INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING
Int. J. Numer. Meth. Engng 2010; **82**:64–98
Published online 23 October 2009 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/nme.2758

HSL_MI20: An efficient AMG preconditioner for finite element problems in 3D

Jonathan Boyle[1], Milan Mihajlović[2] and Jennifer Scott[3,*,†]

[1]*School of Mathematics, University of Manchester, Alan Turing Building, Manchester M13 9PL, England, U.K.*
[2]*School of Computer Science, University of Manchester, Kilburn Building, Manchester M13 9PL, England, U.K.*
[3]*Computational Science and Engineering Department, Rutherford Appleton Laboratory, Chilton, Oxfordshire, OX11 0QX, England, U.K.*

SUMMARY

Algebraic multigrid (AMG) is one of the most effective iterative methods for the solution of large, sparse linear systems obtained from the discretization of second-order scalar elliptic self-adjoint partial differential equations. It can also be used as a preconditioner for Krylov subspace methods. In this communication, we report on the design and development of a robust, effective and portable Fortran 95 implementation of the classical Ruge–Stüben AMG, which is available as package HSL_MI20 within the HSL mathematical software library. The routine can be used as a 'black-box' preconditioner, but it also offers the user a range of options and parameters. Proper tuning of these parameters for a particular application can significantly enhance the performance of an AMG-preconditioned Krylov solver. This is illustrated using a number of examples arising in the unstructured finite element discretization of the diffusion, the convection–diffusion, and the Stokes equations, as well as transient thermal convection problems associated with the Boussinesq approximation of the Navier–Stokes equations in 3D. Copyright © 2009 John Wiley & Sons, Ltd.

Received 1 October 2008; Revised 7 July 2009; Accepted 24 August 2009

KEY WORDS: large sparse linear systems; Krylov subspace methods; preconditioning; algebraic multigrid; Fortran 95

1. INTRODUCTION

Mathematical models of important physical systems and phenomena are often expressed in terms of differential equations (DEs). Numerical algorithms for the solution of DEs generally assume some form of discretization of a continuous problem, resulting in a system of linear or non-linear

*Correspondence to: Jennifer Scott, Computational Science and Engineering Department, Rutherford Appleton Laboratory, Chilton, Oxfordshire, OX11 0QX, England, U.K.
†E-mail: jennifer.scott@stfc.ac.uk

Contract/grant sponsor: EPSRC; contract/grant number: EP/C000528/1
Contract/grant sponsor: EPSRC; contract/grant number: GR/S42170

equations. In most cases, the coefficient matrices are large, sparse and ill-conditioned (see, for example [1, p. 57, 148]). Research into the development of efficient algorithms for both the direct and iterative solution of such systems has resulted in a host of robust and effective methods (an introduction is provided by the standard texts [2, 3]). One of the desirable properties of a linear solver is optimal scaling of the execution time and requested computational resources with the discrete problem size, irrespective of other problem parameters. Modern sparse direct solvers are not capable of achieving optimal scaling; in particular, the memory and operation count for a direct solver generally increase much more rapidly than the system size. This limits the suitability of direct solvers for solving the linear systems obtained from the discretization of partial differential equations (PDEs) in three spatial dimensions.

Iterative solution techniques, such as Krylov subspace methods [3, Chapter 7], have the optimal storage requirements and the computational cost per iteration, thus offering the potential to be optimal solvers. Unfortunately, the total number of iterations for convergence is normally not known in advance and can potentially be very large (it is determined by the spectral properties of the system coefficient matrix and is bounded by the size of the linear system). In such cases, the optimal property of the solver is lost, although it may be achieved (at least, approximately) if the system is appropriately preconditioned. A good preconditioner [1, Section 2.2], [3, Chapter 10], [4, Section 9.2] is a matrix (or linear operator) that is spectrally close to the system coefficient matrix, but is computationally cheap to assemble and to apply the action of its inverse to a vector. Development of optimal preconditioners is inevitably problem-specific; they have been developed for a wide range of important problems in structural mechanics [5, 6], fluid mechanics [1, 7] and electromagnetism [8, 9].

An important class of optimal iterative solvers for the linear systems that are obtained from common discretizations of scalar, elliptic, self-adjoint PDEs is based on multigrid (MG) methods [10, 11]. In the sequel, we assume that a PDE is discretized by a suitable method, such as the finite element method (FEM) or finite volume method, on a given partitioning of the continuous domain (referred to as the grid), resulting in a system of linear equations. The dimension of this system is proportional to the number of points (nodes) in the grid, with each discrete unknown associated with a grid point. MG exploits the observation that simple iterative solvers, such as Gauss–Seidel (GS) or damped Jacobi (DJ) [3, Chapter 4] (which are also referred to as relaxation methods or smoothers) are capable of reducing only certain components of the solution error, namely the high-frequency (oscillatory) components [10, Chapter 2]. Such behaviour is a consequence of their local nature—the update of each unknown is influenced by only a very few neighbouring unknowns. On the other hand, the effective reduction of slowly-changing (low-frequency or smooth) components of the solution error requires global communication between all the unknowns in the system. Smooth error components will become progressively more oscillatory if represented on grids that are coarser than the grid on which the problem is discretized. Thus, the basic concept of geometric multigrid (GMG) is to reduce, by the application of a relaxation method, the solution error components that are regarded as high-frequency on the current grid, and then to project the remaining error to a coarser grid. On this grid, some of the error components will be further reduced by relaxation, before another projection to an even coarser grid. The implementation of GMG requires knowledge of the geometric information for a nested hierarchy of progressively coarser grids, the discrete representation of the problem on these grids, and the means of communicating the information between the grids. These requirements are frequently difficult to meet. For example, finite element modelling of realistic problems in industrial applications involves complex domain geometries, often with internal boundaries. In such cases, non-structured grids, or composite

DOI: 10.1002/nme

adaptively refined grids, are used for which it is not possible or convenient to generate a nested sequence of coarser grids [10, p. 172]. In an attempt to overcome these difficulties while retaining the benefits of the mulitgrid concept, algebraic multigrid (AMG) was introduced [12]. In AMG, a sequence of successively coarser (smaller) problems is generated by an automatic coarsening procedure. The heuristics of this procedure are based on the properties of discrete scalar, elliptic, self-adjoint operators and no grid information is needed. This extends considerably the application of the multigrid approach and even allows its use on the problems that are not directly associated with a grid. There are a number of different coarsening strategies used in this context including the classical Ruge–Stüben coarsening and its variants [13], the smoothed aggregation (SA) AMG [14], and the agglomeration AMG [15, 16]. There are generalizations of the Ruge–Stüben coarsening procedure aimed at the solution of systems of PDEs (see Section 2.4 for more details). The classical Ruge–Stüben coarsening procedure can be parallelized using domain decomposition [17].

The purpose of this communication is twofold: to introduce to a wider engineering community the new code `HSL_MI20` in the mathematical software library HSL [18] and to demonstrate how a proper selection of the parameters within `HSL_MI20` leads to highly effective and robust AMG-preconditioned Krylov solvers for scalar elliptic problems in 3D and AMG-based block preconditioners for systems of PDEs that arise in fluid mechanics (the Stokes problem and the simulations of the transient thermal convection problem—the Boussinesq problem). `HSL_MI20` computes an AMG preconditioner using the classical Ruge–Stüben coarsening procedure.

The paper is organized as follows. In Section 2, we introduce the basics of the classical AMG method. Section 3 describes the main features of the code `HSL_MI20`. Section 4 reports numerical results for Krylov solvers preconditioned by AMG (using `HSL_MI20`) when applied to discrete diffusion and convection–diffusion problems in 3D and, within a block preconditioning framework, to the Stokes problem in 3D and the transient Boussinesq equations in 3D. The problems are discretized by the FEM using unstructured tetrahedral grids. For a standard model problem, we present a brief performance comparison between `HSL_MI20` and the well-established AMG codes BoomerAMG [19] (a part of the `hypre` library [20]) and ML (a part of Trilinos library [21]). Finally, in Section 5, we summarize our findings.

2. ALGEBRAIC MULTIGRID (AMG)

In this section, we present a brief review of the principles and techniques deployed in the classical Ruge–Stüben AMG method [22]; this approach is implemented in our new code `HSL_MI20`. An essential ingredient of any GMG algorithm is the existence of a hierarchy of nested grids [23]. Such a hierarchy can be obtained by starting from an initial coarse grid, and by adding successively finer discretization levels (for example, by using nested or non-nested global and local mesh refinement). However, many practical problems are defined on domains with complex geometries and internal boundaries, or have solutions that exhibit small-scale phenomena, such as shocks, singularities, and boundary layers. In such cases, an unstructured mesh or a set of composite block-structured meshes [10, p. 172] are required to resolve all features of the domain and the problem solution. Thus, creating a nested sequence of progressively coarser grids is difficult (if, indeed, it is possible at all). Moreover, the book-keeping of the geometric information associated with such a grid hierarchy requires sophisticated data structures and is memory consuming. This may adversely impact the performance of the software that implements them. In addition, some practical problems result in large sparse linear systems that are not associated with mesh discretizations. To apply

DOI: 10.1002/nme

MG principles, a suitable 'grid' coarsening technique is needed that does not explicitly use any geometric information. At the heart of AMG is a sequence of progressively smaller (coarser) representations of the linear system coefficient matrix. Such a technique is based on heuristics, developed from purely algebraic relations between the unknowns in the system matrix. The AMG algorithm requires certain components, namely, given the discrete operator $A=A_1$ on the finest level, a sequence of coarse level operators A_ℓ, $\ell=2,\ldots,L$ is required ($L-1$ is the number of coarse levels), together with a relaxation method, a sequence of intergrid transfer operators $P_\ell^{\ell+1}$ and $P_{\ell+1}^{\ell}$, and a solver for the linear problem at the coarsest level L. Construction of these components is based on two fundamental heuristics: *algebraically smooth errors* and *strength of dependence* principle.

2.1. Two fundamental AMG principles

Consider the linear system

$$A\mathbf{x}=\mathbf{b} \tag{1}$$

where $A=\{a_{ij}\}\in\mathbb{R}^{n\times n}$ is the coefficient matrix, $\mathbf{x}=\{x_i\}\in\mathbb{R}^n$ is the vector of unknowns, and $\mathbf{b}=\{b_i\}\in\mathbb{R}^n$ is the given right-hand side vector. If the computed solution is denoted by $\tilde{\mathbf{x}}$, we can define the *solution error* to be $\mathbf{e}=\mathbf{x}-\tilde{\mathbf{x}}$ and the *residual* to be $\mathbf{r}=\mathbf{b}-A\tilde{\mathbf{x}}$. The solution error and the residual are connected by the residual equation

$$A\mathbf{e}=\mathbf{r} \tag{2}$$

The effectiveness of MG in reducing the solution error is based on two complementary numerical procedures: fixed-point iteration (also called the smoothing or relaxation method), and the coarse grid correction. If the solution error is decomposed into discrete Fourier components [10, Chapter 2], it can be shown that a fixed-point iteration is effective in reducing its high-frequency (oscillatory) components, while the coarse grid correction reduces the low-frequency components. In GMG, the solution error obtained after the application of a few steps of a fixed-point iteration (smoothing) algorithm looks geometrically smooth, and coarse grids are selected with the aim of representing this error with sufficient accuracy. In AMG, the underlying grid is not explicitly known, and thus Fourier analysis cannot be used to define smoothness of the solution error. Instead, we first select a relaxation procedure and then define the *algebraically smooth error* to be the error that cannot be reduced efficiently by relaxation. Note that the algebraically smooth error may not be a smooth function in a geometric sense (see [10, p. 140]). The algebraically smooth error is characterized by small residuals, that is,

$$A\mathbf{e}\simeq\mathbf{0} \tag{3}$$

Another important property of the algebraically smooth error is that its value at any point can be interpolated accurately by a weighted average of its neighbours (the neighbouring unknowns of a given unknown x_i are defined by the non-zero entries a_{ij}, $j\neq i$ of the ith row in the matrix A). This property is relevant in defining the interpolation procedure. To reduce smooth errors further, they need to be represented by a smaller (coarse level) defect equation of the form (2), that is cheaper to solve than the original problem (1). The coarse level correction is the process of transferring the residual $\mathbf{r}$ from a fine to a coarse level, solving the defect equation (exactly or approximately) and projecting (interpolating) the correction $\mathbf{e}$ back to a fine level. The whole

DOI: 10.1002/nme

process can be applied recursively by introducing a hierarchy of coarse levels and by solving the defect equation approximately on each level (on the coarsest one, it may be solved exactly).

In the linear system (1), each equation i represents a constraint for uniquely determining the unknown x_i. Although all the equations are needed to compute the x_i exactly, it is possible to determine from the ith equation a set of the neighbouring unknowns x_j $(j \neq i)$ that most influence the unknown x_i. Heuristically, these are the unknowns x_j for which the absolute value of the corresponding entries a_{ij} are large compared with the other off-diagonal entries in the ith equation. Such unknowns x_j are good candidates for determining the value x_i by interpolation, that is, they are the candidates for coarse level points. This heuristical argument can be formalized by defining the unknown x_i to be strongly connected to all unknowns x_j for which

$$-a_{ij} \geqslant \theta \max_{k \neq i} \{-a_{ik}\} \quad (0 < \theta \leqslant 1) \tag{4}$$

i.e. x_i is strongly connected to all unknowns whose entries a_{ij} are of comparable magnitude with the largest off-diagonal entry in the i-th equation. Otherwise, x_i is considered to be weakly connected to x_j. Note that in the expression (4) it is assumed that all the off-diagonal entries in A are negative, i.e. A is an M-matrix [3, p. 28]. If x_i is strongly (respectively, weakly) connected to x_j, then x_j is said to strongly (respectively, weakly) influence x_i. The parameter θ in (4) is referred to as the *strength of dependence threshold*, and (4) is a formal expression of the *strength of dependence* principle. An important property of the smooth error is that it varies slowly along the strong connections.

2.2. The interpolation operator

Let us assume that the unknowns x_i have been partitioned into n_C coarse (C) and n_F fine (F) level unknowns. With this partition it is assumed that the C unknowns will form the next coarse level, and the F unknowns will have their values interpolated from the C unknowns. We define a neighbourhood of each fine level unknown $x_i \in F$ as the set of unknowns $N_i = \{x_j \mid j \neq i,\ a_{ij} \neq 0\}$. Each unknown belonging to N_i is either a coarse level unknown that strongly influences x_i (C_i), a fine level unknown that strongly influences x_i (D_i^{s}), or either a C or F unknown that weakly influences x_i (D_i^{w}). Thus

$$N_i = C_i \cup D_i^{\mathrm{s}} \cup D_i^{\mathrm{w}} \tag{5}$$

The interpolation operator $P \in \mathbb{R}^{n_F \times n_C}$ is then componentwise defined as follows:

$$(P\mathbf{e})_i = \begin{cases} e_i & \text{if } x_i \in C \\ \displaystyle\sum_{j \in C_i} w_{ij} e_j & \text{if } x_i \in F \end{cases} \tag{6}$$

where w_{ij} are the unknown interpolation weights. These weights are obtained from Equation (3), which can be expressed componentwise as

$$a_{ii} e_i \simeq -\sum_{j \in N_i} a_{ij} e_j \tag{7}$$

DOI: 10.1002/nme

Using (5), and adding the contributions from D_i^{w} to the diagonal terms (see [10, p. 143] for more details), yields

$$w_{ij} = -\frac{a_{ij} + \sum_{m \in D_i^{\mathrm{s}}} (a_{im} a_{mj} / \sum_{k \in C_i} a_{mk})}{a_{ii} + \sum_{n \in D_i^{w}} a_{in}} \tag{8}$$

The choice (8) of interpolation weights w_{ij} ensures that constant functions are interpolated exactly (the matrix P has its row sums equal to 1). The approximations involved in the derivation of the formula (8) assume that if x_i and x_j are strongly connected F unknowns, there must be at least one common point in their respective coarse sets C_i and C_j (see [10, pp. 143–144] for details).

The restriction operator R in AMG is commonly selected to be the transpose of the interpolation operator, that is, $R = P^{\mathrm{T}}$.

2.3. Coarse level operators

Once the interpolation operators $P_{\ell-1}^{\ell}$ between each pair of levels $\ell - 1$ and ℓ have been assembled, the coarse level operators (matrices) A_ℓ, $\ell = 2, \ldots, L$ $(A_1 = A)$, are constructed recursively using the Galerkin projection

$$A_\ell = (P_{\ell-1}^{\ell})^{\mathrm{T}} A_{\ell-1} P_{\ell-1}^{\ell} \tag{9}$$

As the interpolation matrix $P_{\ell-1}^{\ell}$ is of full rank, any properties of the matrix $A_{\ell-1}$ (such as symmetry, positive definiteness) will be inherited by its coarse representation A_ℓ [3, p. 423]. Note, however, that this feature applies only when the coefficient matrix A arises in discretizations of scalar PDEs and does not extend to PDE systems.

2.4. The coarsening procedure

The process of assembling the interpolation operators assumes the partitioning of the $\{x_i\}$ into coarse and fine level unknowns. The criteria for this partitioning are based on the AMG principles of smooth error and strength of dependence. In GMG, the choice of coarse grids is influenced by the following (mutually exclusive) requirements: (i) smooth errors should be approximated accurately, (ii) smooth functions should be interpolated from coarse to fine grids accurately, and (iii) the size of each coarse grid should be considerably smaller than that of the preceding grid to ensure computational efficiency.

We denote by S_i the set of unknowns that strongly influence x_i, and by S_i^{T} the set of unknowns that are strongly connected to x_i, based on the heuristics (4). The AMG coarsening procedure for determining the sets C and F (introduced in Section 2) is based on two heuristic criteria:

- **H1**. For each $x_i \in F_i$, each unknown $x_j \in S_i$ should either be in C_i or be strongly connected to at least one unknown in C_i.
- **H2**. The set C should be a maximal subset of all unknowns with the property that no two unknowns from it are strongly connected to each other.

To make the approximations that lead to the interpolation formula (8), each unknown $x_j \in D_i^{\mathrm{s}}$ must be strongly connected to at least one unknown in C_i; heuristic **H1** ensures this. In this way, the unknowns $x_j \in D_i^{\mathrm{s}}$ that are not chosen to be C unknowns, are represented in the interpolation formula (8) through their values that are distributed to the points in C_i. Heuristic **H2** ensures the balance in sizes of coarse levels. If the number of C unknowns is not much smaller than the number

DOI: 10.1002/nme

of unknowns on the fine level, the interpolation is likely to be accurate and AMG will be more effective, but at the expense of increased computational cost. Heuristic **H2** controls the size of the coarse levels by requiring that there is no strong dependence between C unknowns, thus keeping the C unknowns at sufficient 'distance' from each other. On the other hand, the requirement that the set C must be a maximal subset ensures the quality of the interpolation.

The requirements **H1** and **H2** are usually mutually exclusive. In practice, **H1** is normally enforced strictly, while **H2** is satisfied approximately (the usual tradeoff between accuracy and efficiency). This approach, however, leads to coarse levels that are often larger than necessary. For 2D problems this is usually beneficial, but for 3D problems discretized by a finite element or finite difference method, a weak enforcement of **H2** may lead to a considerable increase in the size of coarse levels and, consequently, the execution time. A recent study [24] reveals that coarse levels with lower complexities (i.e. with a smaller number of unknowns than that obtained by strictly enforcing **H1**) may still produce good convergence rates when AMG is used as a preconditioner, rather than as a solver. We found the same trend in our numerical studies for scalar elliptic problems in 3D (see results given in Section 4). This trend extends to case studies of PDE systems.

The standard coarsening algorithm uses two passes of the data. In the first pass, a preliminary partitioning into C and F unknowns is done with the aim of enforcing **H2**, while retaining the interpolation quality. In the second, some of the initial F unknowns are reassigned as C unknowns to strictly enforce **H1** (and thus violate **H2**). If the second pass is not performed, the coarsening strategy is referred to as one pass coarsening (RS1); otherwise, we have two pass coarsening (RS2). The effectiveness of both strategies is evaluated in Section 4.

Recently, generalizations of the standard Ruge–Stüben coarsening techniques have been introduced for applying AMG to PDE systems. The classical AMG coarsening procedure is based on the *variable approach*, in which all the unknowns are treated equally. Unless there is a very weak coupling between unknowns, this does not work effectively for PDE systems and some modifications and extensions are needed to achieve good convergence and robustness (see [25]). There exists several generalizations of the variable-based approach. The *unknown-based approach*, where different types of unknowns are treated separately, is the simplest way of generalizing AMG for PDE systems. This works effectively if diagonal blocks are close to M-matrices (e.g. for systems of non-scalar PDEs, such as linear elasticity [26]). Further generalizations (the *point-based approach*) were introduced in the code SAMG [27]. Here all the unknowns share the same grid hierarchy, and the coarsening process is based on an auxiliary matrix, referred to as the primary matrix. This matrix should be representative of the connectivity patterns of all the unknowns in the system. Although substantial progress has been made in developing AMG for systems of PDEs, there is still no single technique that works universally well for a wide class of problems. Some concrete implementations, however, perform well for certain important applications, such as fluid mechanics, multiphase flow in porous media [28], structural mechanics, and semiconductor processes and device simulation [29, 30].

2.5. The algorithm complexity

Multigrid methods are attractive as preconditioners because, when used in conjunction with Krylov solvers, they aim to solve large sparse linear systems of order n in $\mathcal{O}(n)$ time, thus exhibiting a (near-optimal) scaling. This optimality is achieved through a (nearly) constant number of iterations being needed to obtain the prescribed accuracy, regardless of the problem size or any other problem parameters, and by the optimal computational cost of each iteration. Larger than necessary

DOI: 10.1002/nme

coarse level operators increase the computational cost of AMG iterations and, in extreme cases, can undermine the algorithm optimal scaling property.

To enable us to monitor how the choice of the coarsening strategy (RS1 or RS2) and the strength of dependence threshold θ from (4) affect the quality of coarsening and the size of the coarse level operators, we introduce standard quantitative measures: the grid complexity c_G, the operator complexity c_A, and the average stencil size c_S.

The grid complexity c_G is defined as

$$c_G = \frac{1}{n_1} \sum_{\ell=1}^{L} n_\ell \tag{10}$$

where n_ℓ is the size of the matrix A_ℓ ($\ell = 1, \ldots, L$), and A_1 is the fine grid matrix.[‡] If a 3D problem is discretized on a sequence of nested, uniformly refined grids, and a full coarsening is used (the usual case in GMG), then $c_G = \frac{8}{7}$ (see [10, pp. 45–48]). Full coarsening of a uniform grid assumes the reduction of the number of grid points in each spatial direction by a factor of 2. A large discrepancy from this value indicates a strong violation of heuristic **H2**.

The operator complexity c_A is defined as

$$c_A = \frac{1}{\mathrm{nnz}(A_1)} \sum_{\ell=1}^{L} \mathrm{nnz}(A_\ell) \tag{11}$$

where $\mathrm{nnz}(A_\ell)$ is the number of non-zero entries in A_ℓ ($\ell = 1, \ldots, L$). c_A provides an indication of the AMG storage requirements (the storage of the interpolation matrices is not included). In the case of full coarsening of a problem defined on a sequence of nested, uniformly refined grids, and direct discretization of a PDE on each level (the approach usually adopted in GMG), $c_A \simeq c_G$. In AMG, a difference in these two parameters may occur (especially for 3D problems), indicating the coarse level matrices have, on average, much larger stencil sizes (numbers of non-zero elements per row) than the fine grid matrix A_1. It also implies that the interpolation matrices are fairly dense, making the restriction and the interpolation operations more expensive to apply. In addition, the application of GS smoothing becomes more expensive. The value of c_A can be reduced (at the expense of reducing the preconditioner effectiveness) by increasing the strength of dependence threshold θ (see Section 4).

The average stencil size c_S is defined as

$$c_S = \frac{1}{L} \sum_{\ell=1}^{L} \frac{\mathrm{nnz}(A_\ell)}{n_\ell} \tag{12}$$

This quantity should be compared with the average stencil size of the original problem $c_S^{(1)} = \mathrm{nnz}(A_1)/n_1$. Large differences between c_S and $c_S^{(1)}$ indicate that the coarse level matrices have larger stencils than the original matrix. Note that, because of the small contributions from the coarsest levels, c_S may represent an optimistic estimate (see the examples in Section 4).

[‡]Note that here and elsewhere, $A_1 = A$ and $n_1 = n$ is the number of unknowns.

DOI: 10.1002/nme

2.6. AMG solver and preconditioner

When all the components of the AMG algorithm are assembled, the solution phase consists of repeating the V-cycles until a prescribed level of accuracy is achieved. A V-cycle is essentially the same in both GMG and AMG and is described in detail in [10, p. 40–41]. In brief, a V-cycle consists of an approximate solution of the system (1) by a small number (nit_1) of fixed-point iterations, and a projection of the residual to a coarse level ℓ using the restriction operator. At each of these levels $\ell=2,\ldots,L$, the residual Equation (2) is solved (by a fixed point iteration for $\ell=2,\ldots,L-1$ and exactly or approximately for $\ell=L$). The process of approximately solving (1) and (2) by a fixed-point iteration is referred to as *pre-smoothing*. The resulting correction to the solution needs to be projected (interpolated) for $\ell=L,\ldots,1$ using the interpolation operator. This process introduces high-frequency error components. To eliminate them, at each level we need to perform a small number (nit_2) of fixed-point iterations. This process is referred to as *post-smoothing*. A V-cycle with nit_1 pre-smoothing steps and nit_2 post-smoothing steps is symbolically denoted as $\mathrm{V}(nit_1, nit_2)$.

Although our new package `HSL_MI20` can be used as a standalone AMG solver for scalar, elliptic, self-adjoint problems,[§] our primary goal was to develop an efficient 'black-box' preconditioner for scalar elliptic problems, and to allow the use of the package within the block preconditioning framework for more complex systems of PDEs, that can be applied in conjunction with, for example, the Krylov solvers [3, Chapter 6] available within the HSL library. The design of `HSL_MI20` is discussed in the next section.

If using a standard V-cycle, it is usually the case that the AMG-preconditioned Krylov solver performs more robustly than the standalone AMG solver, especially when applied to some non-trivial scalar elliptic problems (such as anisotropic diffusion, diffusion with jumping coefficients [23], convection problems with recirculating convective field, etc.). Note that recently a K-cycle was introduced [31] that aims to improve the robustness of the AMG solver by using Krylov acceleration (usually consisting of a small number of iterations of a Krylov solver) at each coarse level.

3. DESIGN OF `HSL_MI20`

Given a sparse matrix $A\in\mathbb{R}^{n\times n}$ and a vector $\mathbf{z}\in\mathbb{R}^n$, `HSL_MI20` computes the vector $\mathbf{y}=\mathrm{AMG}(A,\mathbf{z})$, where $\mathrm{AMG}(A,\mathbf{z})$ denotes the approximate solution of a linear system $A\mathbf{y}=\mathbf{z}$ by a number of AMG V-cycles. The matrix A (which may be symmetric or unsymmetric) should ideally be 'close' to an M-matrix, that is, it must have positive diagonal entries and ideally (most of) the off-diagonal entries should be negative. Moreover, the diagonal should not be small compared with the sum of the modulii of the off-diagonals [10, p. 138]. Strong violation of these requirements may lead to poor convergence characteristics of the AMG solver/preconditioner. In order to quantify the term 'close to M-matrix', in Section 4 we report the number of positive off-diagonal entries as a percentage of the total number of off-diagonal entries and the magnitude of the largest off-diagonal entry in A.

[§] One needs to perform a simple Richardson iteration of the form $\mathbf{x}^{[k+1]}=\mathbf{x}^{[k]}+\mathrm{AMG}(A,\mathbf{r}^{[k]})$, $k=0,1,\ldots,$ where $\mathrm{AMG}(A,\mathbf{r}^{[k]})$ assumes the approximate solution of the residual Equation (2) by one V-cycle of AMG.

DOI: 10.1002/nme

3.1. The user interface and derived data types

HSL is a Fortran library and HSL_MI20 is written in Fortran 95. One of the main reasons for this choice was because it allows a much simpler user interface than if Fortran 77 was used. A reverse communication interface is used with the following procedures available to the user:

MI20_setup takes the matrix A, optionally checks it for errors (duplicates and out-of-range entries are not allowed), and then generates all the data required by the AMG preconditioner. That is, it selects the coarse levels, builds the interpolation operators, and constructs the coarse level matrices. The matrix A must be supplied in compressed sparse row format (CSR), that is, the non-zero entries in A must be stored row by row.

MI20_precondition performs the preconditioning operation $\mathbf{y} = \mathrm{AMG}(A, \mathbf{z})$, where $\mathbf{z}$ is a user-supplied vector (typically a residual vector if AMG is being used as a preconditioner for a Krylov solver). More than one call to MI20_precondition may be made after a single call to MI20_setup.

MI20_finalize should be called after all other calls to MI20_setup and MI20_precondition are complete for a given problem. It deallocates all array components of the derived data types set up by HSL_MI20.

HSL_MI20 offers the user a large number of options. These are controlled using a derived-type MI20_control. The components of MI20_control are automatically given default values in the definition of the type but may be reset by the user to control certain aspects of the algorithm. In MI20_setup, controls allow the user to choose the maximum number of coarse levels $L_{\max}$ and the maximum size of the coarsest level $(n_L)_{\max}$, and to control the construction of the coarse levels. In MI20_precondition, there are controls to select the method for the solution of the residual equation at the coarsest level, as well as the type of pre- and post-smoother, the numbers of pre- and post-smoothing steps, and the number of V-cycles performed. In addition, the user can specify the amount of diagnostic printing that is required. The default settings for the control parameters have been chosen to make the code robust and, in general, efficient. They are also the values commonly recommended in the literature. If these settings are used, HSL_MI20 provides the user with a 'black-box' preconditioner but, for some practical applications, it may be worthwhile for the user to experiment with different choices of the control parameters (this is illustrated by our results in Section 4). Full details of all the controls used by HSL_MI20 are given in the user documentation.

3.2. MI20_setup

During the setup phase, the coarsening procedure is applied recursively with the aim of producing a sequence of coarse level problems of progressively smaller size. Constructing each level ℓ, $\ell = 2, \ldots, L$ of the AMG hierarchy comprises the following steps:

- Splitting the vertices into fine level (F) vertices and coarse level (C) vertices. The heuristics of this procedure were described in Section 2.4.
- Constructing the interpolation matrix $P_{\ell-1}^{\ell}$ to transfer information between the fine level $\ell-1$ and the coarse level ℓ. This was described in Section 2.2.
- Constructing the coarse level matrix A_ℓ. This was described in Section 2.3.

The quality of coarsening is controlled by two parameters. The real parameter control% st_parameter corresponds to the strength of dependence threshold θ from (4) and has default

DOI: 10.1002/nme

value 0.25. Increasing this parameter produces coarse levels with progressively smaller c_A, c_G and c_S, thus reducing the computational cost of assembling and applying the preconditioner. However, it also affects the quality of the preconditioner and generally leads to an increase in the iteration count. The Boolean parameter `control%one_pass_coarsen` controls the type of the coarsening algorithm. If set to `.TRUE.`, one pass coarsening (RS1) is performed. Otherwise, the standard RS2 coarsening is performed (this is the default). For a given θ, RS2 coarsening typically produces more coarse levels and larger values of c_A, c_G, and c_S and leads to a better quality preconditioner. However, in some cases (especially for 3D problems), RS1 coarsening produces a preconditioner that is computationally cheaper than the one obtained by RS2 coarsening; thus there is a tradeoff between the quality and the computational cost of the preconditioner.

The parameter `control%c_fail` controls the coarsening failure criteria. A value of `1` (the default) indicates that coarsening terminates if *any* row in a coarse level matrix has at least one strictly positive entry but no negative off-diagonal entries. A value of `2` indicates that coarsening terminates if *all* the rows in a coarse level matrix have at least one strictly positive entry and no negative off-diagonal entries or if the lack of negative off-diagonals causes coarsening to fail.

3.3. `MI20_precondition`

After a successful call to `MI20_setup`, `MI20_precondition` may be called repeatedly by the user to perform the preconditioning operations. Options available to the user include the choice of smoother and the coarse level solver.

3.3.1. Choice of smoother. Two different smoothers: DJ and GS are available (the choice is controlled by the parameter `control%smoother`). By default, the damping factor (`control%damping`) for DJ is 0.8. This is the optimal value for the Laplacian in 2D discretized by the FEM using uniform triangular grids (see [11, p. 31]). The optimal choice for the damping parameter for certain regular discretizations of the Laplace operator can be obtained from the analysis of the smoothing and the approximation property of the two-grid scheme introduced by Hackbusch [32] (see also [1, pp. 95–100]).[¶] The default smoother within `HSL_MI20` is GS (`control%smoother=2`) with two pre-smoothing and two post-smoothing iterations (`control%pre_smoothing=2` and `control%post_smoothing=2`). The GS sweep direction is reversed for the post smoothing. If A is symmetric, this choice guarantees symmetry of the preconditioner, and thus the conjugate gradient (CG) method can be used as the Krylov solver. The user may wish to experiment with different settings for the parameters that control smoothing to try and optimize the performance for his or her applications.

3.3.2. Coarse level solver. On the coarsest level, `HSL_MI20` offers the following choice of solvers (controlled by `control%coarse_solver`):

- damped Jacobi;
- Gauss–Seidel;
- sparse direct solver `HSL_MA48`; and
- `LAPACK` dense direct solver `_GETRF`.

[¶]The optimal value for the discrete Laplace operator in 3D (the standard 7 point approximation) is $\omega=\frac{6}{7}$ [11, p. 73], and for 1D model problem, discretized by the central finite differences, $\omega=\frac{2}{3}$ [10, p. 21].

DOI: 10.1002/nme

For robustness, the default is HSL_MA48 [33, 34]. HSL_MA48 is a sophisticated, general-purpose sparse direct solver that is included within the HSL library, but it may be faster to use an iterative solver or, if the coarse level matrix is almost dense, to use the dense solver _GETRF from the LAPACK library [35]. For the Jacobi and Gauss–Seidel solvers, the number of iterations may be selected by the user (control parameter control%coarse_solver_its). Further details of the use of HSL_MA48 within HSL_MI20 are given in [36].

Typically, MI20_precondition will be called a number of times after the call to MI20_setup. We have designed the code to be flexible so that the user does not have to use the same coarse level solver on each call. For example, using the information returned by HSL_MA48, the user may wish to switch to another solver. This is possible without recalling MI20_setup and allows the user to easily experiment with the different options.

4. NUMERICAL EXPERIMENTS

In this section, we present numerical results obtained using the HSL_MI20 AMG preconditioner with Krylov solvers for several classes of problems: the scalar diffusion problem [1, Chapter 1,2], the scalar convection–diffusion problem [1, Chapter 3,4], the Stokes problem [1, Chapter 5,6], and the transient thermal convection problem modelled by the Boussinesq approximation of the Navier–Stokes equations. For the Boussinesq problem, we consider two widely studied practical configurations: a laterally heated cavity (see, for example, [37]), and the Rayleigh–Benard convection (see [38, Chapter 2]). Our test examples are from unstructured FEM discretizations of these problems on non-trivial domains in 3D, which lead to coefficient matrices with highly irregular sparsity patterns. AMG-preconditioned Krylov methods or AMG-based optimal block preconditioners are often used in these situations in preference to both direct sparse solvers and Krylov methods with general algebraic preconditioners (such as the ILU(0) method [3, p. 287]).

The definition of domain geometries and the finite element mesh generation is performed in FEMLAB [39]. The finite element discretizations, performed by the code femFluidMechanics, use linear or quadratic tetrahedral elements, with the standard Galerkin FEM for the diffusion problem [1, p. 17], and the streamline upwinding Petrov–Galerkin (SUPG) FEM with linear elements and an optimal choice of the stabilization parameter (as suggested in [1, p. 126]) deployed in the convection–diffusion examples. For the finite element discretization of the Stokes problem, we deploy both the stabilized equal-order velocity–pressure approximation $P_1 - P_1$ [1, p. 243], with pressure projection stabilization [40], and the uniformly stable $P_2 - P_1$ velocity–pressure approximation [1, p. 229]. The Boussinesq problem is discretized using the stable $P_2 - P_1 - P_2$ velocity-pressure-temperature approximation. The resulting system of differential algebraic equations (DAEs) is solved by the stabilized trapezoidal rule (sTR) algorithm from [41].

All experiments are performed on a PC with a Pentium Core Duo processor with a 2.66 GHz clock and 4 GB of RAM. The g95 Fortran compiler (see g95.sourceforge.net) with optimization flag -O3 is used throughout. The reported times are elapsed times (in seconds) for the setup phase and for the total execution (that is, the time for the setup followed by the solution phase) measured using the Fortran intrinsic function dtime. We use the control parameters control%v_iterations=1, and control%pre_smoothing=control% post_smoothing=1, corresponding to one V(1,1) cycle; otherwise, unless explicitly stated, default values are used for the remaining HSL_MI20 control parameters. We use the default setting of 1 for the parameter control%c_fail for the diffusion problems and use 2 for the

DOI: 10.1002/nme

convection–diffusion problems. The latter choice is because, in the convection–diffusion case, the coarsening can give coarse level matrices that have some rows with negative off-diagonal entries connected to the rows with no negative off-diagonal entries. This causes premature termination of the coarsening.

4.1. The Poisson equation in 3D

We first consider the solution of the linear systems that arise in a Galerkin FEM approximation of the Poisson equation

$$-\nabla^2 u = f \quad \text{in } \Omega \subset \mathbb{R}^3 \tag{13}$$

subject to the boundary conditions

$$u = u_D \quad \text{on } \partial\Omega_D \quad \text{and} \quad \frac{\partial u}{\partial \hat{n}} = u_N \quad \text{on } \partial\Omega_N \tag{14}$$

where $\partial\Omega = \partial\Omega_D \cup \partial\Omega_N$ is the boundary of Ω and $\partial\Omega_D \cap \partial\Omega_N = \emptyset$ ($\partial\Omega_D \neq \emptyset$). Following the discretization procedure described in [1, Chapter 1], we obtain a linear system

$$A\mathbf{x} = \mathbf{b} \tag{15}$$

where A is symmetric and positive definite [1, p. 18]. In this case, the Krylov method of choice is the CG method. In our experiments, we use the HSL implementation of CG (routine `MI21`) and terminate the computation when

$$\|\mathbf{r}^{(k)}\|_2 < \varepsilon * \|\mathbf{r}^{(0)}\|_2 \tag{16}$$

where $\mathbf{x}^{(k)}$ is the computed solution and $\mathbf{r}^{(k)} = \mathbf{b} - A\mathbf{x}^{(k)}$ is the residual vector at the kth iteration, and $\mathbf{r}^{(0)} = \mathbf{b}$ is the initial residual (in all cases we set $\mathbf{x}^{(0)} = \mathbf{0}$). We adopt $\varepsilon = 10^{-6}$ as this is the standard value used in this context [1, p. 77] and we check the forward error $\|\mathbf{x}^{(k)} - \tilde{\mathbf{x}}\|_2$, where $\tilde{\mathbf{x}}$ is the exact solution (if known) or the solution obtained by a direct sparse solver. It should be mentioned that when the Poisson equation is discretized by linear elements, the resulting coefficient matrix is an M-matrix, provided that certain restrictions on the underlying FE grid are met. However, this property does not extend to matrices obtained from higher-order approximations. To establish some quantitative measure of the distance of a coefficient matrix from an M-matrix, we define the quantity η to be

$$\eta = 100 * k_o^p / k_o \tag{17}$$

where k_o and k_o^p are, respectively, the number of non-zero off-diagonal entries and the number of strictly positive off-diagonal entries in A. If $\eta = 0$, A is an M-matrix and if $\eta = 100$, A is a positive matrix.

Example 4.1.1

We start by solving (13) on a cylindrical domain $\Omega = \{x^2 + y^2 \leqslant 1\} \times [0, 5] \subset \mathbb{R}^3$. We construct the problem with a known analytical solution $u = x^2 + y^2 + z^2$ and set the forcing term $\mathbf{f}$ in (13) and the non-homogeneous Dirichlet boundary conditions in (14) accordingly. The FE discretization of this problem is performed by linear elements.

DOI: 10.1002/nme

Table I. The iteration counts and, in brackets, the setup (top) and total (bottom) execution times (in seconds) for the CG method preconditioned using either AMG or ILU(0), applied to the linear system obtained from the linear FEM discretization of the Poisson problem.

n		22 764		69 285		208 965	
	θ	RS2	RS1	RS2	RS1	RS2	RS1
DJ	0.25	8 ($\frac{0.27}{0.40}$)	12 ($\frac{0.09}{\mathbf{0.21}}$)	8($\frac{1.08}{1.57}$)	13 ($\frac{0.30}{\mathbf{0.75}}$)	8 ($\frac{4.16}{5.96}$)	15 ($\frac{1.03}{\mathbf{2.93}}$)
	0.50	8 ($\frac{0.19}{0.33}$)	13 ($\frac{0.08}{0.22}$)	9 ($\frac{0.75}{1.29}$)	15 ($\frac{0.29}{0.84}$)	9 ($\frac{2.67}{4.62}$)	17 ($\frac{1.03}{3.31}$)
	0.67	10 ($\frac{0.14}{0.29}$)	16 ($\frac{0.08}{0.25}$)	11 ($\frac{0.54}{1.13}$)	20 ($\frac{0.27}{1.03}$)	11 ($\frac{1.85}{3.96}$)	25 ($\frac{0.97}{4.38}$)
GS	0.25	8 ($\frac{0.27}{0.47}$)	11 ($\frac{0.09}{\mathbf{0.23}}$)	8 ($\frac{1.08}{1.82}$)	12 ($\frac{0.30}{\mathbf{0.82}}$)	8 ($\frac{4.16}{6.87}$)	13 ($\frac{1.03}{\mathbf{3.06}}$)
	0.50	8 ($\frac{0.19}{0.40}$)	11 ($\frac{0.08}{\mathbf{0.23}}$)	8 ($\frac{0.75}{1.47}$)	14 ($\frac{0.29}{0.94}$)	8 ($\frac{2.67}{5.18}$)	15 ($\frac{1.03}{3.56}$)
	0.67	9 ($\frac{0.14}{0.33}$)	14 ($\frac{0.08}{0.26}$)	10 ($\frac{0.54}{1.29}$)	18 ($\frac{0.27}{1.13}$)	10 ($\frac{1.85}{4.50}$)	23 ($\frac{0.97}{4.91}$)
ILU(0)		49 ($\frac{4.83}{5.02}$)		71 ($\frac{45.7}{46.8}$)		99 ($\frac{445}{450}$)	

For each smoother (DJ or GS), the smallest total times are in bold.

In Table I we present the iteration counts and execution times for the RS1 and RS2 coarsening strategies. For each, we test three values of the strength of dependence threshold θ, and use both the DJ (with `control%damping =0.8`) and GS smoothers. Our experiments with different values of θ and the comparison of the coarsening strategies are motivated by recent results [24], where the authors suggest that, if AMG is used as a preconditioner for 3D problems, it is advantageous to use coarsening strategies that produce relatively sparse coarse level operators. One way of achieving this is to increase θ from its default of 0.25. For comparison, we include the iteration counts and the times for CG using an ILU(0) preconditioner. The ILU(0) factorization is performed using the HSL package `MI11`.

We see that the fastest times are obtained for RS1 coarsening with $\theta = 0.25$ and DJ smoothing. Furthermore, the times using AMG increase by a factor of 4 when the problem size n is increased by a factor 3. Ideally, the time and problem size should scale at the same rate but this does not happen because of the non-optimality of the AMG coarsening procedure for 3D problems on non-structured grids, which results from the coarse level matrices being much denser than the original matrix A (see Table II). Note that, because the Galerkin projection (9) is used to generate the coarse level matrices, this also implies that the interpolation matrices are fairly dense. Furthermore, if GS smoothing is used, its application becomes more expensive at the coarser levels. Denser coarse level matrices adversely affect data caching, slowing the algorithm performance.

It is clear that the ILU(0) preconditioner is far from optimal, with the iteration counts and times growing significantly with n (the asymptotic execution time behaves like $\mathcal{O}(n^2)$). It should be noted, however, that most of the time is spent on computing the incomplete factorization (the setup time) and the average time of each CG iteration with the ILU(0) preconditioner is smaller than with the AMG preconditioner.

In Table II, we summarize the AMG coarsening statistics in terms of the complexity measures c_G, c_A, and c_S for the same combinations of the coarsening parameters used in Table I. Considering the results from Tables I and II in conjunction it can be concluded that, if the coarse level matrices

DOI: 10.1002/nme

Table II. Coarsening statistics for the linear case as a function of the coarsening strategy, and the threshold θ.

n		22 764		69 285		208 965	
θ		RS2	RS1	RS2	RS1	RS2	RS1
0.25	L	10	6	13	7	13	7
	c_G	1.67	1.23	1.68	1.23	1.71	1.24
	c_A	4.64	1.59	5.21	1.58	5.82	1.59
	c_S	68.3	20.2	96.8	21.3	161.8	26.2
0.50	L	13	7	14	8	15	9
	c_G	1.93	1.34	1.98	1.35	2.02	1.36
	c_A	4.36	1.78	4.67	1.79	4.85	1.79
	c_S	44.3	18.5	58.1	20.5	71.9	23.2
0.67	L	12	9	14	9	17	10
	c_G	1.91	1.42	1.99	1.44	2.01	1.45
	c_A	3.35	1.82	3.60	1.85	3.61	1.85
	c_S	28.2	14.0	34.1	17.0	37.6	19.6
$c_S^{(1)}$		11.2		12.0		12.8	
η		0		0		0	

L is the number of coarse levels (including the finest level), c_G is the grid complexity, c_A is the operator complexity, c_S is the average matrix stencil size across all coarse levels, $c_S^{(1)}$ is the average stencil size of A, and η is defined by (17).

are too sparse, this can adversely affect the preconditioner quality (see, for example, the case $\theta = 0.67$ and RS1 coarsening).

Note that, because of small contributions from the coarsest levels, the value of c_S is usually an optimistic estimate. For example, for RS2 coarsening with $\theta = 0.25$ and $n = 208\,965$, the average stencil size was $c_S = 161.8$ (see Table II) while the largest value of $\mathrm{nnz}(A_i)/n_i$ was 435 (this was at level 7, with $n_7 = 1976$).

Finally, we comment on the total memory use, as measured by the Linux program `top`. The total memory required is modest in all cases. For example, for $n = 208\ 965$ with $\theta = 0.25$ and RS2 coarsening, the memory use was 250 MB, while for RS1 coarsening it was 175 MB. For the ILU(0) preconditioner, the total memory required was 125 MB.

Example 4.1.2

In this example we consider the diffusion problem with the same forcing term and boundary conditions as in Example 4.1.1. We discretize Equations (13)–(14) using quadratic approximation over the same tetrahedral grids introduced in Example 4.1.1. The matrix $A = A_1$ is again symmetric positive definite, but is not an M-matrix. The values of η in Table V indicate that A_1 has a considerable number of positive off-diagonal entries. In the coarsening algorithm such connections are regarded as weak, and assigned to the set D_i^{w} in (5). When constructing an interpolation operator (Equation (8)), the sum of the positive off-diagonal entries is added (lumped) to the main diagonal (the denominator in (8)). If the proportion of positive off-diagonal entries is considerable and/or the magnitude of such elements is large, the resulting intergrid transfer operators fail to preserve the properties of the original matrix, and the coarse grid problems can be considered as discretizations of a perturbed PDE. As a result, the computed preconditioner may be ineffective.

DOI: 10.1002/nme

Table III. The iteration counts for the AMG-preconditioned CG solver with DJ smoothing, applied to the linear system obtained from the quadratic FEM discretization of the Poisson problem, as a function of the damping parameter (ω).

n	53 968		165 911		508 289	
ω	RS2	RS1	RS2	RS1	RS2	RS1
1.00	>100	>100	>100	>100	>100	>100
0.80	>100	>100	>100	>100	>100	>100
0.67	12	17	12	18	12	18
0.50	13	19	13	20	13	21

Although quadratic approximation of the Poisson equation has its own merits, the real significance of this problem is to assess the effectiveness of the AMG preconditioner in the context of more complex problems. For example, the inf–sup stable mixed FEM approximation of fluid mechanics problems [1, p. 229] requires a higher-order approximation of the velocity components than is used for pressure (e.g. the P_2-P_1 approximation), see Example 4.3.1, and Example 4.4.1 and Example 4.4.2.

We first investigate the impact that the change in the properties of A has on the optimal value of the damping parameter (`control%damping`) in the DJ smoother. The convergence results are summarized in Table III ($\theta=0.25$). Here (and throughout the remainder of the paper) ω is used to denote the damping parameter. Using $\omega>0.75$, we observed a significant increase in the iteration count for the convergence of the CG solver (and for some values, convergence was not achieved). The smallest number of iterations are obtained for $\omega=0.67$ and we use this value in all remaining experiments for this example (and later in Example 4.3.1 for the P_2-P_1 case).

The results from Table III, together with the previous discussion about the coarse level matrices in a non M-matrix case, suggest that a form of a variable DJ smoother might be appropriate in this context (with a different value of the damping parameter ω_ℓ, $\ell=1,\ldots,L$ used at each level to achieve good error reduction).[‖] For this, one needs to develop a heuristic strategy to determine ω_ℓ as a function of the number of positive off-diagonal entries and their magnitudes. We remark that some preliminary experiments with a version of a variable smoother, based on truncated ILU(0) factorization, show considerable potential in the context of the convection–diffusion problem, when compared with the standard smoothers but this is outside the scope of the current paper.

In Table IV, we summarize the convergence results for the RS1 and RS2 coarsening strategies with both DJ and GS smoothing, as a function of the problem size n and the threshold θ. The best results (in terms of time) are observed for RS1 coarsening with $\theta=0.25$ or 0.5 (although the results obtained with RS2 and $\theta=0.67$ are only slightly worse). Again, in most cases, we observe the near optimal performance (as n increases, the iterations increase only slowly, with a mild superlinear increase in the execution time). The main exceptions are for RS1 coarsening with $\theta=0.67$. Comparing the results in Tables I and IV, we see that the iteration counts in the quadratic case are only slightly higher than those in the linear case.

Table V summarizes the coarsening statistics. Comparing these with those in Table II, we conclude that the average stencil sizes of the original matrix $c_S^{(1)}$ are much larger in the quadratic

[‖] In practice, divergence of the AMG-preconditioned solver reported in Table III for $\omega=1$ and $\omega=0.8$ might be the consequence of the smoother failing only at a few levels, rather than globally.

DOI: 10.1002/nme

Table IV. The iteration counts and, in brackets, the setup (top) and total (bottom) execution times (in seconds) for the AMG-preconditioned CG method applied to the linear system obtained from the quadratic FEM discretization of the Poisson problem, as a function of the coarsening strategy, the smoother, and the threshold θ.

n		53 968		165 911		508 289	
	θ	RS2	RS1	RS2	RS1	RS2	RS1
DJ	0.25	12 ($\frac{1.94}{3.04}$)	17 ($\frac{0.40}{1.05}$)	12 ($\frac{7.60}{11.8}$)	18 ($\frac{1.44}{4.07}$)	12 ($\frac{30.2}{48.6}$)	18 ($\frac{5.38}{\mathbf{16.7}}$)
	0.50	11 ($\frac{0.76}{1.41}$)	16 ($\frac{0.31}{\mathbf{0.91}}$)	11 ($\frac{2.96}{5.48}$)	19 ($\frac{1.18}{\mathbf{3.97}}$)	12 ($\frac{10.9}{22.5}$)	22 ($\frac{4.39}{18.1}$)
	0.67	12 ($\frac{0.50}{1.12}$)	22 ($\frac{0.29}{1.13}$)	12($\frac{1.96}{4.32}$)	29 ($\frac{1.10}{5.39}$)	13 ($\frac{7.38}{18.30}$)	39 ($\frac{4.13}{28.80}$)
GS	0.25	9 ($\frac{1.94}{3.31}$)	15 ($\frac{0.40}{1.17}$)	10 ($\frac{7.60}{13.40}$)	15 ($\frac{1.44}{4.33}$)	9 ($\frac{30.2}{51.8}$)	16 ($\frac{5.38}{\mathbf{18.2}}$)
	0.50	9 ($\frac{0.76}{1.56}$)	14 ($\frac{0.31}{\mathbf{1.00}}$)	9 ($\frac{2.96}{6.01}$)	16 ($\frac{1.18}{\mathbf{4.19}}$)	10 ($\frac{10.9}{24.6}$)	19 ($\frac{4.39}{19.3}$)
	0.67	10 ($\frac{0.50}{1.24}$)	19 ($\frac{0.29}{1.24}$)	11 ($\frac{1.96}{4.99}$)	25 ($\frac{1.10}{5.85}$)	12 ($\frac{7.38}{21.1}$)	33 ($\frac{4.13}{30.3}$)

For each smoother (DJ or GS), the smallest total times are in bold.

case. For smaller values of the parameter θ and RS2 coarsening this leads to a significant increase in the coarsening complexity parameters and results in longer execution times. Note, however, that for larger values of θ with both RS2 and RS1 coarsenings the complexity parameters values from Tables II and V are comparable. As before, the values of c_S are optimistic estimates of the average stencil size across all coarse levels. For example, for $n=508\,289$ and $\theta=0.25$, the largest stencil size for RS2 coarsening is $c_S^{(9)}=851.8$ with $n_9=2746$, while for RS1 coarsening the maximum is $c_S^{(3)}=77.5$ with $n_3=222\,258$. The memory use for RS2 and RS1 coarsening is 1.6 GB and 520 MB, respectively.

4.2. The convection–diffusion equation in 3D

In this section, we examine the effectiveness of an AMG-preconditioned Krylov method for solving the linear systems that arise in the FEM discretization of the 3D convection–diffusion equation. The convection–diffusion equation is a scalar elliptic PDE that models a host of important processes and phenomena in different areas, including fluid mechanics and electronics. It can occur either as a standalone problem or as part of more complex systems of PDEs (such as the Navier–Stokes equation, the Boussinesq equation or the drift-diffusion equation).

We consider the linear systems that arise in the SUPG FEM discretization of the convection–diffusion equation

$$-\nu\nabla^2 u+\vec{w}\cdot\nabla u=f \quad \text{in } \Omega\subset\mathbb{R}^3 \tag{18}$$

subject to the boundary conditions (14).

Details of the FEM discretization can be found in [1, p. 126]. Here $\nu>0$ is the diffusivity parameter** measuring the relative contributions of the convection and the diffusion (in most

**When the convection–diffusion equation occurs as a part of the momentum equation in the Navier–Stokes system, the quantity $1/\nu$ is known as the Reynolds number *Re*. In the case of the Boussinesq problem, the diffusivity parameters in the momentum and the temperature diffusion equation are expressed as functions of the Rayleigh number *Ra* and the Prandtl number *Pr* (see Section 4.4).

DOI: 10.1002/nme

Table V. Coarsening statistics for the quadratic case as a function of the coarsening type (RS2/RS1), and the threshold θ.

n		53 968		165 911		508 289	
θ		RS2	RS1	RS2	RS1	RS2	RS1
0.25	L	13	7	14	7	16	7
	c_G	2.08	1.27	2.15	1.28	2.19	1.29
	c_A	7.10	1.61	7.78	1.59	8.63	1.63
	c_S	144.2	29.7	219.0	35.3	302.0	42.1
0.50	L	14	8	15	9	16	9
	c_G	1.96	1.35	2.03	1.37	2.06	1.38
	c_A	3.40	1.50	3.68	1.52	3.78	1.53
	c_S	59.0	22.9	79.6	25.5	96.3	30.5
0.67	L	14	9	15	10	18	11
	c_G	1.93	1.43	2.00	1.44	2.03	1.46
	c_A	2.55	1.53	2.65	1.54	2.71	1.55
	c_S	36.0	20.3	41.6	21.7	45.9	23.5
$c_S^{(1)}$		22.7		24.6		25.7	
η		47.5		47.6		48.1	

L is the number of coarse levels (including the finest level), c_G is the grid complexity, c_A is the operator complexity, c_S is the average matrix stencil size across all coarse levels, $c_S^{(1)}$ is the average stencil size of A, and η is defined by (17).

practical cases, $v \ll 1$), $\vec{w}: \mathbb{R}^3 \mapsto \mathbb{R}^3$ is the vector-valued function referred to as the convective field (or wind) that determines the direction of the convection, and f is the forcing term.

The SUPG discretization of (18) and the boundary conditions (14) leads to a linear system

$$\Phi \mathbf{x} = \mathbf{b} \tag{19}$$

where the system matrix $\Phi \in \mathbb{R}^{n \times n}$ can be expressed in the form

$$\Phi = vA + C + S \tag{20}$$

Here A is the diffusion matrix defined in (15), C is the convection matrix, and S is the stabilization matrix. A is symmetric positive-definite; provided the convective field is incompressible ($\nabla \cdot \vec{w} = 0$), C is skew-symmetric (that is, $c_{ij} = -c_{ji}$, $c_{ii} = 0$, $i, j = 1, \ldots, n$); S is symmetric, but possibly indefinite. In particular, if the Galerkin FEM is used, $S = 0$. If GMG is used to precondition (19), it is essential that the SUPG discretization is applied at all the coarse levels (see [1, p. 194]). In the case of an AMG preconditioner, the coarse level matrices are created automatically. However, experience has shown that the performance of the AMG preconditioner is more robust if the original problem is discretized using the SUPG method, rather than the standard Galerkin FEM.

Although the original design of the classical AMG coarsening procedure is based on the properties of A, in practice, it adapts well to the system matrix Φ given by (20). This makes AMG a robust preconditioner for convection-dominated problems, even though it uses simple point smoothers. By the contrast, the robust performance of GMG (which uses full coarsening) requires not only SUPG discretization at all coarse levels, but also fairly sophisticated smoothing techniques, based on the

DOI: 10.1002/nme

directional ordering of vertices [1, p. 195]. A particularly difficult problem to solve in this context is the case of a highly convective flow with the wind function containing global recirculations (or multiple local recirculations).

The system matrix Φ is unsymmetric and this influences the choice of Krylov solver. Moreover, in cases $v \ll 1$, considerable amount of the off-diagonal elements are positive, and, depending on the choice of the wind $\vec{w}$, some of these elements can be quite large in magnitude. In order to provide some insight, we present in Table VIII the ratio η defined by (17), the largest positive off-diagonal entry ϕ_{ij}, and the largest diagonal entry ϕ_{ii} in the matrix Φ. We use right-preconditioned GMRES to solve (19). The HSL package `MI24` implements a restarted variant of the algorithm GMRES(m) [3, p. 172] but GMRES with no restarting can be obtained by setting the parameter m equal to the maximal allowed number of iterations; we use this setting in our tests. As in the diffusion experiments, the stopping criterion is given by (16) with $\varepsilon = 10^{-6}$.

Example 4.2.1
We solve (18) on a cylinder domain $\Omega = \{x^2 + y^2 \leqslant 1\} \times [0, 5] \subset \mathbb{R}^3$. We set the convective field to be circular $\vec{w} = (2y(1-x^2), -2x(1-y^2), \sqrt{x^2+y^2}\sin x/\sqrt{x^2+y^2})$. To avoid singularity of the wind function at the points $(0, 0, z)$, we set $w_z = 0$. The forcing term is chosen to be $f = 0$ and we use Dirichlet boundary conditions $u_D = 0$ for $z = 0$ and $z = 5$ (the bottom and the top lids of the cylinder), and $u_D = 0$ for $x^2 + y^2 = 1$ and $x \leqslant 0$, $u_D = 1$ for $x^2 + y^2 = 1$ and $x > 0$.

Table VI presents iteration counts and timings for different values of v for RS1 and RS2 coarsening. Here we use the threshold parameter $\theta = 0.25$ and GS smoothing. Results for preconditioning with ILU(0) are also presented. For ILU(0), the iteration counts deteriorate when either n is increased and/or v is decreased. However, comparing the AMG results with those for Example 4.1.1 (Table I, GS smoothing, $\theta = 0.25$), we see that, for small to moderate values of v, the effectiveness of the AMG preconditioner is almost the same as for the diffusion problem. A modest increase in the iteration count is observed for highly convective flows ($v = 0.001$) and RS1 coarsening. We also tested highly convective cases up to $v \sim 10^{-5}$ and the performance of the AMG preconditioner remains robust. We examined the norm of the forward error $\|\mathbf{x}^{(k)} - \tilde{\mathbf{x}}\|_2$ for each computed solution and found that they are slightly larger than for Example 4.1.1 but, in all cases, were less than 10^{-3}. The 'exact' solution $\tilde{\mathbf{x}}$ is computed by a sparse direct solver.

In Table VII, we report on the influence of the damping parameter on the convergence characteristics of the AMG-preconditioned GMRES method when DJ smoothing is used instead of GS. The results confirm our choice of `control%damping`$=0.8$ as the default setting for the damping parameter, although for RS2 and $v = 0.001$, using 0.5 gave similar results. This suggests that for even smaller values of v, using a smaller value of `control%damping` than the default might be preferable. However, the number of iterations required when using DJ was larger than for GS so that, even though each GS smoothing operation is more expensive, in terms of the total execution time, using DJ was not advantageous.

In Table VIII, we report the coarsening statistics for $v = 0.001$ and $\theta = 0.25$. Comparing these with the diffusion case (Table II), we see that the number of coarse levels L, the grid complexities c_G, and the operator complexities c_A are larger in the convection–diffusion case. This suggests, as expected, that AMG performs a version of semi-coarsening. By checking the geometric positions of the nodes at coarse levels, we conclude that the semi-coarsening is performed in characteristic directions, determined by the strength and the direction of the wind (unlike to the semi-coarsening performed in the GMG case, which is usually aligned with the Cartesian coordinate directions).

DOI: 10.1002/nme

Table VI. Iteration counts and, in brackets, the setup (top) and total (bottom) execution times (in seconds) for the AMG- and ILU(0)-preconditioned GMRES method applied to the linear system obtained from the linear SUPG FEM discretization of the convection–diffusion equation.

n		22 764		69 285		208 965	
	v	RS2	RS1	RS2	RS1	RS2	RS1
AMG	0.02	7 ($\frac{0.28}{0.52}$)	11 ($\frac{0.09}{0.25}$)	7 ($\frac{1.10}{1.97}$)	12 ($\frac{0.31}{0.90}$)	7 ($\frac{4.24}{7.36}$)	13 ($\frac{1.04}{3.32}$)
	0.004	9 ($\frac{0.25}{0.55}$)	15 ($\frac{0.09}{0.33}$)	8 ($\frac{0.98}{2.00}$)	15 ($\frac{0.32}{1.12}$)	7 ($\frac{3.73}{6.97}$)	15 ($\frac{1.09}{3.91}$)
	0.001	12 ($\frac{0.23}{0.64}$)	24 ($\frac{0.09}{0.49}$)	11 ($\frac{0.94}{2.30}$)	24 ($\frac{0.33}{1.69}$)	10 ($\frac{3.46}{7.94}$)	24 ($\frac{1.16}{6.00}$)
ILU(0)	0.02	81 ($\frac{4.83}{5.41}$)		131 ($\frac{45.4}{49.3}$)		202 ($\frac{445}{472}$)	
	0.004	118 ($\frac{4.83}{5.83}$)		174 ($\frac{45.4}{51.4}$)		253 ($\frac{445}{483}$)	
	0.001	187 ($\frac{4.83}{6.90}$)		282 ($\frac{45.4}{58.6}$)		553 ($\frac{445}{592}$)	

GS smoothing is used and threshold $\theta = 0.25$.

Table VII. Iteration counts for the AMG-preconditioned GMRES method, with a range of values of the damping parameter (ω) and of the diffusivity parameter v.

n		22 764		69 285		208 965	
v	ω	RS2	RS1	RS2	RS1	RS2	RS1
	1	37	50	74	52	83	62
0.001	0.8	16	33	14	35	12	34
	0.5	17	40	14	43	13	41

DJ smoothing is used and threshold $\theta = 0.25$.

Table VIII. Coarsening statistics for $\theta = 0.25$ and $v = 0.001$.

n	22 764		69 285		208 965	
	RS2	RS1	RS2	RS1	RS2	RS1
L	16	9	17	10	20	11
c_G	2.24	1.45	2.32	1.47	2.37	1.48
c_A	5.62	2.09	6.36	2.13	6.82	2.12
c_S	53.7	20.8	76.9	24.3	96.0	30.7
$c_S^{(1)}$	11.2		12.0		12.8	
η	44.1		43.6		42.7	
$\max \phi_{ij}$	0.004		0.002		0.001	
$\max \phi_{ii}$	0.026		0.015		0.009	

L is the number of coarse levels (including the finest level), c_G is the grid complexity, c_A is the operator complexity, c_S is the average matrix stencil size across all coarse levels, $c_S^{(1)}$ is the average stencil size of A, η is defined by (17), $\max \phi_{ij}$ is the largest positive off-diagonal entry, and $\max \phi_{ii}$ is the largest diagonal entry in the matrix Φ.

DOI: 10.1002/nme

4.3. The Stokes problem in 3D

The Stokes equations have their main application in modelling viscous fluids moving at low speed (e.g. blood flow in the peripheral vascular system) or are tightly contained. We are interested in the efficient iterative solution of linear systems that arise in mixed FE approximation of the Stokes system:

$$-\nabla^2\vec{u}+\nabla p=\vec{0} \tag{21}$$

$$\nabla\cdot\vec{u}=0 \tag{22}$$

in $\Omega\subset\mathbb{R}^3$. Here $\vec{u}:\mathbb{R}^3\mapsto\mathbb{R}^3$ is a vector-valued function representing velocity and $p:\mathbb{R}^3\mapsto\mathbb{R}$ is pressure. Equation (21) is referred to as the momentum equation (conservation of the momentum), while (22) is the incompressibility constraint (conservation of mass). The typical boundary conditions associated with the system (21)–(22) are of the form

$$\vec{u}=\vec{u}_D \quad \text{on } \partial\Omega_D \quad \text{and} \quad \frac{\partial\vec{u}}{\partial\hat{n}}-\hat{n}p=\vec{u}_N \quad \text{on } \partial\Omega_N \tag{23}$$

In order to ensure a unique velocity solution, we assume $\partial\Omega_D\neq\emptyset$. If $\partial\Omega_D=\partial\Omega$, then the pressure solution of (21)–(22) is unique up to a constant [1, p. 215]. This case corresponds to an enclosed flow. Otherwise, we have an inflow/outflow problem. In order to ensure global incompressibility constraint in this case, a natural outflow condition ($\vec{u}_N=\vec{0}$) is specified.

The finite element discretization of (21)–(22) is based on mixed methods, that is, the approximation spaces for the velocity components and pressure can be the same or different. However, the absence of pressure from (22) makes the choice of velocity and pressure approximation spaces dependent of each other. In order to make FE approximation effective, the approximation spaces need to satisfy the uniform inf–sup stability condition [1, p. 228]. A number of the approximation spaces combinations satisfy the stability condition (see [1, Section 5.3.1]). On the other hand, some well-known (and, because of their simplicity and the desirable properties of the resulting linear system, frequently used in engineering practice) discretization methods do not satisfy the stability criterion. The best-known example involves the use of equal-order velocity–pressure approximations. The consequence of applying a unstable mixed FEM is the presence of spurious pressure modes in the discrete pressure solution (see [1, Section 5.3.2]). To alleviate this problem, a concept of stabilization of the discrete equations is introduced. The idea behind the stabilization is to relax the discrete incompressibility condition. The procedures for this are inevitably discretization specific and usually depend on certain parameters (the exception is a recent stabilization method for linear/bilinear velocity–pressure approximation, which is parameter-free [40]).

When a stable/stabilized mixed FE discretization is applied to (21)–(22), the resulting linear system can be represented in the form:

$$\begin{bmatrix}\mathbf{A} & B^t\\ B & -C\end{bmatrix}\begin{bmatrix}\bar{\mathbf{u}}\\ \bar{p}\end{bmatrix}=\begin{bmatrix}\bar{\mathbf{f}}\\ \bar{g}\end{bmatrix} \tag{24}$$

Here $\mathbf{A}$ represents a discrete vector Laplacian ($\mathbf{A}\in\mathbb{R}^{3n_u\times 3n_u}$, where n_u is the number of velocity nodes). The matrix $\mathbf{A}$ is a block diagonal matrix, with diagonal blocks $A\in\mathbb{R}^{n_u\times n_u}$ being scalar discrete Laplacians defined in (15). The matrix $B\in\mathbb{R}^{n_p\times 3n_u}$ is a discrete divergence matrix, and the matrix $C\in\mathbb{R}^{n_p\times n_p}$ is a symmetric positive semi-definite stabilization matrix (note that if a

DOI: 10.1002/nme

stable discretization method is deployed $C=0$). The right-hand side vector components $\bar{\mathbf{f}}\in\mathbb{R}^{3n_u}$ and $\bar{g}\in\mathbb{R}^{n_p}$ (with $\sum_{i=1}^{n_p} g_i=0$) come from the non-zero velocity boundary conditions. The vectors $\bar{\mathbf{u}}=[\bar{u}_x\ \bar{u}_y\ \bar{u}_z]^{\mathrm{T}}\in\mathbb{R}^{3n_u}$ contain the three unknown velocity components for each of the velocity nodes $i=1,\ldots,n_u$ and $\bar{p}\in\mathbb{R}^{n_p}$ are the pressure unknowns.

The linear system (24) has a coefficient matrix that is symmetric but indefinite. This influences the choice of Krylov solver. One of the earliest methods for the solution of the system (24) is based on the Uzawa algorithm [42]. The Uzawa algorithm, which is widely used in engineering practice, is effectively a fixed-parameter, first-order Richardson iteration applied to the Schur complement system

$$(B\mathbf{A}^{-1}B^t+C)\bar{p}=B\mathbf{A}^{-1}\bar{\mathbf{f}}+\bar{g} \tag{25}$$

To make the method computationally more effective, inexact and preconditioned variants of the classical algorithm were introduced (see, for example [43]). In general, convergence of an inexact Uzawa algorithm is slow and depends on several parameters.

Here we study an efficient block preconditioning of the system (24) proposed in [44]. In particular, the optimal block preconditioner for the system (24) is of the form

$$P=\begin{bmatrix}\mathbf{A} & 0\\ 0 & M_p\end{bmatrix} \tag{26}$$

where $M_p\in\mathbb{R}^{n_p\times n_p}$ is the mass matrix defined on the pressure approximation space. The Stokes matrix preconditioned by (26) has its spectral bounds constant (independent of the problem size), thus offering the potential of developing an optimal solver for discrete Stokes systems (24). To achieve this, the efficient implementation of the preconditioner is needed, which requires the approximate (but spectrally equivalent) inversion of the matrix blocks $\mathbf{A}$ and M_p at the optimal cost. For the block $\mathbf{A}$ this can be accomplished by applying AMG to each of the diagonal blocks. Note also that, if each of the velocity components has the same type of boundary conditions at each boundary node (e.g. there is no stress-free boundary), then all the block diagonal matrices in $\mathbf{A}$ are the same and the AMG coarsening procedure needs to be applied only once. A spectrally equivalent inverse of M_p can be constructed by finding an inverse of its diagonal [45].

Example 4.3.1

We solve the Stokes flow problem (24) for a flow through a pipe with a sudden expansion $\Omega=\{x^2+y^2\leqslant r_1^2\}\times[0,\ell_1]\cup\{x^2+y^2\leqslant r_2^2\}\times[0,-\ell_2]$ (see Figure 1). The parameters used in our calculations are $r_1=0.5$, $r_2=1$, $\ell_1=5$, $\ell_2=7.5$. We set a Pouiselle parabolic velocity profile on the top lid $z=\ell_1$ as $u_x=u_y=0$, $u_z=-1+(x^2+y^2)/r_1^2$ (see [1, p. 219]), natural outflow condition for $z=\ell_2$ and no-slip velocity boundary condition ($\vec{u}=\vec{0}$) elsewhere on $\partial\Omega$. We test the effectiveness of the GMRES iterative method preconditioned by (26) for two different discretizations of the Stokes problem: the uniformly stable P_2-P_1 (thetrahedral elements with quadratic velocities and linear pressure) and the stabilized P_1-P_1 (equal order linear velocities and pressure, stabilized by the method of Dohrmann and Bochev [40]). Again, the stopping criterion for GMRES is defined by (16).

In Table IX, we present the iteration counts and execution times for the P_1-P_1 discretization. Results are for RS1 and RS2 coarsenings with $\theta=0.25$ and for both GS and DJ ($\omega=0.8$) smoothers. Each of the scalar discrete Laplacians is preconditioned by one V(1,1) cycle of AMG and the mass matrix is approximated by its diagonal.

DOI: 10.1002/nme

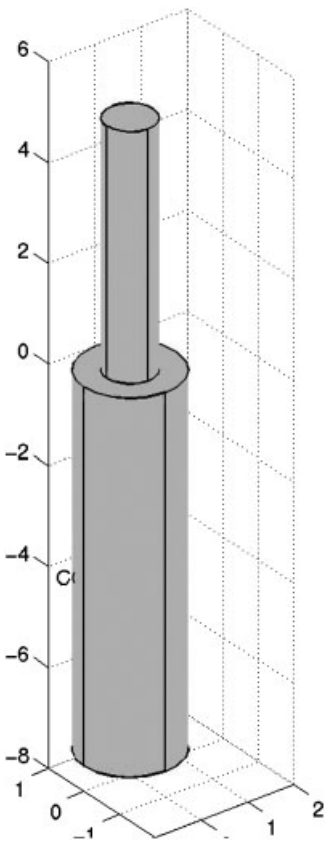

Figure 1. Domain geometry for a pipe with sudden expansion $\Omega=\{x^2+y^2\leqslant r_1^2\}\times[0,\ell_1]\cup\{x^2+y^2\leqslant r_2^2\}\times[0,-\ell_2]$ with $r_1=0.5$, $r_2=1$, $\ell_1=5$, $\ell_2=7.5$.

We see that for RS2 coarsening the iteration counts are roughly constant when the problem size is increased. As in the case of the diffusion problem, when the problem size is increased by a factor 3, the total times increase roughly by a factor of 4, indicating that the asymptotic behaviour of the iterative algorithm is dominated by the asymptotic behaviour of the AMG preconditioner. For RS1 coarsening there is a moderate increase in the iteration counts with the problem size. Despite this growth, for the problem sizes tested, the total times using RS1 coarsening are slightly less than using RS2 coarsening (for larger problem sizes, however, we anticipate the roles being reversed). In both cases (RS2 and RS1) using the DJ smoother resulted in faster total times than using the GS smoother.

In Table X, we summarize the convergence results and the execution times for the stable P_2-P_1 discretization. Note that, in the light of the results from Table III, we adopt $\omega=0.67$ for the DJ smoothing. All other experiment details are the same as for P_1-P_1 discretization.

We see the same asymptotic behaviour in the execution time as for the P_1-P_1 discretization. Moreover, longer execution times in the P_2-P_1 case are a consequence of larger matrix stencils for the P_2 discretization of the Laplacian (cf. Example 4.1.2). In terms of convergence characteristics, for GS smoothing we observe spectral equivalence of the preconditiner (26) for both RS2 and

DOI: 10.1002/nme

Table IX. The iteration counts and, in brackets, the setup (top) and total (bottom) execution times (in seconds) for the GMRES method preconditioned by the block-diagonal preconditioner (26) applied to the linear system (24) obtained using stabilized the $P_1 - P_1$ mixed FEM.

Refinement level	1	2	3
n	26 476	83 080	254 216
RS2 with GS	87 ($\frac{0.04}{1.79}$)	90 ($\frac{0.19}{7.43}$)	92 ($\frac{0.86}{30.98}$)
RS2 with DJ	89 ($\frac{0.04}{1.51}$)	94 ($\frac{0.19}{6.08}$)	96 ($\frac{0.86}{25.02}$)
RS1 with GS	93 ($\frac{0.02}{1.43}$)	102 ($\frac{0.07}{5.70}$)	112 ($\frac{0.26}{25.10}$)
RS1 with DJ	99 ($\frac{0.02}{\mathbf{1.41}}$)	108 ($\frac{0.07}{\mathbf{5.48}}$)	119 ($\frac{0.26}{\mathbf{24.37}}$)

For each problem size, the smallest total times are in bold.

Table X. The iteration counts and, in brackets, the setup (top) and total (bottom) execution times (in seconds) for the GMRES method preconditioned by the block-diagonal preconditioner (26) applied to the linear system (24) obtained using stable $P_2 - P_1$ mixed FEM.

Refinement level	0	1	2
n	42 367	145 384	463 747
RS2 with GS	111 ($\frac{0.16}{6.84}$)	127 ($\frac{1.24}{48.26}$)	123 ($\frac{5.54}{200.69}$)
RS2 with DJ	123 ($\frac{0.16}{5.43}$)	145 ($\frac{1.24}{37.36}$)	146 ($\frac{5.54}{170.28}$)
RS1 with GS	126 ($\frac{0.05}{4.10}$)	144 ($\frac{0.30}{24.84}$)	146 ($\frac{1.19}{111.18}$)
RS1 with DJ	134 ($\frac{0.05}{\mathbf{3.85}}$)	157 ($\frac{0.30}{\mathbf{23.07}}$)	163 ($\frac{1.19}{\mathbf{109.21}}$)

For each problem size, the smallest total times are in bold.

RS1 coarsening, and a very small increase in iteration counts in RS1/DJ case. Unlike the $P_1 - P_1$ discretization, the total times using RS1 coarsening are much less (by approximately 40–50%) than using RS2 coarsening.

As a conclusion, we give some comparisons of the convergence results obtained for a 3D problem with the convergence results for a 2D model problem (driven lid cavity, see [1, p. 210]). Convergence results for a 2D model problem are reported in [1, p. 301] and from them we can see roughly the same asymptotic behaviour that we observed between the iteration counts for the $P_1 - P_1$ and the $P_2 - P_1$ discretizations. The main difference between 2D and 3D is the total iteration counts, which are considerably higher for a 3D problem. For the example tested here, relatively high iteration counts can be attributed to a stretched domain. It is known that multigrid gradually loses its effectiveness when the domain/grid dimension in one or more coordinate directions is much larger than in the remaining directions (see [46]).†† One possible way of alleviating this problem is to limit the number of coarse levels, so that on the coarsest level the size of the system

††To make a fair comparison with 2D results, we also tested the preconditioned GMRES method on a driven lid cavity problem in 3D (the unit cube domain). In this case we observed $\sim$45 iterations for the $P_1 - P_1$ discretization and $\sim$80 iterations for the $P_2 - P_1$ discretization—a 50% increase compared with a 2D analogue.

DOI: 10.1002/nme

is greater than 1 (in our experiments, we coarsen to a single point). Another potential reason for the loss of preconditioner efficiency may be that in 3D the diagonal approximation of the mass matrix M_p loses its effectiveness. Some better choices include lumping or performing ILU(0) factorization of M_p.

4.4. The transient Boussinesq problem in 3D

The Boussinesq system of PDEs (sometimes referred to as the Boussinesq approximation of the Navier–Stokes system) is a non-linear, transient system of PDEs. This system is a mathematical model for the motion of incompressible viscous fluids buoyed by thermal effects. There are a number of very important physical and engineering applications that use the Boussinesq model, including heat exchange systems [47] (e.g. nuclear reactor cooling), semiconductor crystal growth [37], convection of the earth's mantle and continental drift [48], cooling of electronic equipment, weather prediction, modelling of katabaric winds, ocean convection modelling, solar granulation, superconductivity and cryogenic flows [49]. We also mention that magneto-hydrodynamic processes are modelled by essentially the same system of PDEs, whereby the magnetic force is added to the external forcing term (see, for example [50]).

There are a number of different configurations for which relevant applications of thermally driven flows are studied both experimentally and computationally. In each case, a fluid is driven into convective motion by the existence of the temperature gradient that plays the role of the buoyancy force. Here we restrict our attention to enclosed cavity thermally driven flows. We study two physically relevant configurations: laterally heated cavities (Example 4.4.1), and the Rayleigh–Benard convection (Example 4.4.2).

To normalize the Boussinesq equations we follow the approach introduced in [51]. Let g be the gravitational acceleration, β the thermal expansion coefficient, ΔT the temperature gradient applied to a fluid layer and ρ the fluid density. Given a characteristic domain scale l, buoyancy velocity scale $U=\sqrt{g\beta l\Delta T}$, time scale $\tau=l/U$, and pressure $\tilde{p}=\rho U^2$, we can write the non-dimensional governing equations for the transient thermal convection problem as the incompressible Navier–Stokes equations (conservation of the momentum and the incompressibility condition), coupled with the temperature diffusion equation (conservation of energy) as:

$$\frac{\partial\vec{u}}{\partial t}-\sqrt{\frac{Pr}{Ra}}\nabla^2\vec{u}+\vec{u}\cdot\nabla\vec{u}+\nabla p-\hat{k}T=\vec{0} \tag{27}$$

$$\nabla\cdot\vec{u}=0 \tag{28}$$

$$\frac{\partial T}{\partial t}-\frac{1}{\sqrt{Pr\cdot Ra}}\nabla^2T+\vec{u}\cdot\nabla T=0 \tag{29}$$

defined in $\Omega\subset\mathbb{R}^3\times[0,\Theta]$, where Θ is the length of the time interval. Here Ra and Pr are, respectively, the Rayleigh number given by

$$Ra=\frac{g\beta\Delta Tl^3}{\mu\alpha} \tag{30}$$

and the Prandtl number given by

$$Pr=\frac{\mu}{\alpha} \tag{31}$$

DOI: 10.1002/nme

where μ is the kinematic viscosity, and α is the thermal diffusivity. In (27), $\hat{k}$ denotes the unit vector in the z direction (it is assumed that the gravity force acts in the negative z direction). We seek the unknown velocity $\vec{u}:\mathbb{R}^3\to\mathbb{R}^3$, pressure $p:\mathbb{R}^3\to\mathbb{R}$, and the temperature $T:\mathbb{R}^3\to\mathbb{R}$ that satisfy (27)–(29) subject to a suitable set of boundary conditions and initial conditions. We will specify the exact set of these conditions for each of the two problems below.

Although numerical simulations of the transient problem are usually required in engineering practice, solution of the steady-state Boussinesq system (where $\partial\vec{u}/\partial t=\vec{0}$ and $\partial T/\partial t=0$) in (27)–(29) is also an important problem.[‡‡] The main applications include detection of the bifurcation phenomena [52] (leading to multiple stable solutions, time-periodic solutions, transition to chaotic flows, etc.). There is a wide body of the literature devoted to this problem (see, for example [53–56]). This is the reason why we report results in our examples for sub-critical regimes. As the solution tends to a steady state, the adaptive time-stepping algorithm will take increasingly large time steps, and ultimately the solution of the transient problem will be as difficult as the solution of the steady-state problem.

The finite element discretization of the system (27)–(29) subject to appropriate boundary and initial conditions is based on the mixed method for the Navier–Stokes part (with the same stability issues as in the case of the Stokes problem), while the temperature is usually discretized using a quadratic approximation. This results in a system of non-linear DAEs

$$\begin{bmatrix}\mathbf{M}_u\\0\\M_T\end{bmatrix}\begin{bmatrix}\dot{\bar{\mathbf{u}}}\\0\\\dot{\bar{T}}\end{bmatrix}+\begin{bmatrix}\mathbf{F}_u(\bar{\mathbf{u}})&B^t&-M_{u,T}\\B&0&0\\0&0&F_T(\bar{\mathbf{u}})\end{bmatrix}\begin{bmatrix}\bar{\mathbf{u}}\\\bar{p}\\\bar{T}\end{bmatrix}=\begin{bmatrix}\bar{\mathbf{f}}_{\mathbf{u}}\\\bar{f}_p\\\bar{f}_T\end{bmatrix}\tag{32}$$

Here $\mathbf{F}_u(\bar{\mathbf{u}})$ denotes a discrete momentum block, B is a discrete divergence operator, and $F_T(\bar{\mathbf{u}})$ is a discrete temperature diffusion block. The matrix $M_{u,T}$ is the coupling block arising from the buoyancy term in the momentum equation. The right-hand side vector contains the non-zero terms arising from Dirichlet boundary conditions. The standard 'dot' notation denotes differentiation with respect to time ($\dot{\bar{\mathbf{u}}}=\partial\bar{\mathbf{u}}/\partial t$, $\dot{\bar{T}}=\partial\bar{T}/\partial t$). $\mathbf{M}_u$ is the velocity mass matrix (with a block-diagonal structure), and M_T is the temperature mass matrix. The unknown quantities in (32) are the velocity $\bar{\mathbf{u}}=[\bar{u}_x\ \bar{u}_y\ \bar{u}_z]^{\mathrm{T}}$, pressure $\bar{p}$, and the temperature $\bar{T}$. The DAE system (32) is solved using the sTR method [41]. A modified version is introduced to prevent the locking effect in the step size selection. The method also minimizes any numerical damping in the solution. The DAE system (32) is reformulated in terms of accelerations, and the non-linear convective terms $\mathbf{F}_u(\bar{\mathbf{u}})$ and $F_T(\bar{\mathbf{u}})$ are linearized using a second-order scheme described in [41]. This leaves us with the task of solving a linear system of the form

$$\begin{bmatrix}\tilde{\mathbf{F}}_u&B^t&-\tilde{M}_{u,T}\\B&0&0\\0&0&\tilde{F}_T\end{bmatrix}\begin{bmatrix}\bar{\mathbf{v}}_u\\\bar{p}\\\bar{v}_T\end{bmatrix}=\begin{bmatrix}\tilde{\mathbf{f}}_{\mathbf{u}}\\\tilde{f}_p\\\tilde{f}_T\end{bmatrix}\tag{33}$$

at each time step. This system is of order $n=3n_u+n_p+n_T$, where n_u, n_p, and n_T are, respectively, the number of velocity nodes, pressure nodes, and temperature nodes. Further details are given

[‡‡]The problem of iterative solution of steady-state equations is usually more difficult than the solution of systems arising in discretizations of transient equations.

DOI: 10.1002/nme

in [57, p. 71]. To precondition the system (33), we apply a recently introduced block preconditioner (Elman *et al.*, 2009; in preparation), which represents the extension of the block preconditioning framework for the Navier–Stokes equations [1, p. 353] referred to as the least-square commutator (LSC) preconditioner. The preconditiner has the block form

$$P = \begin{bmatrix} \tilde{\mathbf{F}}_u & B^t & -\tilde{M}_{u,T} \\ 0 & -\tilde{S} & 0 \\ 0 & 0 & \tilde{F}_T \end{bmatrix} \tag{34}$$

In (34), $\tilde{S}$ represents an LSC approximation of the pressure Schur complement $S = B\tilde{\mathbf{F}}_u^{-1}B^t$ given by

$$\tilde{S} = (B\mathbf{M}_u^{-1}B^t)(B\mathbf{M}_u^{-1}\tilde{\mathbf{F}}_u\mathbf{M}_u^{-1}B^t)^{-1}(B\mathbf{M}_u^{-1}B^t) \tag{35}$$

(see [1, p.354]).

Efficient application of the action of $\tilde{S}^{-1}$ to a vector involves two approximate inversions of the scaled discrete Laplacian $B\mathbf{M}_u^{-1}B^t$, which can be achieved by a small number of AMG V-cycles, and a number of sparse matrix–vector multiplications. The operator $\tilde{\mathbf{F}}_u$ is block diagonal,[§§] with the main diagonal blocks being the convection-diffusion matrices, whose inverses are also approximated by a small number of AMG V-cycles (for the optimal choice of coarsening/smoothers for this subproblem, see Section 4.2). If the velocity components all have the same type of boundary conditions on the whole boundary $\partial\Omega$, the diagonal blocks in $\tilde{\mathbf{F}}_u$ are identical, and the coarsening needs only to be applied to one of them and then reused in the solution phase for each of the three blocks. Note that this is not the case if Newton's linearization is used. Then $\tilde{\mathbf{F}}_u$ is either approximated by its block-diagonal part $\tilde{\mathbf{F}}_u \approx \text{blockdiag}(\tilde{F}_u^x, \tilde{F}_u^y, \tilde{F}_u^z)^{\mathrm{T}}$ and the coarsening process is applied separately to each of the sub-blocks, or the coarsening is applied in a black box fashion to the entire block $\tilde{\mathbf{F}}_u$. The temperature block $\tilde{F}_T$ is also a convection–diffusion matrix. It has a different value of the diffusivity parameter $v_T = 1/\sqrt{Pr \cdot Ra}$ from $\tilde{\mathbf{F}}_u$ (given by $v_M = \sqrt{Pr/Ra}$). In practice, the strength of convection in these two blocks can be very different, and usually they have different types of boundary conditions (see the examples below). Thus, the coarsening generated for one of the diagonal blocks in $\tilde{\mathbf{F}}_u$ does not apply to $\tilde{F}_T$. In all our experiments in this subsection the block inverses are approximated by one $V(1,1)$ cycle. In the case of convection blocks, we use GS smoothing, while for the diffusion block we use DJ ($\omega = 0.8$). We use the GMRES method and set the maximum number of iterations to 70.

Example 4.4.1

We first solve the Boussinesq system (27)–(29) for the molten gallium problem (see [37]). The laterally heated cavity problem in liquid metals is an important problem arising in the semiconductor crystal growing process—the Bridgman technique. This involves a slab of molten material being drawn slowly from a furnace and gradually solidified. During the complex solidification process, a temperature gradient between the opposing vertical walls induces a convective flow. To create high-quality materials for optoelectronic applications, it is necessary to ensure that the induced flow is laminar, that is, the cooling process does not induce an unstable flow pattern (if the temperature

[§§] Note that this is only the case when linearized equations or Picard linearization are used [1, p. 327].

DOI: 10.1002/nme

gradient is too large, the system will undergo a transition from a stable to an oscillatory flow via a Hopf bifurcation—see [54]).

We take $\Omega=[0,4]\times[0,1]\times[0,1]$ as the problem domain. The velocity is set to be zero on the whole boundary ($\vec{u}=\vec{0}$ on $\partial\Omega$), and we set the temperature difference on the two opposing vertical walls ($T(x=0,t\to\infty)=0.5, T(x=4,t\to\infty)=-0.5$). The remaining boundary is assumed to be adiabatic (the Neumann boundary conditions $\partial T/\partial\hat{n}=0$ for the temperature is applied). To achieve a gradual emergence of the flow, we start the system from the rest (the initial condition for both the velocity and the temperature is 0 everywhere in $\bar{\Omega}=\Omega\cup\partial\Omega$). Then the opposing vertical walls are gradually heated/cooled following the exponential law

$$T(x=0,t)=0.5(1-\mathrm{e}^{-\sigma t}),\quad T(x=4,t)=-0.5(1-\mathrm{e}^{-\sigma t}) \tag{36}$$

where $\sigma\in\mathbb{R}$. In practice, we found that $\sigma=10$ gives temperature boundary and initial conditions close to a realistic physical scenario. An alternative approach of having a sudden jump in the temperature profile at the vertical boundaries (the Heaviside function) may trigger temporal oscillations in the computed solution. As the sTR algorithm is almost damping-neutral, such oscillations would persist in time and affect the nature and the accuracy of the computed solution.

We study the problem in the laminar regime for $Ra=1250$ and $Ra=2500$ and $Pr=0.025$ (the value for molten gallium used in the literature). According to [37], the transition to time-periodic flow for this configuration occurs at $Ra\simeq 5100$. The domain is discretized by a non-structured tetrahedral grid. In Table XI, we present the summary of the transient simulation. The time interval is $[0,10^4]$. For this type of flow, the momentum equation is strongly convection-dominated ($v_M\sim\frac{1}{225}$ and $v_M\sim\frac{1}{316}$ for $Ra=1250$ and $Ra=2500$, respectively), while the temperature equation is only mildly convective ($v_T\sim\frac{1}{5.6}$ and $v_T\sim\frac{1}{8}$, respectively). We report the number of time steps *nsteps* (in our simulations we did not observe any step rejections), together with the average number of GMRES iterations

$$\bar{N}_{it}=\sum_{t=1}^{nsteps} N_{it}(t)/nsteps \tag{37}$$

where $N_{it}(t)$ is the number of iterations at the tth step, and the maximal number of GMRES iterations $N_{it}^{\max}$. We also report the average time per GMRES iteration for each problem size.

Note that for $Ra=2500$, the coarsest grid was too coarse to achieve the solution convergence (no SUPG stabilization of the momentum and the temperature blocks has been applied). In this case, the computed steady solution is not convergent and this, in turn, causes problems with the convergence of the iterative solver. We observe that the iteration counts exhibit only mild growth with problem size and with increasing Ra. Moreover, the asymptotic behaviour of the execution time matches that of the AMG preconditioner reported in previous sections, indicating that the block preconditioning framework within which AMG is deployed does not cause any deterioration in the overall asymptotic performance.

Example 4.4.2

We solve the Boussinesq problem for the Rayleigh–Benard (R-B) configuration that models convection in liquid helium [49]. This has a number of applications in heat exchanging systems and in understanding some aspects of superconductivity. In the case of R-B convection, the buoyancy force caused by a temperature gradient is balanced by the gravity and the fluid flow does not emerge for any value of Ra. Let $(Ra)_{\mathrm{c}}$ denote the critical value of Ra (the critical magnitude of

DOI: 10.1002/nme

Table XI. The average $\bar{N}_{it}$ and maximal $N_{it}^{\max}$ iteration counts and average time (in seconds) per Krylov iteration for GMRES preconditioned by (34) applied to the system (33) for the molten gallium simulation.

n		71 814	211 263	623 273
$Ra=1250$	*nsteps*	121	122	124
	$\bar{N}_{it}$	20.4	23.8	24.9
	$N_{it}^{\max}$	39	46	53
$Ra=2500$	*nsteps*	*	137	144
	$\bar{N}_{it}$	*	26.3	28.1
	$N_{it}^{\max}$	*	54	59
Average time		0.08	0.29	1.20

n is the problem size and *nsteps* is the total number of time steps in the simulation. * denotes no convergence.

the temperature gradient—see (30)) for which the convective flow emerges. If $Ra<(Ra)_c$, heat is transferred through the fluid by conduction only. For $Ra=(Ra)_c$, the fluid loses its stability as the cold, heavier fluid towards the top of the layer becomes unstable to the hot, lighter fluid at the bottom. The system moves to a new stable configuration (via a pitchfork bifurcation, see [53, Chapter 6]) of convective flow with a well-known straight roll pattern [49]. Further increase in Ra introduces more instabilities into the flow, such as skewed varicose instability, spiral and focal instabilities, etc. For two rigid horizontal boundaries and an infinite fluid layer, the onset of convection occurs at $(Ra)_c \simeq 1708$. In our experiments, we consider stable flow with $Ra=2000$ and $Pr=0.5$.

We consider the R-B problem on a thin cylinder $\Omega=\{x^2+y^2\leqslant r^2\}\times[0,1]$ with $r=10$. The velocity boundary conditions are no slip on the whole boundary ($\vec{u}=\vec{0}$ on $\partial\Omega$), and we set the temperature gradient in the fluid by imposing $T(z=0,t\to\infty)=0.5$, $T(z=1,t\to\infty)=-0.5$. The cylinder envelope is assumed to be adiabatic ($\partial T/\partial\hat{n}=0$) and we apply a similar gradual heating of the system as described by (36). All the parameters in the solver are as in the previous example.

In Table XII, we present a summary of the transient simulation. The time interval $\tau=[0,10^5]$ is selected to be long enough for the flow to settle. For the chosen Ra and Pr, both the momentum and the temperature equation are mildly convective ($v_M \sim \frac{1}{63}$ and $v_T \sim \frac{1}{32}$). However, a substantial challenge for the linear solver is the stretch of the domain in the x and y directions and the fact that the velocity field contains multiple recirculating flows. We report the same data as in Table XI but in this case the number of rejected time steps was 10, because the local truncation error produced by the predictor–corrector scheme in the sTR method at these steps exceeded the prescribed tolerance.

In Figure 2, we present the time step size evolution (left) and the iteration counts needed to solve (33) at each time step (right).

4.5. Comparison with existing codes

Finally, for completeness, we present a brief comparison of the performance of `HSL_MI20` with two well-known AMG codes: BoomerAMG [19], which is a part of `hypre` [20], a software library

DOI: 10.1002/nme

Table XII. The average $\bar{N}_{it}$ and maximal $N_{it}^{\max}$ iteration counts and average time (in seconds) per Krylov iteration for GMRES preconditioned by (34) applied to the system (33) for the liquid helium problem.

n		67 401	203 379	574 631
$Ra = 2000$	*nsteps*	215	200	196
	$\bar{N}_{it}$	19.1	22.5	23.4
	$N_{it}^{\max}$	25	28	29
Average time		0.07	0.26	1.07

n is the problem size and *nsteps* is the total number of time steps in the simulation.

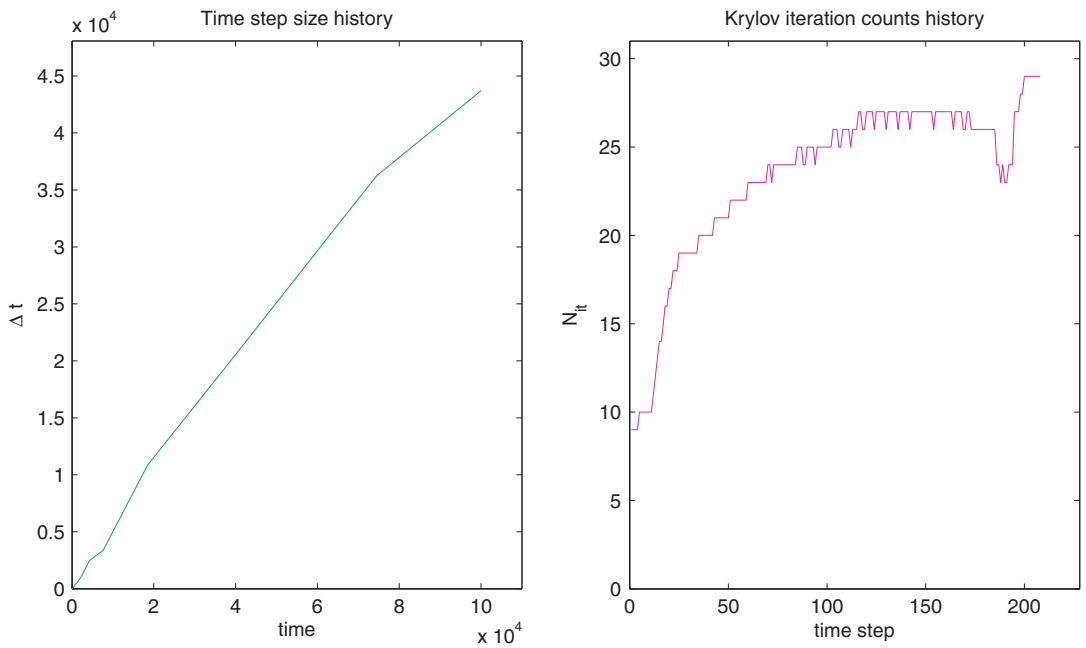

Figure 2. Time step size history (left) and the convergence history of the preconditioned GMRES at each time step (right) for the liquid helium problem (Example 4.4.2).

developed at the Lawrence Livermore National Laboratory for the parallel solution of large, sparse linear systems, and ML, which is a part of the Trilinos project [21], developed at Sandia National Laboratory as an object-oriented framework for the solution of large-scale complex multiphysics applications.

The parallel code BoomerAMG offers a number of different coarsening strategies of which we consider Ruge–Stüben coarsening and CLJP coarsening [58]. CLJP coarsening is based on a parallel graph partitioning procedure, introduced with the aim of generating in parallel the same coarse grid hierarchy, regardless of the domain partitioning. This is not the case with parallel

DOI: 10.1002/nme

Table XIII. The iteration counts and, brackets in, the setup (top) and total (bottom) execution times (in seconds) for three AMG codes used with the CG method to solve the system obtained from the bilinear FEM discretization of the Poisson problem.

Grid	$24\times24\times24$	$36\times36\times36$	$48\times48\times48$
n	103 823	357 911	857 375
BoomerAMG with RS2	4 ($\frac{2.81}{3.95}$)	4 ($\frac{10.62}{14.69}$)	4 ($\frac{26.69}{36.79}$)
BoomerAMG with CLJP	4 ($\frac{3.04}{4.26}$)	4 ($\frac{11.26}{15.63}$)	4 ($\frac{28.06}{38.86}$)
ML with SA	17 ($\frac{1.17}{3.80}$)	17 ($\frac{4.12}{14.77}$)	19 ($\frac{11.49}{39.56}$)
`HSL_MI20` with RS2	5 ($\frac{2.37}{3.50}$)	5 ($\frac{8.56}{12.60}$)	5 ($\frac{21.17}{31.00}$)
`HSL_MI20` with RS1	5 ($\frac{1.47}{\mathbf{2.58}}$)	5 ($\frac{5.23}{\mathbf{9.20}}$)	5 ($\frac{12.78}{\mathbf{22.47}}$)

The smallest total times are in bold.

Ruge–Stüben coarsening [17], where different strategies are deployed to coarsen the nodes in subdomain interiors and at the subdomain boundaries (so-called third pass coarsening). A drawback of CLJP coarsening is that it selects the coarse grids with more points than necessary.

ML is a parallel multi-level preconditioner package for sparse linear systems. It is based on an SA approach [14], where a sparse matrix-associated graph is coloured to create groups of nodes (aggregates). The preliminary projection operator is defined from this splitting. The final projection operator is defined by applying a smoother to the preliminary operator.

Example 4.5.1

We select a simple model problem for the purpose of this comparison. We solve (13) on a unit cube domain $\Omega=[0,1]^3$. The right-hand side f in (15) is selected to be constant, and we assume the case of homogeneous Dirichlet boundary conditions. The problem is discretized using `oomph-lib` [59] with Q_1 bilinear finite elements on a uniform, tensor product grid. This implies coefficient matrices with a regular sparsity pattern, reflecting a standard 27-point stencil ($c_S^{(1)}=27$ in this case). We use CG preconditioned by AMG. For BoomerAMG and ML we use the implementation of the CG method from `hypre` and Trilinos, respectively. In this way we avoid any incompatibility between different packages and programming languages. We use one V(2,2) cycle of AMG (this is a default in all three packages) with DJ smoother ($\omega=0.8$). The strength of dependence parameter is $\theta=0.25$ for the Ruge-Stüben coarsening. All other parameters in BoomerAMG, ML, and `HSL_MI20` are given their default values. The results are summarized in Table XIII for three problem sizes.

We see that `HSL_MI20` is competitive with the well-established codes when they run in serial. Moreover, if the execution time asymptotics are examined, the `HSL_MI20` times are closer to the ideal scaling for the case of structured grid problems than for non-structured grids that we reported on in Sections 4.1–4.4.

We remark that recently a new code AGMG [60] has appeared. It uses an aggregation approach, where two passes of a pairwise matching algorithm are applied to the matrix graph. The matching algorithm generates a problem-dependent coarsening. A piecewise constant prolongation ensures low algorithm cost. The scalability is improved using a K-cycle MG scheme. Although we have not explicitly compared our code with AGMG (mainly because of the different nature of the MG

DOI: 10.1002/nme

cycles used by the two codes), an inspection of the results reported in [60] suggests that the operator complexities generated by AGMG for model problems obtained by 7-point stencils in 3D are comparable with those obtained by the RS1 coarsening within HSL_MI20.

5. CONCLUDING REMARKS AND FUTURE DIRECTIONS

In this paper, a new and efficient Fortran implementation of the classical AMG algorithm has been described. We have demonstrated the successful use of the code as a preconditioner for discrete scalar elliptic problems in 3D and as a building block for efficient block preconditioner for the Stokes and transient Boussinesq problem. Our initial study of the effectiveness of AMG within the block preconditioner suggests such solvers can be obtained for other important problems in 3D (see [3, p. 337]). Block preconditioning is suitable for the discrete problems obtained from approximations of systems of PDEs and/or PDEs in which the unknown functions are vector fields. Such problems arise in modelling multi-physics problems, that is, problems with their constitutive parts coming from different areas (for instance, fluid–solid interaction problems, magnetohydrodynamics, etc.). In such cases, each node in the mesh is associated with several degrees of freedom corresponding to different physical quantities. Given that a suitable enumeration scheme for the unknowns is adopted (whereby the unknowns of the same kind are enumerated consecutively), we obtain a natural blocking of the underlying coefficient matrix. Then an effective preconditioner is usually associated with a block algorithm for finding an approximate inverse of the coefficient matrix. In this context, AMG is frequently used for (approximately) inverting the principal diagonal blocks or the associated Schur complements (which, in some cases, are close to scalar elliptic discrete operators, for which AMG is ideally suited). Such solvers are the essential part of large-scale scientific computations, such as the simulation of transient phenomena in 3D (if modern implicit adaptive time-stepping methods are used for solving the systems of DAEs [61, Section 3.16]), or in linear stability analysis (where the continuation methods are used for bifurcation tracking [52]). Some examples of effective block preconditioners based on AMG include [7] in reservoir simulation, [5] in linear elasticity, and [1] in fluid mechanics. However, most of the studies in this context are carried out for problems in 2D. A case study using our AMG code HSL_MI20 for 3D problems in fluid mechanics is currently being carried out (Boyle *et al.*, 2009; in preparation).

HSL_MI20 is available as part of the 2007 release of the mathematical software library HSL. All use of HSL requires a licence. Individual HSL packages (together with their dependencies and accompanying documentation) are available without charge to individual academic users for their personal (non-commercial) research and for teaching; licences for other uses involve a fee. Details of the packages and how to obtain a licence plus conditions of use are available at www.cse.clrc.ac.uk/nag/hsl/. A Matlab interface to HSL_MI20 has been developed and made available via the above web site, making it very straightforward for Matlab users to experiment with AMG.

ACKNOWLEDGEMENTS

We acknowledge the financial support of EPSRC under grant EP/C000528/1 (PI David Silvester). The unstructured finite element discretization of problems considered in Section 4 was performed by the parallel Fortran code femFluidMechanics, which is designed for 3D simulations of problems in

DOI: 10.1002/nme

fluid mechanics. The code was developed by Christopher Smethurst in the School of Computer Science at the University of Manchester. We also acknowledge the use of the `oomph-lib` for discretizing the structured problem in Section 4.5. `oomph-lib` is a parallel, object-oriented multi-physics finite element library, developed by Matthias Heil and Andrew Hazel in the School of Mathematics of the University of Manchester. We are also grateful to the anonymous referees for their helpful and constructive comments.

REFERENCES

1. Elman H, Silvester D, Wathen A. *Finite Elements and Fast Iterative Solvers*. Oxford University Press: Oxford, 2005.
2. Duff IS, Erisman AM, Reid JK. *Direct Methods for Sparse Matrices*. Oxford University Press: Oxford, 1986.
3. Saad Y. *Iterative Methods for Sparse Linear Systems* (2nd edn). SIAM: Philadelphia, PA, 2003.
4. Dongarra JJ, Duff IS, Sorensen DC, van der Vorst HA. *Numerical Linear Algebra for High-Performance Computers*. SIAM: Philadelphia, PA, 1998.
5. Mijalković S, Mihajlović M. A component decomposition preconditioning for 3D stress analysis problems. *Numerical Linear Algebra and its Applications* 2002; **9**:567–583.
6. Silvester DJ, Mihajlović MD. A black-box multigrid preconditioner for the biharmonic equation. *BIT* 2004; **44**:151–163.
7. Powell C, Silvester D. Optimal preconditioning for Raviart–Thomas mixed formulation of second-order elliptic problems. *SIAM Journal on Matrix Analysis and Applications* 2004; **25**:718–738.
8. Haase G, Kuhn M, Langer U. Parallel multigrid 3D Maxwell solvers. *Parallel Computing* 2001; **27**:761–775.
9. Hu J, Tuminaro R, Bochev P, Garassi C, Robinson A. Toward an *h*-independent algebraic multigrid for Maxwell's equations. *SIAM Journal on Scientific Computing* 2006; **27**:1669–1688.
10. Briggs WL, Emden Henson V, McCormick SF. *A Multigrid Tutorial* (2nd edn). SIAM: Philadelphia, PA, 2000.
11. Trottenberg U, Oosterlee C, Schüller A. *Multigrid*. Academic Press: New York, 2001.
12. Brandt A. Algebraic multigrid theory: the symmetric case. *Applied Mathematics Computation* 1986; **19**:23–56.
13. Ruge JW, Stüben K. Algebraic multigrid. In *Multigrid Methods*, McCormick SF (ed.). Frontiers in Applied Mathematics, vol. 3. SIAM: Philadelphia, PA, 1987; 73–130.
14. Vanek P, Mandel J, Brezina M. Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems. *Computing* 1996; **56**:179–196.
15. Brezina M, Cleary AJ, Falgout RD, Henson VE, Jones JE, Manteuffel TA, McCormick SF, Ruge JW. Algebraic multigrid based on element interpolation. *SIAM Journal on Scientific Computing* 2000; **22**(5):1570–1592.
16. Jones JE, Vassilevski PM. AMGe based on element agglomeration. *SIAM Journal on Scientific Computing* 2001; **23**(1):109–133.
17. Krechel A, Stüben K. Parallel algebraic multigrid based on subdomain blocking. *Parallel Computing* 2001; **27**:1009–1031.
18. HSL. A collection of Fortran codes for large-scale scientific computation, 2007. Available from: http://www.cse.clrc.ac.uk/nag/hsl/.
19. Henson VE, Yang UM. BoomerAMG: a parallel algebraic multigrid solver and preconditioner. *Applied Numerical Mathematics* 2002; **41**:155–177.
20. Falgout RD, Yang UM. /hypre/: a library of high performance preconditioners. *Proceedings of the ICCS 2002*. Lecture Notes in Computer Science, vol. 2331. Springer: Berlin, 2002; 632–641.
21. Heroux M, Hu J, Lehoucq R, Thornquist H, Tuminaro R, Willenbring J, Bartlett R, Howle V, Kolda T, Long K, Hoekstra R, Pawlowski R, Phipps E, Salinger A, Williams A. An Overview of Trilinos. *Sandia Report*, *SAND2003-2927*, 2003.
22. Stüben K. An introduction to algebraic multigrid. In *Multigrid*, Trottenberg U, Oosterlee C, Schüller A (eds). Academic Press: New York, 2001; 413–532.
23. Brandt A. Multi-level adaptive solutions to boundary-value problems. *Mathematics of Computation* 1977; **31**: 333–390.
24. De Sterck H, Meier Yang U, Heys JJ. Reducing complexity in parallel algebraic multigrid preconditioners. *SIAM Journal on Matrix Analysis and Applications* 2006; **27**:1019–1039.
25. Stüben K. A review of algebraic multigrid. *Journal of Computational and Applied Mathematics* 2001; **128**: 281–309.

DOI: 10.1002/nme

26. Füllenbach T, Stüben K, Mijalković S. Application of algebraic multigrid solver to process simulation problems. *Proceedings of the International Conference on Simulattion of Semiconductor Processes and Devices*, Seattle, WA, U.S.A., 2000; 225–228.
27. Stüben K, Clees T. *SAMG User's Manual*. Available from: http://www.scai.fhg.de/samg/.
28. Stüben K, Delaney P, Chmakov S. Algebraic multigrid (AMG) for ground water flow and oil reservoir simulation, 2003.
29. Clees T, Stüben K. Algebraic multigrid for industrial semiconductor device simulation. *Proceedings of the 1st International Conference on Challenges*, *in Scientific Computing*. Lecture Notes in Computer Science. Springer: Berlin, 2003.
30. Füllenbach T, Stüben K. Algebraic multigrid for selected PDE systems. *Proceedings of the 4th European Conference on Elliptic and Parabolic Problems*. World Scientific: London, 2002; 399–410.
31. Notay Y, Vassilevski PS. Recursive Krylov-based multigrid cycles. *Numerical Linear Algebra with Applications* 2008; **15**:473–487.
32. Hackbusch W. *Multi-grid Methods and Applications*. Springer: Berlin, 1985.
33. Duff IS, Reid JK. MA48 a Fortran code for direct solution of sparse unsymmetric linear systems of equations. *Report RAL-93-072*, Rutherford Appleton Laboratory, 1993.
34. Duff IS, Reid JK. The design of MA48, a code for the direct solution of sparse unsymmetric linear systems of equations. *ACM Transactions on Mathematical Software* 1996; **22**:187–226.
35. Anderson E, Bai Z, Bischof C, Blackford S, Demmel J, Dongarra J, Du Croz J, Greenbaum A, Hammarling S, McKenney A, Sorensen D. *LAPACK User's Guide* (3rd edn). SIAM: Philadelphia, PA, 1999.
36. Boyle JW, Mihajlović MD, Scott JA. HSL_MI20: an efficient AMG preconditioner. *Technical Report RAL-TR-2007-021*, RAL, 2007.
37. Juel A, Mullin T, BenHadid H, Henry D. Three-dimensional free convection in molten gallium. *Journal of Fluid Mechanics* 2001; **436**:267–281.
38. Chandrasekhar S. *Hydrodynamic and Hydromagnetic Stability*, Chapter 2. Dover: New York, 1981.
39. COMSOL, *FEMLAB Version 2.3 Reference Manual*. COMSOL, 2003.
40. Dohrmann C, Bochev P. A stabilized finite element method for the Stokes problem based on polynomial pressure projections. *International Journal for Numerical Methods in Fluids* 2004; **46**:183–201.
41. Kay DA, Gresho PM, Griffiths DF, Silvester DJ. Adaptive time-stepping for incompressible flow Part II: Navier–Stokes equations. Available from: http://eprints.ma.man.ac.uk/1110/.Manchester Institute for Mathematical Sciences (MIMS), Preprint 2008.61, Manchester, U.K.
42. Arrow K, Hurwicz L, Uzawa H. *Studies in Nonlinear Programming*. Stanford University Press: Stanford, 1958.
43. Elman HC, Golub GH. Inexact and preconditioned Uzawa algorithms for saddle point problems. *SIAM Journal on Numerical Analysis* 1994; **31**:1645–1661.
44. Silvester D, Wathen A. Fast iterative solution of stabilized Stokes systems. Part II: using general block preconditioners. *SIAM Journal on Numerical Analysis* 1994; **31**:1352–1367.
45. Wathen A. Realistic eigenvalue bounds for the Galerkin mass matrix. *IMA Journal of Numerical Analysis* 1987; **7**:449–457.
46. Mihajlović M, Mijalković S. *Efficiency Study of the 'Black-box' Component Decomposition Preconditioning for Discrete Stress Analysis Problems*. Lecture Notes in Computer Science, vol. 3037. Springer: Berlin, 2004; 97–104.
47. Flynn MR, Caulfield CP. Natural ventialtion in interconnected chambers. *Jounal of Fluid Mechanics* 2006; **564**:139–158.
48. King SD. A numerical journey to the Earth's interior. *IEEE Transactions on Computing in Science and Engineering* 1995; **2**:12–23.
49. Lees MJ, Thurlow MS, Seddon JRT, Lucas PG. Convective roll dynamics in liquid ^{4}He near the onset of convection. *Physical Review Letters* 2004; **93**:144502.
50. Kaddeche S, Henry D, BenHadid H. Magnetic stabilization of the buoyant convection between infinite horizontal walls with a horizontal temperature gradient. *Journal of Fluid Mechanics* 2003; **480**:185–216.
51. Christon MA, Gresho PM, Sani RL. Computational predictability of time-dependent natural convection flows in enclosures (including a benchmark solution). *International Journal for Numerical Methods in Fluids* 2002; **40**:953–980.
52. Govaerts W. *Numerical Methods for Bifurcations of Dynamic Equilibria*. SIAM: Philadelphia, PA, 2000.
53. Drazin PG. *Introduction to Hydrodynamic Stability*. Cambridge University Press: Cambridge, 2002.
54. Gelfgat AY, Bar-Jozeph PZ, Yarin AL. Stability of multiple steady states of convection in laterally heated cavities. *Journal of Fluid Mechanics* 1999; **388**:315–334.

DOI: 10.1002/nme

55. Le Quere P, Behnia M. From onset of unsteadiness to chaos in a differentially heated square cavity. *Journal of Fluid Mechanics* 1998; **359**:81–107.
56. Winters KH. Oscillatory convection in liquid metals in a horizontal temperature gradient. *International Journal for Numerical Methods in Engineering* 1988; **25**:401–414.
57. Smethurst CA. A finite element solution of the natural convection problem in 3D. *Ph.D. Thesis*, University of Manchester, 2008.
58. Cleary AJ, Falgout RD, Henson VE, Jones JE. Coarse grid selection for parallel algebraic multigrid. *Proceedings of the 5th International Symposium on Solving Irregularly Structured Problems in Parallel*. Lecture Notes in Computer Science, vol. 1457, 1998.
59. Heil M, Hazel A. `oomph-lib`—the object-oriented multi-physics finite element library. Available from: http://www.oomph-lib.org.
60. Notay Y. An aggregation-based algebraic multigrid method. *Report GANMN 08-02*, 2009.
61. Gresho P, Sani R. *Incompressible Flow and the Finite Element Method*. Wiley: New York, 1998.

DOI: 10.1002/nme