Soc*, 112(11):77–1194, 1986.

[57] T Melvin, M Dubal, N Wood, A Staniforth, and M Zerroukat. An inherently mass conserving iterative semi-implicit semi-Lagrangian discretization of the non-hydroststic vertical-slice equations. *Quart. J. Roy. Meteor. Soc*, 136:799–814, 2010.

[58] N. Neuss. V-cycle convergence with unsymmetric smoothers and application to an anisotropic model problem. *SIAM Journal of Numerical Analysis*, 35(3):1201–1212, 1998.

[59] Y. Notay. Convergence analysis of perturbed two-grid and multigrid methods. *SIAM J. Numer. Anal.*, 45(3):1035–1044.

[60] P.S. Pacheco. *Parallel programming with MPI.* Morgan Kaufmann, 1997.

[61] F. Rawlins, S.P. Ballard, K.J. Bovis, A.M. Clayton, D. Li, G.W. Inverarity, A.C. Lorenc, and T.J. Payne. The Met Office global four-dimensional variational data assimilation scheme. *Quarterly Journal of the Royal Meteorological Society*, 133(623):347–362, 2007.

[62] J.W. Ruge, Y. Li, S. McCormick, A. Brandt, and J.R. Bates. A nonlinear multigrid solver for a semi-Lagrangian potential vorticity-based shallow-water model on the sphere. *SIAM Journal on Scientific Computing*, 21:2381, 2000.

[63] J.W. Ruge and K. Stüben. Algebraic multigrid. *Multigrid methods*, 3:73–130, 1987.

[64] Y. Saad. *Iterative methods for sparse linear systems*. Society for Industrial Mathematics, 2003.

[65] Y. Saad and M.H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 7(3):856–869, 1986.

[66] R. Scheichl. Parallel solution of the transient multigroup neutron diffusion equation wth multigrid and preconditioned Krylov subspace methods. Master's thesis, Johannes Kepler University Linz, Department of Mathematics, 1997.

[67] R. Scheichl. *Numerical Solution of PDEs: The Finite Element Method, Lecture Notes in Mathematical Sciences*. University of Bath, 2006.

[68] N. Schieweck. A multigrid convergence proof by a strengthened Cauchy inequality for symmetric elliptic boundary value problems. In *Second Multigrid Seminar: Garzau, Nov. 5-8, 1985*, page 49. Akademie der Wissenschaften der DDR, Karl-Weierstrass-Institut für Mathematik, 1986.

[69] W.C. Skamarock, P.K. Smolarkiewicz, and J.B. Klemp. Preconditioned conjugate-residual solvers for Helmholtz equations in nonhydrostatic models. *Monthly Weather Review*, 125(4):587–599, 1997.

[70] R. Stevenson. Robustness of multi-grid applied to anisotropic equations on convex domains and on domains with re-entrant corners. *Numerische Mathematik*, 66(1):373–398, 1993.

[71] K. Stüben. *Algebraic multigrid (AMG): An introduction with applications*. GMD-Forschungszentrum Informationstechnik, 1999.

[72] P.N. Swarztrauber. The direct solution of the discrete Poisson equation on the surface of a sphere. *Journal of Computational Physics*, 15(1):46–54, 1974.

[73] J. Thuburn, M. Zerroukat, N. Wood, and A. Staniforth. Coupling a mass-conserving semi-Lagrangian scheme (SLICE) to a semi-implicit discretization of

the shallow-water equations: Minimizing the dependence on a reference atmosphere. *Quarterly Journal of the Royal Meteorological Society*, 136(646):146–154, 2010.

[74] L.N. Trefethen and D. Bau. *Numerical linear algebra*. Society for Industrial Mathematics, 1997.

[75] U. Trottenbert, C.W. Oosterlee, and A. Schuller. Multigrid. 2001.

[76] H.A. van der Vorst. Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric systems. *J. Sci. Statist. Comput.*, 13(609):631–644, 1992.

[77] H.K. Versteeg and W. Malalasekera. *An introduction to computational fluid dynamics: the finite volume method*. Prentice Hall, 2007.

[78] A.A. White, B.J. Hoskins, I. Roulstone, and A. Staniforth. Consistent approximate models of the global atmosphere: shallow, deep, hydrostatic, quasi-hydrostatic and non-hydrostatic. *Quarterly Journal of the Royal Meteorological Society*, 131(609):2081–2107, 2005.

[79] D.L. Williamson. The evolution of dynamical cores for global atmospheric models. *J. Meteorol. Soc. Jpn.*, 85B:241–269, 2007.

[80] M. Wlasak. Control Variable Transforms: Parameter Transforms, 2006.

[81] M. Wlasak, N.K. Nichols, and I. Roulstone. Use of potential vorticity for incremental data assimilation. *Quarterly Journal of the Royal Meteorological Society*, 132(621):2867–2886, 2006.

[82] J. Xu. Iterative methods by space decomposition and subspace correction. *Siam Review*, pages 581–613, 1992.

[83] I. Yavneh and J.C. McWilliams. Robust multigrid solution of the shallow-water balance equations. *Journal of Computational Physics*, 119(1):1–25, 1995.

[84] I. Yavneh and J.C. McWilliams. Multigrid solution of stably stratified flows: the quasigeostrophic equations. *Journal of Scientific Computing*, 11(1):47–69, 1996.

[85] I. Yavneh, A.F. Shchepetkin, J.C. McWilliams, and L.P. Graves. Multigrid solution of rotating, stably stratified flows: The balance equations and their turbulent dynamics. *Journal of Computational Physics*, 136(2):245–262, 1997.

[86] H. Yserentant. On the convergence of multi-level methods for strongly nonuniform families of grids and any number of smoothing steps per level. *Computing*, 30(4):305–313, 1983.

[87] H. Yserentant. The convergence of multi-level methods for solving finite-element equations in the presence of singularities. *Mathematics of Computation*, 47(176):399–409, 1986.

[88] H. Yserentant. Old and new convergence proofs for multigrid methods. *Acta numerica*, 2:285–326, 2008.

[89] M. Zerroukat, N. Wood, A. Staniforth, A. White, and J. Thuburn. An inherently mass-conserving semi-implicit semi-Lagrangian discretisation of the shallow-water equations on the sphere. *Quarterly Journal of the Royal Meteorological Society*, 135(642):1104–1116, 2009.

[90] H.B. Zubair, SP MacLachlan, and CW Oosterlee. A geometric multigrid method based on L-shaped coarsening for PDEs on stretched grids. *Numerical Linear Algebra with Applications*, 2009.