Simple and fast multigrid solution of Poisson's equation using diagonally oriented grids

A.J. Roberts*

(Received 7 Septemeber 2000, revised 13 June 2001)

Abstract

We solve Poisson's equation using new multigrid algorithms that converge rapidly. The feature of the 2D and 3D algorithms are the use of diagonally oriented grids in the multigrid hierarchy for a much richer and effective communication between the levels of the multigrid. Numerical investigations into solving Poisson's equation in the unit square and unit cube show simple versions of the proposed algorithms

*Dept. Maths & Computing, University of Southern Queensland, Toowoomba, Queensland 4352, AUSTRALIA. mailto:aroberts@usq.edu.au

[0]See http://anziamj.austms.org.au/V43/E025 for this article and ancillary services,

are up to twice as fast as correspondingly simple multigrid iterations on a standard hierarchy of grids. Similar improvements are found for a basic advection-diffusion equations.

Contents

1 Introduction **E3**

2 A diagonal multigrid for the 2D Poisson equation **E5**
2.1 The smoothing restriction E7
2.2 The Jacobi prolongation E8
2.3 The V-cycle converges rapidly E11
2.4 Optimise parameters of the V-cycle E13
2.5 Application to an advection-diffusion problem E15

3 A diagonal multigrid for the 3D Poisson equation **E17**
3.1 The smoothing restriction steps E20
3.2 The Jacobi prolongation steps E26
3.3 The V-cycle converges well E30
3.4 Optimise parameters of the V-cycle E32

4 Conclusion **E33**

References **E34**

1 Introduction

Multigrid algorithms are effective in the solution of elliptic problems and have found many applications, especially in fluid mechanics [6, e.g.], chemical reactions in flows [10, e.g.] and flows in porous media [7]. Typically, errors in a solution may decrease by a factor of 0.1 each iteration [5, e.g.]. The simple algorithms I present decrease errors by a factor of 0.05 (see Tables 1 and 3 on pages E12 and E32). Further gains in the rate of convergence may be made by further research into more sophisticated versions of the algorithms.

Generally, geometrically based multigrid algorithms use a hierarchy of grids whose grid spacings are proportional to $2^{-\ell}$ where ℓ is the level of the grid [12, 11, e.g.]. The promising possibility reported here is the use of a richer hierarchy of grids with levels of the grids oriented diagonally to other levels. Specifically, for two spatial dimensions (2D) I introduce in Section 2 a hierarchy of grids with grid spacings proportional to $2^{-\ell/2}$ and with grids aligned at 45° to adjacent levels, see Figure 1 (pE6). In three spatial dimensions (3D) the geometry of the grids is much more complicated; in Section 3 we introduce and analyse a hierarchy of 3D grids with grid spacings roughly $2^{-\ell/3}$ on the different levels, see Figure 5 (pE19). Algebraic multigrid methods are based only upon the matrix of the linear equations. These more flexible methods have previously generated the hierarchy of diagonally oriented grids in 2D, see Example 1 by Reusken [8, p577], but appear not to have been investigated in 3D. However, it seems that because of their generality the algebraic based multigrid algorithms do not converge as quickly as

that attained here. The geometrically based hierarchy of diagonal grids employed here are an alternative to the semi-coarsening hierarchy of multigrids used by Dendy [1] and algebraic multigrid methods [8, 11] in more difficult problems.

Now Laplace's operator is isotropic so that its discretisation is straightforward on diagonally oriented grids. Thus in this initial work we explore primarily the solution of Poisson's equation,

$$\nabla^2 u = f \,, \tag{1}$$

although in Section 2.5 we investigate a simple anisotropic advection-diffusion problem. Given an approximation $\tilde{u}$ to a solution, each complete iteration of a multigrid scheme seek a correction v so that $u = \tilde{u} + v$ is a better approximation to a solution of Poisson's equation (1). Consequently the update v has to approximately satisfy a Poisson's equation itself, namely

$$\nabla^2 v = r \quad \text{where} \quad r = f - \nabla^2 \tilde{u} \tag{2}$$

is the residual of the current approximation. The multigrid algorithms aim to estimate the error v as accurately as possible from the residual r. Accuracy in the ultimate solution u is determined by the accuracy of the spatial discretisation in the computation of the residual r: here we investigate second-order and fourth-order accurate discretisations [12, e.g.] but so far only find remarkably rapid convergence for second-order discretisations.

In this initial research we only examine the simplest reasonable V-cycle on the hierarchy of grids and use only one Jacobi iteration on each grid. We

find in Sections 2.1 and 3.1 that the smoothing restriction step from one grid to the coarser diagonally orientated grid is done quite simply. Yet the effective smoothing operator from one level to that a factor of 2 coarser, being the convolution of two or three intermediate steps, is relatively sophisticated. One saving in using these diagonally orientated grids is that there is no need to do any interpolation. Thus the transfer of information from a coarser to a finer grid only involves the simple Jacobi iterations described in Sections 2.2 and 3.2. Performance is enhanced within this class of simple multigrid algorithms by a little over relaxation in the Jacobi iteration as found in Sections 2.4 and 3.4. The proposed multigrid algorithms are found to be up to twice as fast as comparably simple conventional multigrid algorithms.

2 A diagonal multigrid for the 2D Poisson equation

To approximately solve Poisson's equation (2) in 2D we use a novel hierarchy of grids in the multigrid method. The length scales of the grid are proportional to $2^{-\ell/2}$. If the finest grid is aligned with the coordinate axes with grid spacing h say, the first coarser grid is at 45° with spacing $\sqrt{2}h$, the second coarser is once again aligned with the axes and of spacing $2h$, as shown in Figure 1, and so on for all other levels on the multigrid. In going from one level to the next coarser level the number of grid points halves.

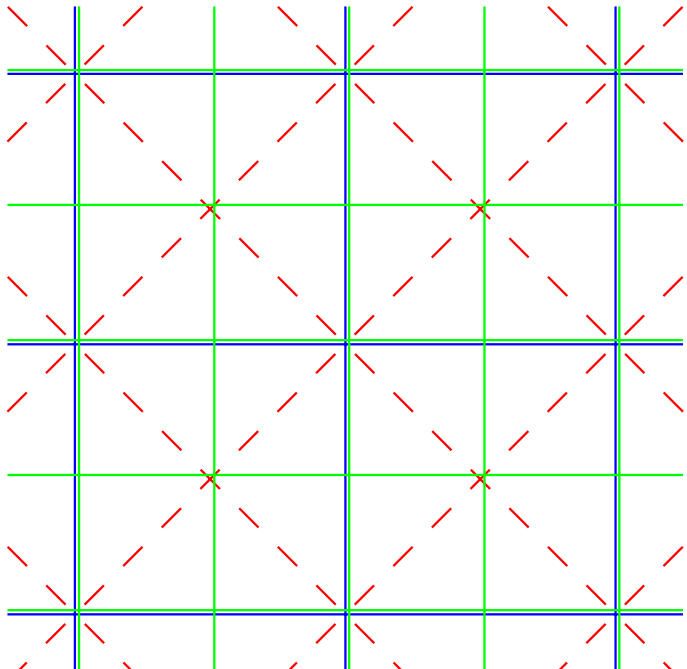

FIGURE 1: three levels of grids in the 2D multigrid hierarchy: the dotted green grid is the finest, spacing h say; the dashed red grid is the next finest diagonal grid with spacing $\sqrt{2}h$; the solid blue grid is the coarsest shown grid with spacing $2h$. Coarser levels of the multigrid follow the same pattern.

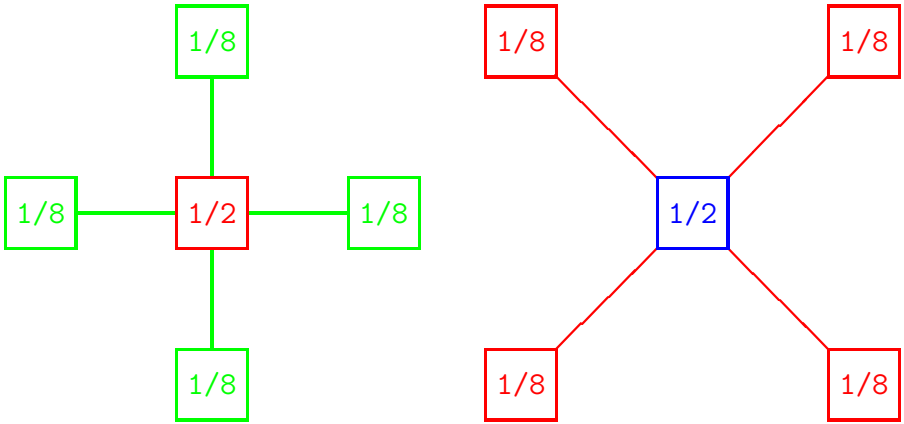

FIGURE 2: restriction stencils are simple weighted averages of neighbouring grid points on all levels of the grid.

2.1 The smoothing restriction

The restriction operator smoothing the residual from one grid to the next coarser grid is the same at all levels. It is simply a weighted average of the grid point and the four nearest neighbours on the finer grid as shown in Figure 2. To restrict from a fine green grid to the diagonal red grid

$$r_{i,j}^{\ell-1} = \frac{1}{8}\left(4r_{i,j}^{\ell} + r_{i-1,j}^{\ell} + r_{i,j-1}^{\ell} + r_{i+1,j}^{\ell} + r_{i,j+1}^{\ell}\right), \tag{3}$$

whereas to restrict from a diagonal red grid to the coarser blue grid

$$r^{\ell-1}_{i,j} = \frac{1}{8}\left(4r^{\ell}_{i,j} + r^{\ell}_{i-1,j-1} + r^{\ell}_{i+1,j-1} + r^{\ell}_{i+1,j+1} + r^{\ell}_{i-1,j+1}\right) . \tag{4}$$

Each of these restrictions takes 6 flops per grid element. Thus assuming the finest grid is $n \times n$ with $N = n^2$ grid points, the restriction to the next finer diagonal grid (red) takes approximately $3N$ flops, the restriction to the next finer takes approximately $3N/2$ flops, etc. Thus to restrict the residuals up $\ell = 2L$ levels to the coarsest grid spacing of $H = 2^L h$ takes

$$K_r \approx 6N\left(1 - \frac{1}{4^L}\right) \text{ flops} \approx 6N \text{ flops} . \tag{5}$$

In contrast a conventional nine point restriction operator from one level to another takes 11 flops per grid point, which then totals to approximately $3\frac{2}{3}N$ flops over the whole conventional multigrid hierarchy. This operation count is somewhat less than the proposed scheme, but we make gains elsewhere. In restricting from the green grid to the blue grid, via the diagonal red grid, the restriction operation is equivalent to a 17-point stencil with a much richer and more effective smoothing than the conventional 9-point stencil.

2.2 The Jacobi prolongation

One immediate saving of this hierarchy of diagonally orientated grids is that there is no need to interpolate in the prolongation step from one level to the

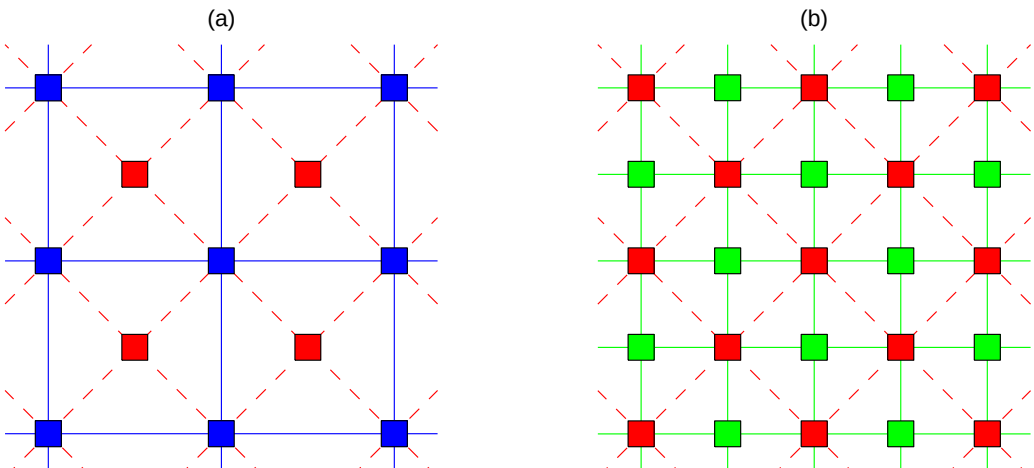

FIGURE 3: the interpolation in a prolongation step is replaced by simply a "red-black" Jacobi iteration: (a) compute the new values at the red grid points, then refine the values at the blue points; (b) compute the new values at the green points, then refine those at the red points.

next finer level. For example, to prolongate from the blue grid to the finer diagonal red grid, shown in Figure 3(a), estimate the new value of v at the red grid points on level ℓ by the red-Jacobi iteration

$$v^\ell_{i,j} = \frac{1}{4}\left(-2h^2 r^\ell_{i,j} + v^{\ell-1}_{i-1,j-1} + v^{\ell-1}_{i+1,j-1} + v^{\ell-1}_{i+1,j+1} + v^{\ell-1}_{i-1,j+1}\right), \tag{6}$$

when the grid spacing on the red grid is $\sqrt{2}h$. Then the values at the blue grid points are refined by the blue-Jacobi iteration

$$v^\ell_{i,j} = \frac{1}{4}\left(-2h^2 r^\ell_{i,j} + v^{\ell}_{i-1,j-1} + v^{\ell}_{i+1,j-1} + v^{\ell}_{i+1,j+1} + v^{\ell}_{i-1,j+1}\right). \tag{7}$$

A similar green-red Jacobi iteration will implicitly prolongate from the red grid to the finer green grid shown in Figure 3(b). These prolongation-iteration steps take 6 flops per grid point. Thus to go from the red to the green grid takes $6N$ flops. As each level of the grid has half as many grid points as the next finer, the total operation count for the prolongation over the hierarchy from grid spacing $H = 2^L h$ is

$$K_p \approx 12N\left(1 - \frac{1}{4^L}\right) \text{ flops} \approx 12N \text{ flops}. \tag{8}$$

The simplest (bilinear) conventional interpolation direct from the blue grid to the green grid would take approximately $2N$ flops, to be followed by $6N$ flops for a Jacobi iteration on the fine green grid (using simply $\nu_1 = 0$ and $\nu_2 = 1$). Over the whole hierarchy this takes approximately $10\frac{2}{3}N$ flops.

Again this is a little less than that proposed here, but the proposed diagonal method achieves virtually two Jacobi iterations instead of just one and so is more effective.

Compare this with the algebraic multigrid Example 1 of Reusken [8, p577]. First note that the red nodes are a maximal independent set of the green nodes, and the blue nodes are a maximal independent set of the red nodes; in this the methods are the same. However, see that the algebraic multigrid restriction to the coarser grid generates a 9-point stencil for the matrix on the coarser grid. This increases the operation count, requires interpolation in effect, and slows the convergence. Instead here we work geometrically to good effect.

2.3 The V-cycle converges rapidly

Numerical investigation shows that although the operation count of the proposed algorithm is a little higher than the simplest standard multigrid scheme, the speed of convergence is much better. The algorithm performs remarkably well on test problems such as those in Gupta *et al* [2]. Here a quantitative comparison between the algorithms shows the proposed diagonal scheme is about twice as fast.

Both the diagonal and usual multigrid algorithms use $7N$ flops to compute the residuals on the finest grid. Thus the proposed method takes approximately $25N$ flops per V-cycle of the multigrid iteration, although 17% more

TABLE 1: comparison of cost, in flops, and performance for various algorithms for solving Poisson's equation in 2D. The column headed "per iter" shows the number of flops per iteration, whereas columns showing "per dig" are flops/$\log_{10}\bar{\rho}$ and indicate the number of flops needed to compute each decimal digit of accuracy. The right-hand columns show the performance for the optimal over relaxation parameter p.

algorithm	per iter	$\bar{\rho}_0$	per dig	p	$\bar{\rho}$	per dig
diagonal, $\mathcal{O}(h^2)$	$25.0N$	.099	$25.0N$	1.052	.052	$19.5N$
standard, $\mathcal{O}(h^2)$	$21.3N$	.340	$45.5N$	1.121	.260	$36.4N$
diagonal, $\mathcal{O}(h^4)$	$30.0N$	.333	$62.8N$	1.200	.200	$42.9N$
standard, $\mathcal{O}(h^4)$	$26.3N$	.343	$56.6N$	1.216	.216	$39.4N$

than the simplest conventional algorithm that takes $21\frac{1}{3}N$ flops, the convergence is much faster. Table 1 shows the rate of convergence $\bar{\rho}_0 \approx 0.1$ for this diagonal multigrid based algorithm. The data is determined using MATLAB's sparse eigenvalue routine to find the largest eigenvalue, and hence the slowest decay, on a 65×65 grid with 12 levels in the multigrid hierarchy. This should be more accurate than limited analytical methods such as a bi-grid analysis [3]. Compared with correspondingly simple schemes based upon the standard hierarchy of grids [11, p11,e.g.], the method proposed here takes much fewer iterations, even though each iteration is a little more expensive, and so should be about twice as fast.

Fourth-order accurate solvers in space may be obtained using the above second-order accurate V-cycle as done by Iyengar & Goyal [4]. The only

necessary change is to compute the residual r in (2) on the finest grid with a fourth-order accurate scheme, such as the compact "Mehrstellen" scheme

$$\begin{aligned} r_{i,j} &= \frac{1}{12}\left(8f_{i,j} + f_{i+1,j} + f_{i,j+1} + f_{i-1,j} + f_{i,j-1}\right) \\ &\quad - \frac{1}{6h^2}\left[-20u_{i,j} + 4\left(u_{i,j-1} + u_{i,j+1} + u_{i-1,j} + u_{i+1,j}\right)\right. \\ &\quad \left. + u_{i+1,j+1} + u_{i-1,j+1} + u_{i-1,j-1} + u_{i+1,j-1}\right] . \end{aligned} \tag{9}$$

Use the V-cycles described above to determine an approximate correction v to the field u based upon these more accurate residuals. The operation count is solely increased by the increased computation in the residual, from $7N$ flops per iteration to $12N$ flops (the combination of f appearing on the right-hand side of (9) need not be computed each iteration). Numerical investigations summarised in Table 1 show that the multigrid methods still converge, but the diagonal method has lost its advantages. Thus fourth-order accurate solutions to Poisson's equation are most quickly obtained by initially using the diagonal multigrid method applied to the second-order accurate computation of residuals. Then use a few multigrid iterations based upon the fourth-order residuals to refine the numerical solution.

2.4 Optimise parameters of the V-cycle

The multigrid iteration is improved by introducing a small amount of over relaxation.

First consider the multigrid method applied to the second-order accurate residuals. Numerical optimisation over a range of introduced parameter values suggested that the simplest, most robust effective change was simply to introduce a parameter p into the Jacobi iterations (6–7) to become

$$v^{\ell}_{i,j} = \frac{1}{4}\left(-2ph^2 r^{\ell}_{i,j} + v^{\ell-1}_{i-1,j-1} + v^{\ell-1}_{i+1,j-1} + v^{\ell-1}_{i+1,j+1} + v^{\ell-1}_{i-1,j+1}\right), \quad (10)$$

$$v^{\ell}_{i,j} = \frac{1}{4}\left(-2ph^2 r^{\ell}_{i,j} + v^{\ell}_{i-1,j-1} + v^{\ell}_{i+1,j-1} + v^{\ell}_{i+1,j+1} + v^{\ell}_{i-1,j+1}\right), \quad (11)$$

on a diagonal red grid and similarly for a green grid. An optimal value of p was determined to be $p = 1.052$. The parameter p just increases the weight of the residuals at each level by about 5%. This simple change, which does not increase the operation count, improves the factor of convergence to $\bar{\rho} \approx 0.052$, which decreases the necessary number of iterations to achieve a given accuracy. As Table 1 shows, this diagonal multigrid is still far better than the standard multigrid even with its optimal choice for over relaxation.

Secondly consider the multigrid method applied to the fourth-order accurate residuals. Numerical optimisation of the parameter p in (10–11) suggests that significantly more relaxation is preferable, namely $p \approx 1.20$. With this choice one V-cycle of the multigrid method generally reduces the residuals by a factor $\bar{\rho} \approx 0.200$. This simple refinement reduces the number of iterations required by about one-third in converging to the fourth-order accurate solution.

2.5 Application to an advection-diffusion problem

Consider in this subsection the anisotropic advection-diffusion problem

$$\nabla^2 u - c u_x = f \,, \tag{12}$$

where c is an advection speed here taken for definiteness to be in the x-direction. In the presence of this advection, we briefly investigate the performance of the multigrid iteration on the diagonally orientated hierarchy introduced in this paper compared to a conventional multigrid hierarchy.

First consider the discretisation on the hierarchy of grids. The dynamical analysis developed in [9, §3] for time-dependent problems suggests that good finite difference approximation to the advection-diffusion problem (12) should have enhanced diffusivity on coarse grids, namely

$$\frac{\nu_1}{h^2}\delta^2 u_{i,j} - \frac{c}{h}\mu\delta u_{i,j} = f_{i,j} \quad \text{where} \quad \nu_1 = \frac{ch}{2}\coth\left(\frac{ch}{2}\right) \tag{13}$$

and δ^2/h^2 and $\mu\delta/h$ are centred difference operators approximating to the Laplacian and ∂_x respectively. Such coarse grid approximations may be particularly good for multigrid methods due to the wide range of grid length scales. Here ν_1 smoothly turns the centred difference approximations on fine grids into upwind approximations on coarse grids.

The simplest modification of the proposed diagonal multigrid method is simply to modify the Jacobi prolongation appropriately. For simplicity we

use unchanged the restriction (3–4) and again there is no need for an interpolation step on the diagonal hierarchy. To account for the advection in the x-direction, at 45° to the red grid in Figure 3(a), the Jacobi prolongation (6–7) is modified to

$$v_{i,j}^{\ell} = \frac{1}{4}\left(-\frac{2h^2}{\nu_1}\left\{r_{i,j}^{\ell} + \frac{c}{4h}\left[v_{i+1,j+1}^{\ell-1} - v_{i-1,j-1}^{\ell-1} + v_{i+1,j-1}^{\ell-1} - v_{i-1,j+1}^{\ell-1}\right]\right\} + v_{i-1,j-1}^{\ell-1} + v_{i+1,j-1}^{\ell-1} + v_{i+1,j+1}^{\ell-1} + v_{i-1,j+1}^{\ell-1}\right), \tag{14}$$

at the red nodes and then on the blue,

$$v_{i,j}^{\ell} = \frac{1}{4}\left(-\frac{2h^2}{\nu_1}\left\{r_{i,j}^{\ell} + \frac{c}{4h}\left[v_{i+1,j+1}^{\ell} - v_{i-1,j-1}^{\ell} + v_{i+1,j-1}^{\ell} - v_{i-1,j+1}^{\ell}\right]\right\} + v_{i-1,j-1}^{\ell} + v_{i+1,j-1}^{\ell} + v_{i+1,j+1}^{\ell} + v_{i-1,j+1}^{\ell}\right). \tag{15}$$

The advection on the finer green grid of Figure 3(b) is a little simpler and leads to the Jacobi prolongation

$$v_{i,j}^{\ell} = \frac{1}{4}\left(-\frac{h^2}{\nu_1}\left\{r_{i,j}^{\ell} + \frac{c}{2h}\left[v_{i+1,j}^{\ell-1} - v_{i-1,j}^{\ell-1}\right]\right\} + v_{i-1,j}^{\ell-1} + v_{i,j-1}^{\ell-1} + v_{i+1,j}^{\ell-1} + v_{i,j+1}^{\ell-1}\right), \tag{16}$$

on the green nodes and then on the red

$$v_{i,j}^{\ell} = \frac{1}{4}\left(-\frac{h^2}{\nu_1}\left\{r_{i,j}^{\ell} + \frac{c}{2h}\left[v_{i+1,j}^{\ell} - v_{i-1,j}^{\ell}\right]\right\} + v_{i-1,j}^{\ell} + v_{i,j-1}^{\ell} + v_{i+1,j}^{\ell} + v_{i,j+1}^{\ell}\right). \tag{17}$$

For the purpose of comparison, a conventional multigrid algorithm was modified by simply replacing its Jacobi iteration step by (16–17).

The modified multigrid algorithms were tried on a number of problems on the unit square. Both algorithms converged for all advection speeds tried (up to $c = 10^5$); without the enhanced dissipation engendered by ν_1 the methods only converge for $|c| < 25$ approximately. For almost all advection speeds c the diagonally based multigrid converged at least twice as quickly. To quantify the rate of convergence I plot in Figure 4 the decay rate per iteration, $\bar{\rho}_0$, of the mode of slowest decay for each of the two methods. See that the diagonal multigrid method proposed here, although converging slower for larger advection speed c, is always significantly better than the corresponding conventional multigrid, up to five times as fast for large advection speeds.

The diagonal hierarchy of grids appears to work well for advection-diffusion problems as well as Poisson problems.

3 A diagonal multigrid for the 3D Poisson equation

The hierarchy of grids we investigate for solving Poisson's equation (2) in 3D is significantly more complicated than that in 2D. Figure 5 shows the three steps between levels that will be taken to go from a fine standard grid (green) of spacing h, via two intermediate grids (red and magenta), to a

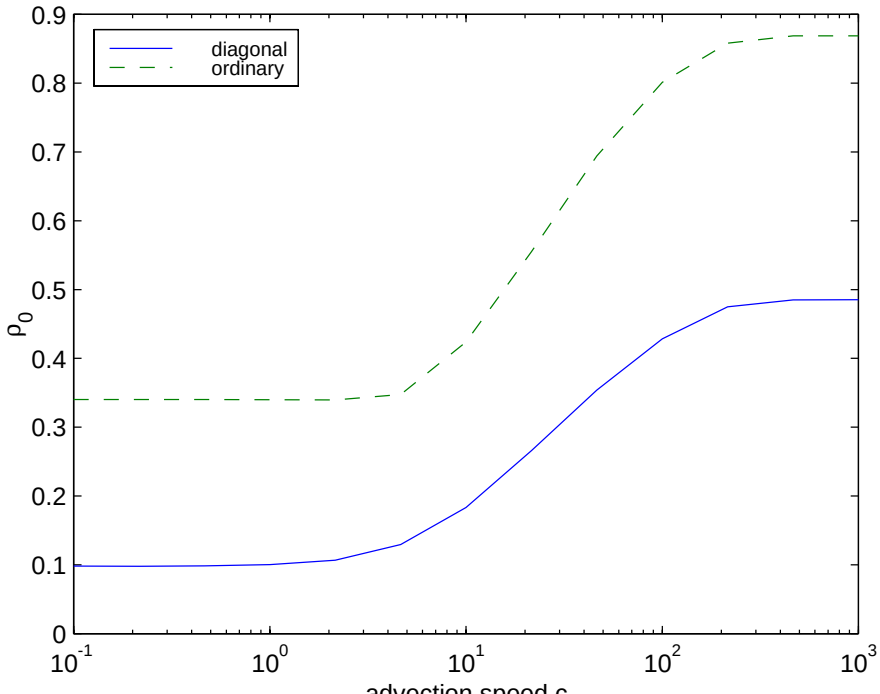

FIGURE 4: the largest eigenvalue of the diagonal (blue solid) and ordinary (green dashed) multigrid algorithms as a function of advection speed c showing the diagonal multigrid is always significantly better. The eigenvalues were determined on a 65×65 grid.

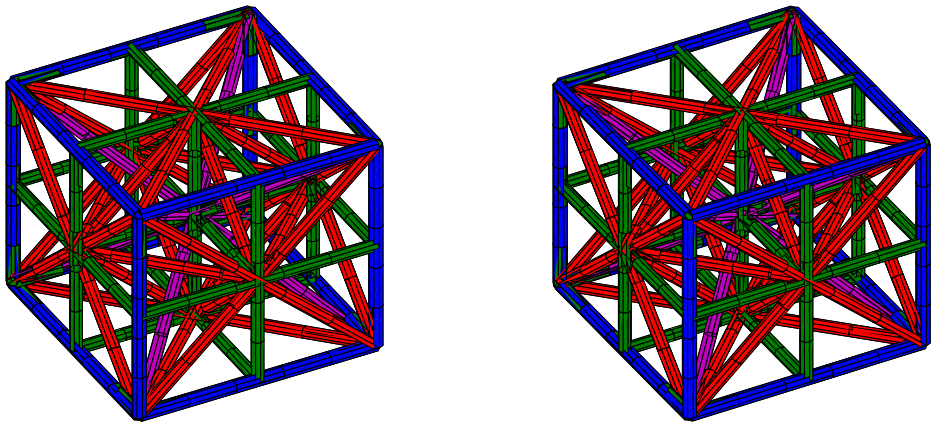

FIGURE 5: one cell of an amalgam of four levels of the hierarchy of grids used to form the multigrid V-cycle in 3D: green is the finest grid shown; red is the next level coarser grid; magenta shows the next coarser grid; and the blue cube is the coarsest to be shown. This stereoscopic view is to be viewed cross-eyed as this seems to be more robust to changes of viewing scale.

coarser regular grid (blue) of spacing $2h$. Although the number of nodes exactly halves in going from one level to another, as we discuss below, there is an unusual aspect in the hierarchy that needs special treatment.

3.1 The smoothing restriction steps

The restriction operation in averaging the residuals from one grid to the next coarser grid is reasonably straightforward.

- The nodes of the red grid are at the corners of the cube and the centre of each of the faces as seen in Figure 6. They each have six neighbours on the green grid so the natural restriction averaging of the residuals onto the red grid is

$$r^{\ell-1}_{i,j,k} = \frac{1}{12}\Big(6r^{\ell}_{i,j,k} + r^{\ell}_{i+1,j,k} + r^{\ell}_{i-1,j,k} + r^{\ell}_{i,j+1,k} + r^{\ell}_{i,j-1,k} + \\ + r^{\ell}_{i,j,k+1} + r^{\ell}_{i,j,k-1}\Big)\,, \tag{18}$$

  for (i, j, k) corresponding to the (red) corners and faces of the coarse (blue) grid. When the fine green grid is $n \times n \times n$ so that there are $N = n^3$ unknowns on the fine green grid, this average takes 8 flops for each of the approximately $N/2$ red nodes. This operation count totals $4N$ flops.

  Note that throughout this discussion of restriction from the green to blue grids via the red and magenta, we index variables using subscripts appropriate to the fine green grid. This also holds for the subsequent discussion of the prolongation from blue to green grids.

- The nodes of the next coarser grid, magenta, are at the corners and centres of the cube as seen in Figure 7. Observe that the centre nodes

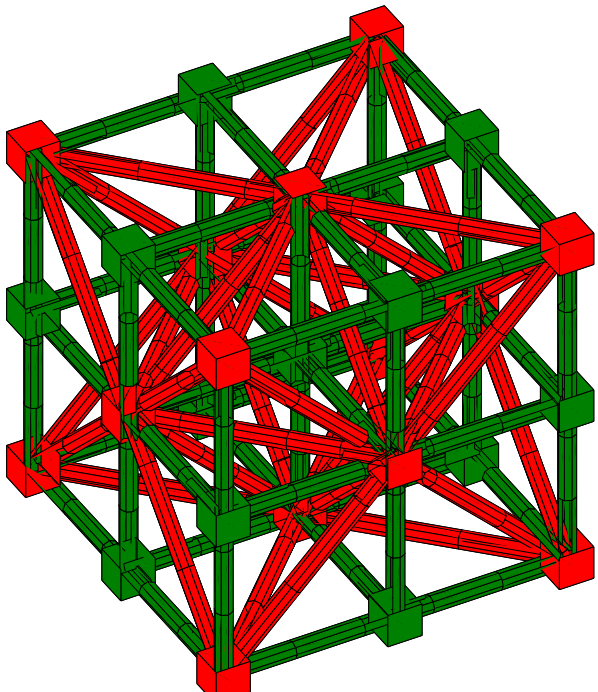

FIGURE 6: the green and red grids superimposed showing the nodes of the red grid at the corners and faces of the cube, and their relationship to their six neighbouring nodes on the finer green grid.

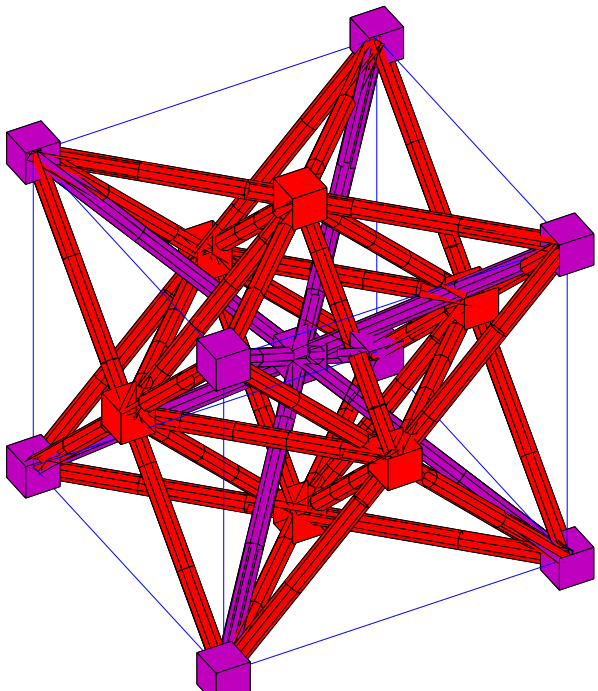

FIGURE 7: the red and magenta grids superimposed showing the nodes of the magenta grid at the corners and the centre of the (blue) cube.

of the magenta grid are not also nodes of the finer red grid; this causes some complications in the treatment of the two different types of magenta nodes. The magenta nodes at the corners are connected to twelve neighbours on the red grid so the natural average of the residuals is

$$\begin{aligned} r^{\ell-1}_{i,j,k} \;=\; \frac{1}{24}\Big(&12r^{\ell}_{i,j,k} + r^{\ell}_{i+1,j+1,k} + r^{\ell}_{i+1,j-1,k} + r^{\ell}_{i-1,j-1,k} + r^{\ell}_{i-1,j+1,k} + \\ &+ r^{\ell}_{i+1,j,k+1} + r^{\ell}_{i+1,j,k-1} + r^{\ell}_{i-1,j,k-1} + r^{\ell}_{i-1,j,k+1} + \\ &+ r^{\ell}_{i,j+1,k+1} + r^{\ell}_{i,j+1,k-1} + r^{\ell}_{i,j-1,k-1} + r^{\ell}_{i,j-1,k+1}\Big)\,, \end{aligned} \tag{19}$$

for (i, j, k) corresponding to the magenta corner nodes. This average takes 14 flops for each of $N/8$ nodes. The magenta nodes at the centre of the coarse (blue) cube is not connected to red nodes by the red grid, see Figure 7. However, it has six red nodes in close proximity, those at the centre of the faces, so the natural average is

$$r^{\ell-1}_{i,j,k} = \frac{1}{6}\left(r^{\ell}_{i+1,j,k} + r^{\ell}_{i-1,j,k} + r^{\ell}_{i,j+1,k} + r^{\ell}_{i,j-1,k} + r^{\ell}_{i,j,k+1} + r^{\ell}_{i,j,k-1}\right)\,, \tag{20}$$

for (i, j, k) corresponding to the magenta centre nodes. This averaging takes 6 flops for each of $N/8$ nodes. The operation count for all of this restriction step from red to magenta is $2\frac{1}{2}N$ flops.

Now compare this restriction with the algebraic multigrid principles of Reusken [8] or Wagner [11, p41]. See that the red nodes are a maximally independent set on the green grid and so could be found by an algebraic multigrid method, and similarly as we see next the blue nodes are a

maximally independent set on the magenta grid. However, the centre magenta nodes are not on the red grid and so the magenta grid could not be found by an algebraic multigrid method. This 3D geometrically inspired multigrid method will not be found using algebraic multigrid algorithms.

- The nodes of the coarse blue grid are at the corners of the shown cube, see Figure 8. On the magenta grid they are connected to eight neighbours, one for each octant, so the natural average of residuals from the magenta to the blue grid is

$$\begin{aligned} r^{\ell-1}_{i,j,k} \;=\; \frac{1}{16}\Big(8r^{\ell}_{i,j,k} + r^{\ell}_{i+1,j+1,k+1} + r^{\ell}_{i+1,j+1,k-1} + r^{\ell}_{i+1,j-1,k+1} + \\ + r^{\ell}_{i+1,j-1,k-1} + r^{\ell}_{i-1,j+1,k+1} + r^{\ell}_{i-1,j+1,k-1} + \\ + r^{\ell}_{i-1,j-1,k+1} + r^{\ell}_{i-1,j-1,k-1}\Big)\,, \end{aligned} \tag{21}$$

for (i, j, k) corresponding to the blue corner nodes. This averaging takes 10 flops for each of $N/8$ blue nodes which thus totals $1\frac{1}{4}N$ flops.

These three restriction steps, to go up three levels of grids, thus total approximately $7\frac{3}{4}N$ flops. Hence, the entire restriction process, averaging the residuals, from a finest grid of spacing h up $3L$ levels to the coarsest grid of spacing $H = 2^L h$ takes

$$K_r \approx \frac{62}{7}N\left(1 - \frac{1}{8^L}\right) \text{ flops} \approx 8\tfrac{6}{7}N \text{ flops}\,. \tag{22}$$

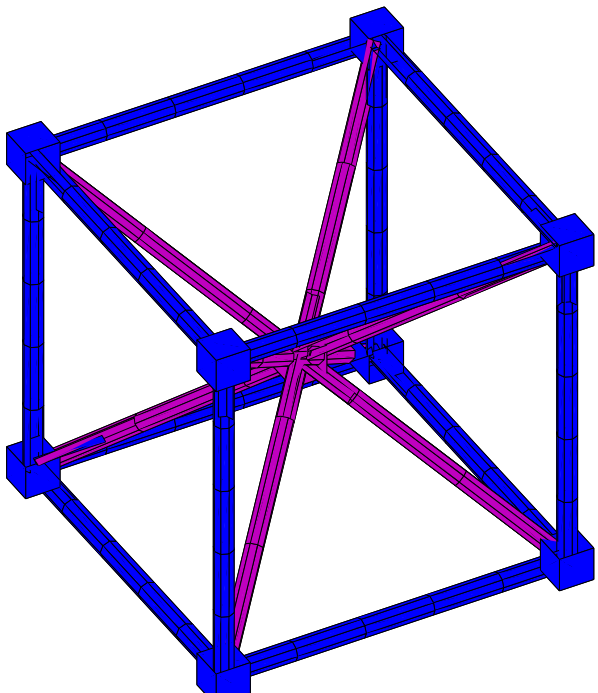

FIGURE 8: the magenta and blue grids superimposed showing the common nodes at the corners of the blue grid and the connections to the magenta centre node.

The simplest standard one-step restriction direct from the fine green grid to the blue grid takes approximately $3\frac{3}{4}N$ flops. Over the whole hierarchy this totals $4\frac{2}{7}N$ flops which is roughly half that of the proposed method. We anticipate that rapid convergence of the V-cycle makes the increase worthwhile.

3.2 The Jacobi prolongation steps

As in 2D, with this rich structure of grids we have no need to interpolate when prolongating from a coarse grid onto a finer grid; an appropriate "red-black" Jacobi iteration of the residual equation (2) avoids interpolation. Given an estimate of corrections $v^{\ell}_{i,j,k}$ at some blue level grid we proceed to the finer green grid via the following three prolongation steps.

- Perform a magenta-blue Jacobi iteration on the nodes of the magenta grid shown in Figure 8. See that each node on the magenta grid is connected to eight neighbours distributed symmetrically about it, each contributes to an estimate of the Laplacian at the node. Thus, given initial approximations on the blue nodes from the coarser blue grid,

$$\begin{aligned} v^{\ell}_{i,j,k} \;=\; \frac{1}{8}\Big(&-4p_m h^2 r^{\ell}_{i,j,k} + v^{\ell-1}_{i+1,j+1,k+1} + v^{\ell-1}_{i+1,j+1,k-1} + v^{\ell-1}_{i+1,j-1,k+1} + \\ &+ v^{\ell-1}_{i+1,j-1,k-1} + v^{\ell-1}_{i-1,j+1,k+1} + v^{\ell-1}_{i-1,j+1,k-1} + \\ &+ v^{\ell-1}_{i-1,j-1,k+1} + v^{\ell-1}_{i-1,j-1,k-1}\Big)\,, \end{aligned} \tag{23}$$

for (i,j,k) on the centre magenta nodes. The following blue-Jacobi iteration uses these updated values in the similar formula

$$\begin{aligned} v^{\ell}_{i,j,k} &= \frac{1}{8}\Big(-4p_m h^2 r^{\ell}_{i,j,k} + v^{\ell}_{i+1,j+1,k+1} + v^{\ell}_{i+1,j+1,k-1} + v^{\ell}_{i+1,j-1,k+1} + \\ &\quad + v^{\ell}_{i+1,j-1,k-1} + v^{\ell}_{i-1,j+1,k+1} + v^{\ell}_{i-1,j+1,k-1} + \\ &\quad + v^{\ell}_{i-1,j-1,k+1} + v^{\ell}_{i-1,j-1,k-1}\Big)\,, \end{aligned} \tag{24}$$

for (i,j,k) on the corner blue nodes. In these formulae the over relaxation parameter p_m has been introduced for later fine tuning; initially take $p_m = 1$. The operation count for this magenta-blue Jacobi iteration is 10 flops on each of $N/4$ nodes giving a total of $2\frac{1}{2}N$ flops.

- Perform a red-magenta Jacobi iteration on the nodes of the red grid shown in Figure 7. However, because the centre node (magenta) is not on the red grid, two features follow: it is not updated in this prolongation step; and it introduces a little asymmetry into the weights used for values at the nodes. The red nodes in the middle of each face are surrounded by four magenta nodes at the corners and two magenta nodes at the centres of the cube. However, the nodes at the centres are closer and so have twice the weight in the estimate of the Laplacian. Hence, given initial approximations on the magenta nodes from the coarser grid,

$$\begin{aligned} v^{\ell}_{i,j,k} &= \frac{1}{8}\Big(-2p_{r1} h^2 r^{\ell}_{i,j,k} + 2\left[v^{\ell-1}_{i,j,k+1} + v^{\ell-1}_{i,j,k-1}\right] + \\ &\quad + v^{\ell-1}_{i+1,j+1,k} + v^{\ell-1}_{i+1,j-1,k} + v^{\ell-1}_{i-1,j-1,k} + v^{\ell-1}_{i-1,j+1,k}\Big)\,, \end{aligned} \tag{25}$$

for (i,j,k) corresponding to the red nodes on the centre of faces normal to the z-direction. Similar formulae apply for red nodes on other faces, cyclically permute the role of the indices. The over relaxation parameters p_{r1} and p_{r2} are introduced for later fine tuning; initially take $p_{r1} = p_{r2} = 1$. The following magenta-Jacobi iteration uses these updated values. Each magenta corner node in Figure 7 is connected to twelve red nodes and so is updated according to

$$\begin{aligned} v^\ell_{i,j,k} &= \frac{1}{12}\Big(-4p_{r2}h^2 r^\ell_{i,j,k} + \\ &\quad + v^\ell_{i+1,j+1,k} + v^\ell_{i+1,j-1,k} + v^\ell_{i-1,j-1,k} + v^\ell_{i-1,j+1,k} + \\ &\quad + v^\ell_{i+1,j,k+1} + v^\ell_{i+1,j,k-1} + v^\ell_{i-1,j,k-1} + v^\ell_{i-1,j,k+1} + \\ &\quad + v^\ell_{i,j+1,k+1} + v^\ell_{i,j+1,k-1} + v^\ell_{i,j-1,k-1} + v^\ell_{i,j-1,k+1}\Big)\,, \end{aligned} \tag{26}$$

for all (i,j,k) corresponding to corner magenta nodes. The operation count for this red-magenta Jacobi iteration is 9 flops on each of $3N/8$ nodes and 14 flops on each of $N/8$ nodes. These total $5\frac{1}{8}N$ flops.

- Perform a green-red Jacobi iteration on the nodes of the fine green grid shown in Figure 6. The green grid is a standard rectangular grid so the Jacobi iteration is also standard. Given initial approximations on the red nodes from the coarser red grid,

$$\begin{aligned} v^\ell_{i,j,k} &= \frac{1}{6}\Big(-p_g h^2 r^\ell_{i,j,k} + v^{\ell-1}_{i+1,j,k} + v^{\ell-1}_{i-1,j,k} + v^{\ell-1}_{i,j+1,k} + v^{\ell-1}_{i,j-1,k} + \\ &\quad + v^{\ell-1}_{i,j,k+1} + v^{\ell-1}_{i,j,k-1}\Big)\,, \end{aligned} \tag{27}$$

for (i, j, k) corresponding to the green nodes (edges and centre of the cube). The over relaxation parameter p_g, initially $p_g = 1$, is introduced for later fine tuning. The red-Jacobi iteration uses these updated values in the similar formula

$$v^{\ell}_{i,j,k} = \frac{1}{6}\left(-p_g h^2 r^{\ell}_{i,j,k} + v^{\ell}_{i+1,j,k} + v^{\ell}_{i-1,j,k} + v^{\ell}_{i,j+1,k} + v^{\ell}_{i,j-1,k} + \right. \\ \left. + v^{\ell}_{i,j,k+1} + v^{\ell}_{i,j,k-1}\right), \tag{28}$$

for the red nodes in Figure 6. This prolongation step is a standard Jacobi iteration and takes 8 flops on each of N nodes for a total of $8N$ flops.

These three prolongation steps together thus total $15\frac{5}{8}N$ flops. To prolongate over $\ell = 3L$ levels from the coarsest grid of spacing $H = 2^L h$ to the finest grid thus takes

$$K_p \approx \frac{125}{7}N\left(1 - \frac{1}{8^L}\right) \text{ flops} \approx 17\tfrac{6}{7}N \text{ flops}. \tag{29}$$

The simplest trilinear interpolation direct from the blue grid to the green grid would take approximately $3\frac{1}{4}N$ flops, to be followed by $8N$ flops for a Jacobi iteration on the fine green grid. Over the whole hierarchy this standard prolongation takes approximately $12\frac{6}{7}N$ flops. This total is smaller, but the proposed diagonal grid achieves virtually three Jacobi iterations instead of one and so is more effective.

TABLE 2: comparison of cost, in flops, and performance for unoptimised algorithms for solving Poisson's equation in three spatial dimensions on a 17^3 grid. The column headed "per iter" shows the number of flops per iteration, whereas column showing "per dig" is flops/$\log_{10}\bar{\rho}_0$ and indicates the number of flops needed to compute each decimal digit of accuracy.

algorithm	per iter	$\bar{\rho}_0$	per dig
diagonal, $\mathcal{O}(h^2)$	$35.7N$	0.140	$42N$
standard, $\mathcal{O}(h^2)$	$26.1N$	0.477	$81N$
diagonal, $\mathcal{O}(h^4)$	$48.7N$	0.659	$269N$
standard, $\mathcal{O}(h^4)$	$39.1N$	0.651	$210N$

3.3 The V-cycle converges well

Numerical investigation shows that, as in 2D, although the operation count of the proposed algorithm is a little higher, the speed of convergence is much better. Both algorithms use $9N$ flops to compute second-order accurate residuals on the finest grid. Thus the proposed method takes approximately $35\frac{5}{7}N$ flops for one V-cycle, some 37% more than the $26\frac{1}{7}N$ flops of the simplest standard algorithm. It achieves a mean factor of convergence $\bar{\rho} \approx 0.140$. This rapid rate of convergence easily compensates for the small increase in computations taking half the number of flops per decimal digit accuracy determined.

As in 2D, fourth-order accurate solvers may be obtained simply by using

the above second-order accurate V-cycle on the fourth-order accurate residuals evaluated on the finest grid. A compact fourth-order accurate scheme for the residuals is the 19 point formula

$$\begin{aligned} r_{i,j,k} &= \frac{1}{12}\left(6f_{i,j,k} + f_{i+1,j,k} + f_{i,j+1,k} + f_{i-1,j,k} + f_{i,j-1,k} + f_{i,j,k+1} + \right. \\ &\quad \left. + f_{i,j,k-1}\right) - \frac{1}{6h^2}\left[-24u_{i,j,k} + 2\left(u_{i,j-1,k} + u_{i,j+1,k} + u_{i-1,j,k} + \right.\right. \\ &\quad \left. + u_{i+1,j,k} + u_{i,j,k+1} + u_{i,j,k-1}\right) + u_{i+1,j+1,k} + u_{i-1,j+1,k} + \\ &\quad + u_{i-1,j-1,k} + u_{i+1,j-1,k} + u_{i,j+1,k+1} + u_{i,j+1,k-1} + u_{i,j-1,k-1} + \\ &\quad \left. + u_{i,j-1,k+1} + u_{i+1,j,k+1} + u_{i-1,j,k+1} + u_{i-1,j,k-1} + u_{i+1,j,k-1}\right] \end{aligned} \tag{30}$$

Then using the V-cycle described above to determine corrections v to the field u leads to an increase in the operation count of $13N$ flops solely from the extra computation in finding the finest residuals. Numerical investigations show that the multigrid iteration still converges, albeit much slower, with $\bar{\rho} \approx 0.659$. Table 2 shows that the rate of convergence on the diagonal hierarchy of grids is little different than that for the simplest standard multigrid algorithm. As in 2D, high accuracy, fourth-order solutions to Poisson's equation are best found by employing a first stage that finds second-order accurate solutions which are then refined in a second stage.

TABLE 3: comparison of cost, in flops, and performance for optimised algorithms for solving Poisson's equation in three spatial dimensions on a 17^3 grid varying over relaxation parameters to determine the best rate of convergence. The column headed "per iter" shows the number of flops per iteration, whereas column showing "per dig" is flops/$\log_{10}\bar{\rho}$ and indicates the number of flops needed to compute each decimal digit of accuracy.

algorithm	per iter	p_m	p_{r1}	p_{r2}	p_g	$\bar{\rho}$	per dig
diag, $\mathcal{O}(h^2)$	$35.7N$	1.11	1.42	1.08	0.99	0.043	$26N$
standard, $\mathcal{O}(h^2)$	$26.1N$				1.30	0.31	$51N$
diag, $\mathcal{O}(h^4)$	$48.7N$	0.91	0.80	0.70	1.77	0.39	$119N$
standard, $\mathcal{O}(h^4)$	$39.1N$				1.70	0.41	$101N$

3.4 Optimise parameters of the V-cycle

As in 2D, the multigrid algorithms are improved by introducing some relaxation in the Jacobi iterations. The four parameters p_m, p_{r1}, p_{r2} and p_g were introduced in the Jacobi iterations (23–28) to do this, values bigger than 1 correspond to some over relaxation.

The search for the optimum parameter set used the Nelder-Mead simplex method encoded in the procedure FMINS in MATLAB. Searches were started from optimum parameters found for coarser grids. As tabulated in Table 3 the optimum parameters on a 17^3 grid[1] were $p_m = 1.11$, $p_{r1} = 1.42$, $p_{r2} =$

[1]Systematic searches on a finer grid were infeasible within one days computer time

1.08 and $p_g = 0.99$ and achieve an astonishingly fast rate of convergence of $\bar{\rho} \approx 0.043$. This ensures convergence to a specified precision at half the cost of the similarly optimised, simple conventional multigrid algorithm.

For the fourth-order accurate residuals an optimised diagonal multigrid performs similarly to the optimised conventional multigrid with a rate of convergence of $\bar{\rho} \approx 0.39$. Again fourth-order accuracy is best obtained after an initial stage in which second-order accuracy is used.

4 Conclusion

The use of a hierarchy of grids at angles to each other can halve the cost of solving Poisson's equation to second-order accuracy in grid spacing. Each iteration of the optimised *simplest* multigrid algorithm decreases errors by a factor of at least 20. This is true in both two and three dimensional problems. Further research is needed to investigate the effective of extra Jacobi iterations at each level of the diagonal grid.

When compared with the amazingly rapid convergence obtained for the second-order scheme, the rate of convergence when using the fourth-order residuals is relatively pedestrian. This suggests that a multigrid V-cycle

due to the large number of unknowns: approximately 30,000 components occur in the eigenvectors on a 33^3 grid.

specifically tailored on these diagonal grids for the fourth-order accurate problem may improve convergence markedly.

There is more scope for W-cycles to be effective using these diagonal grids because there are many more levels in the multigrid hierarchy. An exploration of this aspect of the algorithm is also left for further research.

Acknowledgement: This research has been supported by a grant from the Australian Research Council.

References

[1] J. E. Dendy. Semicoarsening multgrid for systems. *Elect. Trans. Num. Anal.*, 6:97–105, 1997. E4

[2] M. M. Gupta, J. Kouatchou, and J. Zhang. Comparison of second- and fourth-order discretizations for multigrid Poisson solvers. *J. Comput. Phys.*, 132:226–232, 1997. E11

[3] S. O. Ibraheem and A. O. Demuren. On bi-grid local mode analysis of solution techniques for 3-D Euler and NavierStokes equations. *J. Comput. Physics*, 125:354–377, 1996. E12

[4] R. K. Iyengar and A. Goyal. A note on multigrid for the three-dimensional Poisson equation in cylindrical coordinates. *J. Comput. & Appl. Math.*, 33:163–169, 1990. E12

[5] D. J. Mavriplis. Directional coarsening and smoothing for anisotropic Navier-Stokes problems. *Elect. Trans. Num. Anal.*, 6:182–197, 1997. E3

[6] D. J. Mavriplis. Unstructured grid techniques. *Annu. Rev. Fluid Mech.*, 29:473–514, 1997. E3

[7] J. D. Moulton, J. E. Dendy, and J. M. Hyman. The black box multigrid numerical homogenization algorithm. *J. Comput. Physics*, 142:80–108, 1998. E3

[8] Arnold Reusken. On the approximate cyclic reduction preconditioner. *SIAM J. Sci. Comput.*, 21:565–590, 1999. E3, E4, E11, E23

[9] A. J. Roberts. A holistic finite difference approach models linear dynamics consistently. Technical report, [http://arXiv.org/abs/math.NA/0003135], March 2000. E15

[10] S. G. Sheffer, L. Martinelli, and A. Jameson. An efficient multigrid algorithm for compressible reactive flows. *J. Comput. Physics*, 144:484–516, 1998. E3

[11] Christian Wagner. Introduction to algebraic multigrid. Course notes of an algebraic multigrid course at the University of Heidelberg in the

wintersemester 1998/99. Technical report, University of Heidelberg, 1999. [http://www.iwr.uni-heidelberg.de/iwr/techsim/chris]. E3, E4, E12, E23

[12] Jun Zhang. Fast and high accuracy multigrid solution of the three-dimensional Poisson equation. *J. Comput. Phys.*, 143:449–461, 1998. E3, E4