Brief introduction to AMG with a Stokes application

Michael Wathen

December 16, 2013

1 Introduction

Consider solving the following linear system

$$Ax = b,$$

where A is a non-singular $n \times n$ matrix. Consider n large enough so that it is not possible to solve using direct methods due to memory usage and time constraints. With problems such as these we would turn to iterative methods. Most iterative methods can be written in the following form:

$$x_{k+1} = x_k + M^{-1} r_k,$$

where $r_k = b - Ax_k$ is the residual at the k^{th} iteration and M is some sort of operator (usually called a preconditioner). Simple M's would be for example the diagonal of A (Jacobi iterations) or the lower triangular part of A (Gauss-Seidel iterations). However, for larger n, the number of iterations it takes for these two schemes to converge will go up and hence they perform very badly on large problems. Instead of using one of these simple iterative methods we would therefore like to choose a method with the property that as problem size increases the number of iterations that the scheme takes to converge does not grow. The method many choose is multigrid. Multigrid can be defined both in a geometric way (geometric multigrid) and in an

algebraic sense (algebraic multigrid). In this project we will look at both of these methods

2 Geometric Multigrid

Geometric multigrid (GMG) is built on the premise that we have a discretisation on a known grid. From this grid (usually uniform for GMG) we can define a sequence of coarser and coarser grids that we would like to approximate our discretisation on. This can be done by the following process:

Step 1: Smoothing

Before approximating the discretisation on a coarser grid we need to smooth the error (residual). This is usually done be performing a few iterations of a simple iterative scheme such as damped Jacobi or Gauss-Seidel (see figure 1). From just a few iterations one can see that the residual has

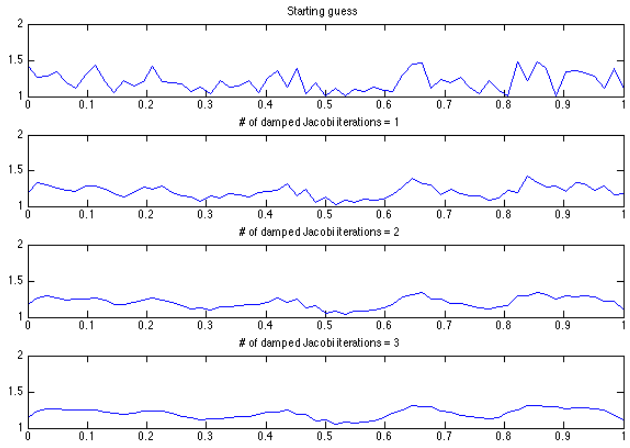


Figure 1: 3 iterations of damped Jacobi

been effectively smoothed, though the magnitude has not been reduced that much. This is why these simple iterative methods have poor convergence properties for smooth residuals. A key idea in multigrid is that smooth residual (r) then can be restricted onto a coarser grid with $m < n$ variables.

Step 2: Restriction

By defining a restriction operator $P^{\mathrm{T}} : \mathbb{R}^n \to \mathbb{R}^m$ we can define the smoothed residual onto the coarser grid by $r_c = P^{\mathrm{T}} r$. On the same coarse grid we can define the coarse grid operator by $A_c = P^{\mathrm{T}} A P$ (the new operator is called the Galerkin operator).

Step 3: Solve

On the coarse grid we solve $A_c e_c = r_c$ for the coarse grid error e_c.

Step 4: Prolongation and correct

From step 3 we solved our smaller system on a coarse mesh. We then need to prolongate the error, e_c, back onto the original mesh. We do this by taking our intial x and adding the prolongated e_c to it, i.e. $x = x + Pe_c$. Notice that the prolongation, P, is the transpose of the restriction operator.

What I have described above is known as a two-grid cycle (see figure 2). Instead of doing a direct solve at step 2 we could apply another two-grid cy-

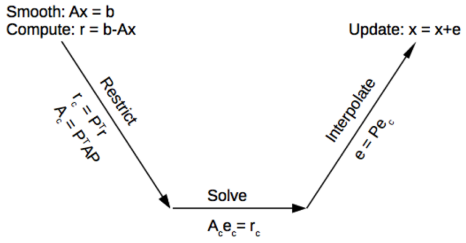


Figure 2: Two grid cycle

cle. Therefore, this algorithm can be done recursively. One type of recursive multigrid cycle is known as a V-Cycle (see figure 3). From the figure we can see that the process is fairly simple: we restrict the residual to coarser and coarser grids with different multiples of h and then interpolate or prolongate back to the original grid with mesh size h.

To illustrate how effective multigrid can be and why we would like to use it, a simple $2D$ MATLAB script was written that performs a simple GMG V-cycle as a preconditioner to Conjugate Gradients (CG) for Poisson's equations. We also tested the code with an incomplete Cholesky factorisation as a preconditioner and MATLAB's inbuilt direct solve to produce the fol-

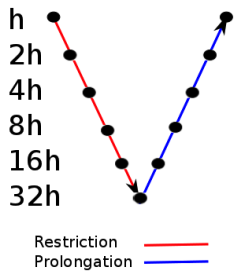


Figure 3: Multigrid V-cycle

lowing table. The - in the table means that we ran out of memory. Table 2

Table 1

Grid	Matrix	MG		ICHOL		Direct
Size	Size	iters	time	iters	time	time
15^2	225	5	0.0113	17	0.0030	0.0004
31^2	961	6	0.0096	30	0.0060	0.0009
63^2	3969	6	0.0193	55	0.0223	0.0042
127^2	16129	6	0.0470	102	0.1103	0.0185
255^2	65025	6	0.1602	187	0.7753	0.0914
511^2	261121	6	0.6182	339	5.473	0.4888
1023^2	1046529	6	2.6700	576	38.4429	2.2417
2047^2	4190209	6	11.1644	1021	279.704	11.6493
4095^2	16769025	6	47.1795	2001	2478.998	-

Table 2: Timing and iteration results for the Poisson problem using GMG

shows that as we increase the mesh size then the number of iterations of CG when using a multigrid V-cycle stays constant. This is the attraction of using a multigrid method. The constant number of iterations we see when using a multigrid V-cycle is not apparent when we use an incomplete Cholesky factorisation. Indeed we see that as the size of the mesh increases the number of iterations roughly doubles. This shows the scalability power of GMG.

3 Algebraic Multigrid

In the previous section we looked at how a GMG scheme worked. One of the main principles of GMG is that we have a sequence of grids (usually uniform grids) on which we can define interpolation and restriction operators. However, if you are considering a discretisation on an unstructured grid then it is very hard (usually impossible) to produce interpolation and restriction operators which would map the fine unstructured grid to a coarse unstructured grid. Therefore, we would like to be able to construct interpolation and restriction operators based solely on the coefficients of the matrix. This leads to an algebraic multigrid (AMG) approach.

In the rest of this section we will go through the process of forming these interpolation and restriction operators for the classical AMG (CAMG) following [Brandt et al., 1985, Ruge and Stüben, 1987, Brandt, 1986].

Smoothness

CAMG is established on the presumption that geometrically smooth functions are in the near null space of A. As we have no geometric information to use we need to find an algebraic way to define smoothness. Defining (λ, e) to be the eigenvalue-eigenvector pairs for a symmetric A, then smoothness is characterised as

$$e^{\mathrm{T}} A e = \lambda \ll 1. \tag{3.1}$$

To make things a little easier we assume that A has been normalised (to ensure that $\lambda_{\max} = 1$). From our assumption (geometrically smooth functions are in the near null space) and the further assumption that the row sums of A are zero then we can rearrange (3.1) to be

$$\begin{aligned} e^{\mathrm{T}} A e &= \sum_i e_i \big(a_{ii} e_i \sum_{i \neq j} a_{ij} e_i \big), \\ &= \sum_i e_i \big(\sum_{i \neq j} (-a_{ij})(e_i - e_j) \big), \\ &= \sum_{i<j} (-a_{ij})(e_i - e_j)^2 \ll 1. \end{aligned} \tag{3.2}$$

From (3.2) we can see that smooth error changes slowly for large matrix coefficients. This leads to one of the main concepts within AMG: Strength of Connection.

Definition 1. *Strength of Connection:*

Given $\theta \in (0, 1]$, variable (point) u_i strongly depends on the variable (point) u_j if

$$-a_{ij} \geq \theta \max_{k \neq i} \{-a_{ik}\}.$$

From **Definition 1**, strength of connection is measured corresponding to the largest off-diagonal entry in each row.

Coarse "grid" selection

In CAMG, the set of coarse points (C-points) are a subset of the fine points (F-points). The C-points are chosen in the direction of strong connections of the matrix. To define the coarse grid we do the following three steps:

1. Define strength matrix A_s using **Definition 1**
2. **Phase 1:** Choose set of fine points based on A_s
3. **Phase 2:** Choose extra points to satisfy interpolation requirements

The easiest way to show this procedure is to look at a simple example. The example we will look at is a finite element discretisation of the 2D Laplacian matrix with the following 9-point stencil:

$$\begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix}. \tag{3.3}$$

This is in fact a discretisation stencil on a uniform grid but the grid selection will not use any geometric information. From the stencil (3.3) we note that all the off-diagonal entries of the matrix will be -1's. Hence, the strength matrix A_s will be exactly the same as the original matrix A.

The degree of a node is defined by how many neighbouring nodes depend upon it. Figure 4 to 6 show the degree of the node at the (i, j) point on the grid. This therefore means that we don't use any geometric information to form these coarse grids.

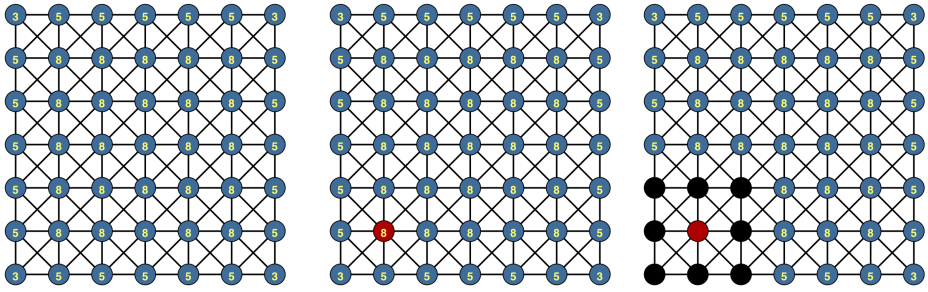
Phase 1: Choose set of C and F-points

i. From strength matrix A_s select a C-point with maximal measure (Figure 4(b))

ii. Set the neighbours of the C-point to be F-points (Figure 4(c))

iii. Update the F-point neighbours so that they are more likely to be chosen (Figure 5(a))

iv. Repeat steps i. to iii. (Figure 5 and 6)



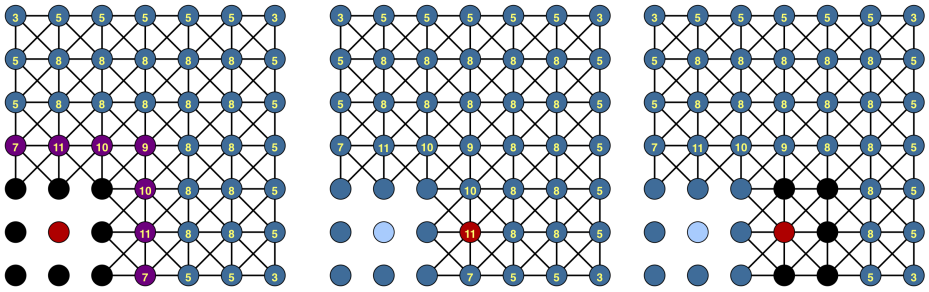
(a) Step 1 (b) Step 2 (c) Step 3

Figure 4: Steps 1 to 3 from [Falgout, 2006]

Carrying out this sequential process yields the coarsened grid given in figure 7.

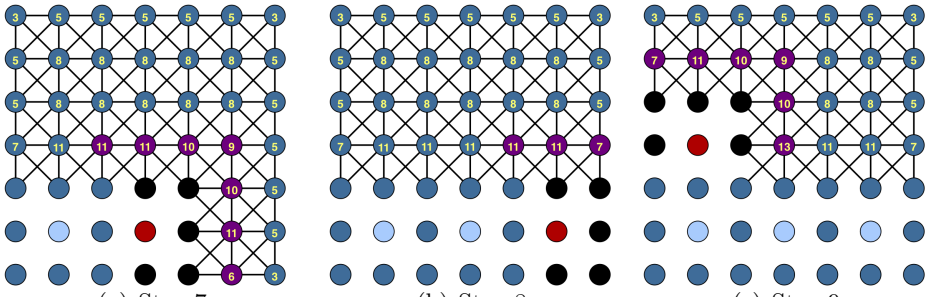
Phase 2: Satisfy interpolation requirements

CAMG requires that all strongly connected F-point pairs must be strongly connected to a common C-point. Therefore, we loop through the F-points to find strongly connected pairs. If we find strongly connected pairs then we check if they are strongly connected to a common C-point. If they



(a) Step 4 (b) Step 5 (c) Step 6

Figure 5: Steps 4 to 6 from [Falgout, 2006]



(a) Step 7 (b) Step 8 (c) Step 9

Figure 6: Steps 7 to 9 from [Falgout, 2006]

are then we move onto the next F-point pair, otherwise we make one of the F-points a C-point.

This second phase leads to higher complexity within the algorithm so more recent AMG methods don't do this and modify interpolation instead.

Interpolation

Recall that we defined algebraic smoothness by (3.1). Since the residual can be defined as $r = Ae$, then re-writing (3.1) gives us:

$$\lambda^2 = e^{\mathrm{T}} A^2 e = r^{\mathrm{T}} r = ||r|| \ll 1.$$

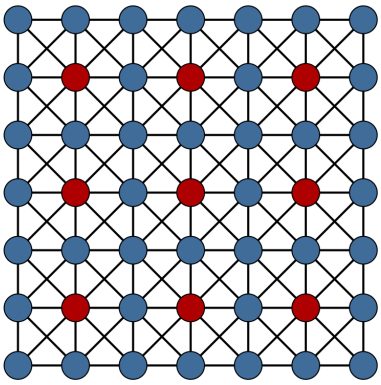


Figure 7: C-point (red) and F-points (blueish) from [Falgout, 2006]

This shows that we can characterise smoothness by a small residual. CAMG uses this property to define interpolation operators. CAMG sets the residual to be point wise 0 i.e.

$$r_i = (Ae)_i = 0.$$

Re-write this equation in terms of 3 separate sets gives

$$a_{ii}e_i = -\sum_{j \in C_i} a_{ij}e_j - \sum_{j \in F_i} a_{ij}e_j - \sum_{j \in N_i} a_{ij}e_j \tag{3.4}$$

C_i: C-points strongly connected to i

F_i: F-points strongly connected to i

N_i: all points weakly connected to i

Define C_i as the set of interpolatory points. Then the goal is to express these points in terms of the points in the sets of F_i and N_i. Returning to the 2D FE Laplacian example then N_i is simply just the empty set. Therefore the aim is to come up with an interpolation that maps the C-Points to the F-points. This is called collapsing the stencil. Again, we will be doing this in a pictorial way as it is easier to visualise (Figure 8).

After one has produced the interpolation stencil we have all the ingredients needed for multigrid. We simply define P^{T} from the coarse set of variables and apply either a two-grid cycle or V-cycle depending the prob-

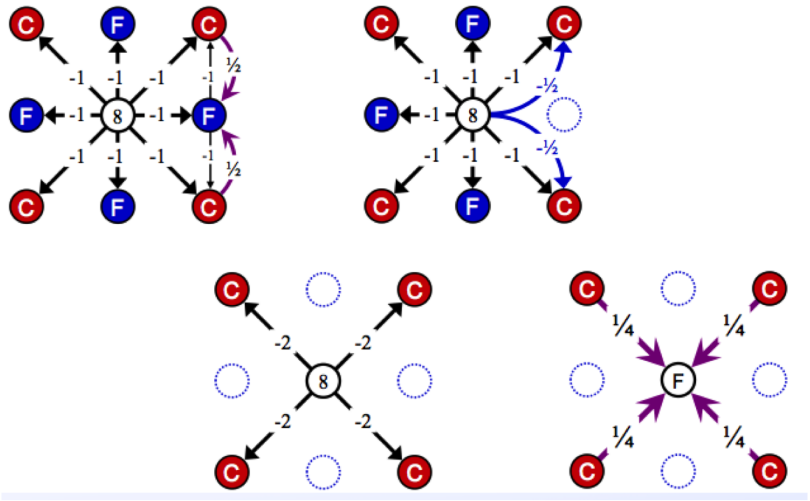


Figure 8: Collapsing the stencil from [Falgout, 2006]

lem.

The method that we have gone through here is the classical Ruge-Stüben AMG. However, there are many variations on this method that could suit a particular problem better. The other main class of AMG is known as smoothed aggregation AMG. This method follows the same structure as CAMG but forms aggregates when forming the coarse grids. This generally leads to coarse grids with smaller dimension for each level of the multigrid V-cycle.

4 Stokes Equations

To illustrate how effective AMG can be, we look at Stokes equations given in (4.1)

$$\begin{aligned} -\nu\Delta\vec{u} + \nabla p &= \vec{f}_s, \\ \nabla \cdot \vec{u} &= 0. \end{aligned} \tag{4.1}$$

Here we define $\vec{u}$, p and ν as the velocity vector, pressure and dynamic viscosity of the fluid respectively. At the moment we will not be applying AMG as a solver to this system but instead we will be applying a V-cycle

as the preconditioner.

4.1 Finite element discretisation

A weak formulation of the Stokes system (4.1) is given by

$$\begin{aligned} \int_\Omega \vec{v} \cdot \big(-\nu \Delta \vec{u} + \nabla p \big) &= \int_\Omega \vec{v} \cdot \vec{f}_{\text{s}}, \\ \int_\Omega q \nabla \cdot \vec{u} &= 0. \end{aligned} \tag{4.2}$$

where $\vec{v}$ and q are test functions choosen from a particular function space. By using vector identities and integration by parts (4.2) can be simplified to

$$\begin{aligned} \int_\Omega \nu \nabla \vec{v} : \nabla \vec{u} - p \nabla \cdot \vec{v} &= \int_\Omega \vec{v} \cdot \vec{f}_{\text{s}} + \int_{\partial\Omega} \vec{g} \cdot \vec{v}, \\ \int_\Omega q \nabla \cdot \vec{u} &= 0. \end{aligned} \tag{4.3}$$

For the test problem we will be considering purely Dirichlet boundary conditions, hence, $\vec{g} = 0$. Also, the notation $\nabla \vec{v} : \nabla \vec{u}$ represents the componentwise scalar product. For this example we will consider using Taylor-Hood elements: the function spaces are

$$\begin{aligned} \mathcal{V} &= \{ \vec{v} \in H_1^2(\Omega) : \vec{v} = \vec{v}_0 \text{ on } \partial\Omega \}, \\ \mathcal{Q} &= \{ q \in H_1(\Omega) \}, \end{aligned} \tag{4.4}$$

where $\vec{u}, \vec{v} \in \mathcal{V}$, $p, q \in \mathcal{Q}$ and $\vec{v}_0$ is the Dirichlet boundary condition. For Stokes it is not necessary to use $q \in H_1$ as $q \in L_2$ forms a stable discretisation, however, when using Taylor-Hood elements we do in fact use $q \in H_1$. Discretisation of (4.3) using the subspaces of function spaces (4.4) yields the following saddle point system

$$\mathcal{A}x = \begin{bmatrix} A & B^{\text{T}} \\ B & 0 \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} f \\ 0 \end{bmatrix} = b. \tag{4.5}$$

4.2 Preconditioning

In this section we will consider preconditioning Stokes equations. When applying a preconditioner to a linear system, $\mathcal{A}x = b$, we need to bear in mind the following two conditions:

1. the preconditioner $\mathcal{P}$ must approximate $\mathcal{A}$,
2. equations with $\mathcal{P}$ should be much easier to solve than equations with $\mathcal{A}$.

A purely theoretical example of an excellent preconditioner is to take $\mathcal{P} = \mathcal{A}$. Then the iterative solution will converge in exactly one iteration. However, taking the preconditioner to be the same matrix is far too expensive. On the other hand, if we take $\mathcal{P}$ to be the identity matrix then we have a very simple system to solve at each step but the iterative scheme will converge at the same rate as a non-preconditioned system. Therefore, we would like to find a preconditioner that is a happy median between easiness to solve and approximation of the original matrix. The following sections outline what is known as the Schur complement preconditioner and an approximation of the Schur complement preconditioner.

4.2.1 Schur complement

If we consider any non-symmetric invertible saddle matrix

$$\mathcal{K}_{\text{ns}} = \begin{bmatrix} A & B \\ C & 0 \end{bmatrix},$$

where A is non-singular then we can perform a block Gaussian elimination ($\mathcal{K}_{\text{ns}} = LDU$) to get the following block decomposition:

$$\begin{bmatrix} A & B \\ C & 0 \end{bmatrix} = \begin{bmatrix} I & 0 \\ CA^{-1} & I \end{bmatrix} \begin{bmatrix} A & 0 \\ 0 & -CA^{-1}B \end{bmatrix} \begin{bmatrix} I & A^{-1}B \\ 0 & I \end{bmatrix}.$$

This decomposition suggests that taking D to be the preconditioner may be effective. In fact, if we do exactly this we can prove [Murphy et al., 2000]

that the preconditioned matrix $\mathcal{T} = D^{-1}\mathcal{K}_{\text{ns}}$ has exactly three eigenvalues and hence D is an excellent preconditioner.

4.2.2 Approximation of Schur complement

In section 4.2.1 we looked at taking a preconditioner of the form,

$$\begin{bmatrix} A & 0 \\ 0 & S \end{bmatrix}, \tag{4.6}$$

where A was the original $(1,1)$ block of your matrix and S was the Schur complement, then an appropriate Krylov iterative scheme will converge in exactly 3 iterations. For the Stokes case, A is the discrete vector Laplacian matrix and $S = BA^{-1}B^{\text{T}}$ where B is the discrete divergence matrix. Therefore (4.6) would be an extremely effective preconditioner for a Krylov iterative scheme. Unfortunately, forming the Schur complement of a system is often computationally expensive and may form a dense matrix. This generally means that we would not like to form S exactly but would like to find a good approximation to it.

Defining $\tilde{S}$ as an approximation to the Schur complement then taking $\tilde{S} = M$ where M is the pressure mass matrix is a very effective approximation. This is shown through inf-sup analysis (for the $2D$ case) to get

$$\gamma^2 \leq \frac{\langle BA^{-1}B^{\text{T}}q, q\rangle}{\langle Mq, q\rangle} \leq 2 \tag{4.7}$$

where γ does not depend on the mesh size. This means that the mass and the Schur complement matrices are spectrally equivalent. For the full analysis see ([Elman et al., 2005] - chapter 6)

4.3 Numerical Results

To discretise this problem we will use the FEniCS finite element package [Logg et al., 2012]. Also, we will use PETSc's [Balay et al., 2013] iterative solvers together with BoomerAMG [Henson and Yang, 2002] from Hypre as

our AMG solver. Using a combination of software packages allows us to test fairly large problems with relative ease.

Unfortunately I have not been able to work out how to use the PETSc `fieldsplit` command (to allow us to apply different solving method to the (1,1) and (2,2) blocks of the preconditioner) before the end of this project. This means that we will consider using an AMG V-cycle as a preconditioner for both the (1,1) and (2,2) blocks. However this is a bit overkill since the pressure mass matrix is extremely well conditioned (it takes only a few iterations of CG to converge with a Jacobi preconditioner).

The kind of mesh that we tested our code on is given in figure 9. The refinement in the middle of the mesh could come from the fact that we know there is a discontinuity in this region. We therefore would like to refine around this area so that we get better accuracy.

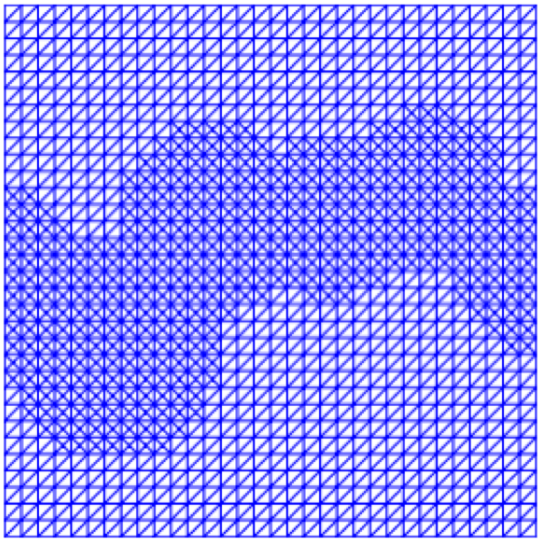


Figure 9: Unstructured mesh

System (4.5) is symmetric but indefinite, and therefore the appropriate Krylov method to solve this system would be MINRES [Paige and Saunders,

1975]. At each iteration of MINRES an AMG V-cycle is applied to

$$\begin{bmatrix} A & 0 \\ 0 & M \end{bmatrix},$$

as the preconditioner. Table 3 shows the iteration numbers and solution times for several unstructured meshes.

Total DoF	V DoF	Q DoF	# iters	Soln Time	$\|\|u_e - u_h\|\|_2$	V-order	$\|\|p_e - p_h\|\|_2$	P-order
86	74	12	26	0.00	1.59e+00	0.00	1.36e+01	0.00
232	202	30	47	0.01	2.35e-01	2.76	2.84e+00	2.26
884	778	106	52	0.02	2.73e-02	3.11	5.91e-01	2.27
3367	2978	389	55	0.10	3.40e-03	3.00	1.43e-01	2.05
13076	11594	1482	57	0.43	4.27e-04	3.00	3.55e-02	2.01
51313	45554	5759	56	1.86	5.34e-05	3.00	8.91e-03	1.99
2034862	180762	22724	58	8.79	6.67e-06	3.00	2.23e-03	2.00
8105652	720274	90291	58	37.50	8.34e-07	3.00	5.59e-04	2.00
32350072	2875106	359901	58	177.84	1.05e-07	2.99	1.40e-04	2.00

Table 3: Stokes iteration and timing results ($2D$)

It is clear to see from table 3 ($2D$ results) that as the total degrees of freedom (the dimension of the matrix) increases, the number of iterations to compute the solution does not change. One can also note that the time taken to compute the solution increases roughly by a factor of 4 (linearly with respect to the matrix size). This is also a very important property of any multigrid solver. We can also see that we get the same sort of behavior from table 4, which is the $3D$ results.

5 Conclusion

In this project, we have looked into AMG preconditioning with a Stokes's problem on an unstructured grid.

When looking at large systems of equations it is important to consider methods that are scalable. We started by considering a geometric approach to multigrid. Applying GMG to problems that are discretisations on struc-

Total DoF	V DoF	Q DoF	# iters	Soln Time
114	105	9	22	0.02
477	447	30	56	0.03
2587	2451	136	80	0.28
17043	16251	792	77	3.43
124074	118707	5367	76	41.66
941837	902619	39218	77	575.14

Table 4: Stokes iteration and timing results ($3D$)

tured grids can perform extremely well (see table 2). This method relies on the fact that there is an underlying grid structure that restriction and prolongation operators can be easily defined. If however one is considering an unstructured grid then it is harder to define restriction and prolongation operators that are determined from a mesh. Therefore an algebraic way to construct these operators is generally preferable.

Looking solely at the matrix coefficients an algorithm was proposed by Ruge-Stüben in the 80's to form restriction and prolongation operators without knowledge of the underling mesh. The main part of this project was to understand and describe the process in which classic AMG was first derived. In section 3 we go through the main step in which classical AMG can be implemented.

The final part of this project looked a specific example in which AMG can be applied. The example that we looked into was solving Stokes flow on an unstructured mesh. First outlining the basic principles of preconditioning, then considered these techniques with the Stokes equations.

In conclusion, using AMG as a preconditioner can be very effective on unstructured grids.

References

[Balay et al., 2013] Balay, S., Brown, J., Buschelman, K., Gropp, W. D., Kaushik, D., Knepley, M. G., McInnes, L. C., Smith, B. F., and Zhang, H. (2013). PETSc Web page. http://www.mcs.anl.gov/petsc.

[Brandt, 1986] Brandt, A. (1986). Algebraic multigrid theory: The symmetric case. *Applied Mathematics and Computation*, 19(1):23–56.

[Brandt et al., 1985] Brandt, A., McCormick, S., and Ruge, J. (1985). Algebraic multigrid (AMG) for sparse matrix equations. *Sparsity and its Applications*, page 257.

[Elman et al., 2005] Elman, H. C., Silvester, D. J., and Wathen, A. J. (2005). *Finite Elements and Fast Iterative Solvers: with Applications in Incompressible Fluid Dynamics: with Applications in Incompressible Fluid Dynamics*. Oxford University Press.

[Falgout, 2006] Falgout, R. D. (2006). An introduction to algebraic multigrid. *Computing in Science and Engineering*, 8(6):24–33.

[Henson and Yang, 2002] Henson, V. E. and Yang, U. M. (2002). ¡ i¿ boomeramg¡/i¿: A parallel algebraic multigrid solver and preconditioner. *Applied Numerical Mathematics*, 41(1):155–177.

[Logg et al., 2012] Logg, A., Mardal, K.-A., and Wells, G. (2012). *Automated solution of differential equations by the finite element method: The fenics book*, volume 84. Springer.

[Murphy et al., 2000] Murphy, M. F., Golub, G. H., and Wathen, A. J. (2000). A note on preconditioning for indefinite linear systems. *SIAM Journal on Scientific Computing*, 21(6):1969–1972.

[Paige and Saunders, 1975] Paige, C. C. and Saunders, M. A. (1975). Solution of sparse indefinite systems of linear equations. *SIAM Journal on Numerical Analysis*, 12(4):617–629.

[Ruge and Stüben, 1987] Ruge, J. and Stüben, K. (1987). Algebraic multigrid. *Multigrid methods*, 3:73–130.