Available online at www.sciencedirect.com
SCIENCE DIRECT®
ELSEVIER
Applied Numerical Mathematics 51 (2004) 305–327
APPLIED NUMERICAL MATHEMATICS
www.elsevier.com/locate/apnum

Variations on algebraic recursive multilevel solvers (ARMS) for the solution of CFD problems

Yousef Saad [a,*], Azzeddine Soulaimani [b], Ridha Touihri [b]

[a] *Department of Computer Science and Engineering, University of Minnesota, 4-192EE/CSci Building, 200 Union Street S.E., Minneapolis, MN 55455, USA*

[b] *Département de Génie Mécanique, École de technologie supérieure, 1100 Notre-Dame Ouest, Montréal, Québec, H3C 1K3, Canada*

Available online 11 September 2004

Abstract

This paper presents results using preconditioners that are based on a number of variations of the Algebraic Recursive Multilevel Solver (ARMS). ARMS is a recursive block ILU factorization based on a multilevel approach. Variations presented in this paper include approaches which incorporate inner iterations, and methods based on standard reordering techniques. Numerical tests are presented for three-dimensional incompressible, compressible and magneto-hydrodynamic (MHD) problems.

Keywords: ILUT; Algebraic multilevel iterations; Preconditioners; ARMS; GMRES; Flexible GMRES; Incompressible; Compressible; MHD flows

1. Introduction

Selecting a suitable solver for handling the linear systems in realistic applications can be a difficult task. On one side of the spectrum of possible choices lies a large collection of special-purpose preconditioners which are tailored to the physical problem at hand. Though these techniques may be vastly superior to contending methods that attempt to be general-purpose, their main weakness is that they are unlikely to be useful for solving other problems with different characteristics. On the other extreme of the

* Corresponding author.
E-mail addresses: saad@cs.umn.edu (Y. Saad), asoulaimani@mec.etsmtl.ca (A. Soulaimani).

0168-9274/$30.00
doi:10.1016/j.apnum.2004.06.017

spectrum, lies the well-developed class of sparse direct solvers which are the most general and reliable techniques available. It is clear that from the point of view of generality and reliability, direct methods, are the most desirable. However, these methods are prohibitive for realistic 3-dimensional problems. A middle-ground solution is to develop methods that attempt to imitate the generality and robustness of sparse direct solvers. In this regard, ILU-type preconditioners constitute the most popular choice. In this paper we further explore a class of methods in this category, which lies in between multilevel techniques and ILU-type preconditioners.

Multilevel methods can be extremely efficient for certain classes of problems. In this category, multigrid techniques can be viewed as "specialized" since their success is narrowly limited and contingent upon tuning of various parameters and operator choices. The Algebraic MultiGrid (AMG) methods, introduced in the seventies to remedy these limitations, offer an alternative whose success has been mixed since it still depends on the underlying PDE problem. In recent years, a number of preconditioners were introduced in the literature which attempt to combine attributes of the standard incomplete LU factorizations and the algebraic multigrid method. These methods possess features of multilevel methods as well as some features of ILU factorizations. ILUM [26] is one such approach and related work by Botta and co-workers [2,3], Bank and Wagner [1] and [28,29], indicates that this type of approach can be fairly robust and scales well with problem size, in contrast with standard ILU preconditioners. The idea was extended to a block version (BILUM) using dense blocks [28] and then this was further extended into BILUTM [29] which treats the diagonal blocks as sparse. Later BILUTM gave rise to the Algebraic Recursive Multilevel Solver (ARMS) [27], which offers more capabilities and a truly recursive implementation in the C programming language. On parallel platforms ARMS provides a general framework for multilevel parallel ILU preconditioning which can accommodate both fine-grain and coarse-grain parallelism [22].

This paper presents the ARMS strategy and its variations with the goal of indicating how parameters can be tuned to enhance performance of the algorithm on realistic problems from Computational Fluid Dynamics. For example, we explore the use of inner-outer combinations which consist of using Krylov accelerations at some of the levels. A few model problems from Computational fluid dynamics are then presented along with solution methodologies. These problems serve to provide realistic test cases in the numerical experiments section.

2. ARMS preconditioning and variations

A preconditioned Krylov subspace method for solving the linear system

$$Ax = b, \tag{2.1}$$

consists of an accelerator and a preconditioner. In what follows we call M the preconditioning matrix, so that, for example, the left-preconditioned system

$$M^{-1}Ax = M^{-1}b, \tag{2.2}$$

or its right-preconditioned version, $AM^{-1}y = b$, $x = M^{-1}y$ is solved instead of the original system (2.1). The above systems are solved via an "accelerator", a term used to include a number of methods of the Krylov subspace class.

One of the most common methods used to obtain a preconditioner, is via Incomplete LU factorizations, which involve an approximate Gaussian elimination process. A related class of preconditioners developed in [28,29,27] exploit the idea of independent sets or "group-independent" sets [17,23,21,12,26]. Block Independent set orderings transform the original linear system into the form

$$\begin{pmatrix} B & F \\ E & C \end{pmatrix} \begin{pmatrix} u \\ y \end{pmatrix} = \begin{pmatrix} f \\ g \end{pmatrix} \tag{2.3}$$

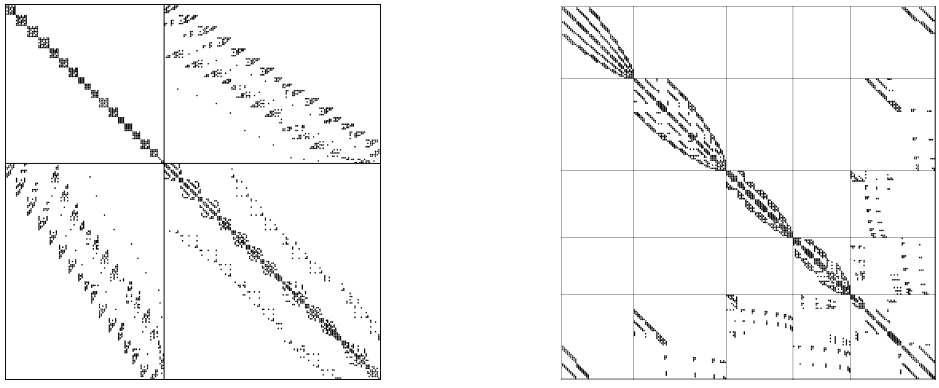
in which the B block is block diagonal. The ILUM [26] factorization utilizes standard independent set orderings which result in a matrix B that is diagonal. In this situation, it is easy to eliminate the u variable to obtain a system with only the y variable. The coefficient matrix for this 'reduced system' is the Schur complement $S = C - EB^{-1}F$ which is still sparse. This idea was used recursively, applying dropping to S to limit fill-in, and reordering the resulting reduced system into the above form via independent sets. This is repeated for a few levels until the system is small enough, or a maximum number of levels is reached. Then the system is solved by some other, standard, means such as an ILUT–GMRES combination.

In [28,29] standard independent set were generalized to *block* (or *group*) independent sets whose goal is to produce a matrix B that is block diagonal. A group-independent set is a collection of subsets (blocks) of unknowns such that there is no coupling between unknowns of any two different groups (blocks) [28]. Unknowns within the same group (block) may be coupled. If the unknowns are permuted such that those associated with the group-independent set are listed first, followed by the other unknowns, the original coefficient system will take the form (2.3) where now the matrix is not diagonal but block diagonal. An illustration is shown in Fig. 1 which shows two block independent reorderings of the same matrix.

ARMS [27] describes a scalar code which exploits block independent sets to implement a generalization of ILUM. At the lth level of ARMS, the coefficient matrix is reordered as in (2.3) where B is now diagonal and then the following block factorization is computed 'approximately' (subscripts corresponding to level numbers are introduced):

$$\begin{pmatrix} B_l & F_l \\ E_l & C_l \end{pmatrix} \approx \begin{pmatrix} L_l & 0 \\ E_l U_l^{-1} & I \end{pmatrix} \times \begin{pmatrix} U_l & L_l^{-1} F_l \\ 0 & A_{l+1} \end{pmatrix}, \tag{2.4}$$

where



(a) (b)

Fig. 1. Group-independent set reorderings of a 9-point matrix: (a) small groups (fine-grain), (b) large groups (coarse-grain).

$$B_l \approx L_l U_l, \tag{2.5}$$

$$A_{l+1} \approx C_l - \left(E_l U_l^{-1}\right)\left(L_l^{-1} F_l\right). \tag{2.6}$$

Exploiting recursivity simplifies the description of the procedure which consists essentially of two steps. First, obtain a group-independent set and reorder the matrix in the form (2.3); second, obtain an ILU factorization $B_l \approx L_l U_l$ for B_l and approximations to the matrices $L_l^{-1} F_l$, $E_l U_l^{-1}$, and A_{l+1}. The process is repeated recursively on the matrix A_{l+1} until a selected number of levels is reached. At the last level, a simple ILUT factorization, possibly with pivoting, or an approximate inverse method can be applied.

The A_i's remain sparse but become denser as the number of levels increases, so small elements are dropped in the block factorization to maintain sparsity. Note that the matrices $E_l U_l^{-1}$, $L_l^{-1} F_l$ or their approximations

$$G_l \approx E_l U_l^{-1}, \qquad W_l \approx L_l^{-1} F_l \tag{2.7}$$

which are used to obtain the Schur complement via (2.6) need not be saved. They are computed only for the purpose of obtaining an approximation to A_{l+1}. Once this is accomplished, they are discarded to save storage. Subsequent operations with $L_l^{-1} F_l$ and $E_l U_l^{-1}$ are performed using U_l, L_l and the blocks E_l and F_l. As an example of the many possible variations that can be made, examine the following forward and backward solves associated with the ARMS factorization. In the following, $\binom{f_l}{g_l}$ is a right side for the system (2.4), and $\binom{y_l}{z_l}$ the solution of the system.

Algorithm 2.1. RMLS—Recursive Multi-Level Solution:

1. Compute $g_l' = g_l - E_l B_l^{-1} f_l$;
2. Solve (recursively) the system $A_{l+1} z_l = g_l'$;
3. Back–Substitute to get y_l, i.e., solve $B_l y_l = [f_l - F_l z_l]$.

The output vector $\binom{y_l}{z_l}$ from the above procedure is the result of one preconditioning operation applied to a given input vector $\binom{f_l}{g_l}$. Step 1 is akin to a fine-to-coarse mesh restriction used in multigrid. Step 3 is the reverse operation which can be viewed as a prolongation. Step 2 is where many possible variations can arise, among which the following ones

1. An iterative process is used (recursively) in the forward/backward RMLS sweeps;
2. The preconditioner for this iterative process is defined using the next $(l + 2)$ level. This variation leads to procedures reminiscent of W-cycles in multigrid;
3. Alternatively, the local B_l block can be used in preconditioning.

The papers [29,27] describe two algorithms (BILUTM, ARMS) based on the above block factorizations. The ARMS [27] version is written in C, is fully recursive, and makes efficient use of storage. It also allows more options that the FORTRAN based BILUTM. The paper [27] also presents comparisons with other solution methods, both iterative and direct. ARMS is often superior to ILUT, which is often used as a benchmark for similar comparisons. Note that setting the number of levels equal to zero in ARMS will lead to ILUT.

2.1. Block independent set strategies

Several methods are available for obtaining block independent sets. One of the simplest techniques in this context is based on a form of reverse Cuthill–McKee ordering. A first node (called the root) is selected from which the traversal is done. In the first step of the traversal, all the neighbors of the root are traversed. These nodes constitute the first level in the traversal. Every node that has been visited is marked. At each step a traversal is made from each of the nodes of the previous level. When a certain number of nodes are reached the set of nodes gathered from the traversal will constitute one block. In essence, the procedure starts a standard reverse Cuthill–McKee procedure and stops as soon as a given number of nodes have been gathered in the current set. The nodes thus gathered will form one of the B-blocks. Within each of these B-blocks, the labeling is reversed in a way similar to the reverse Cuthill–McKee ordering.

Algorithm 2.2. Multiple root reverse Cuthill–McKee

```
1 Set visited = ∅
2 While |visited| < n do
3  Select a new root v; set set := last_level := {v}
4  While |set| < bsize:
5      For each w in last_level
6          If w is not marked
7              Add w to new_level
8              Mark w
9          EndIf
10     EndFor
11     set := set ∪ new_level
12     last_level := new_level
13 EndWhile
14 Reverse Ordering of set
15 Add all vertices of last_level to complement set
16 Select a new root from the nonmarked nodes
17 EndWhile
```

2.2. Diagonal dominance filtration

In order to improve the robustness of the factorization, the rows that are the least diagonally dominant are 'filtered out'. This strategy is simple but quite effective for highly indefinite problems.

A row is rejected to the complement set whenever a certain measure of its diagonal dominance relative to the other rows is poor. To implement this, a vector of weights w is computed as follows. First, some diagonal dominance coefficients are computed as:

$$\hat{w}(i) = \frac{|a_{ii}|}{\sum_{j=1}^{n} |a_{ij}|}.$$

Note that $0 \leqslant \hat{w}(i) \leqslant 1$ and that when $a_{ii} \neq 0$ the inverse of $\hat{w}(i)$ is $\hat{w}(i)^{-1} = 1 + \sum_{j \neq i} |a_{ij}/a_{ii}|$. These weights are close to one for strongly diagonally dominant rows, and close to zero for very non-diagonal

dominant rows. For matrices that have all of their rows far from being diagonally dominant a strategy based on using the above weights might reject all the rows. A more effective strategy is to utilize the criterion on a relative basis by normalizing all the $\hat{w}(i)$ ratios by the average or the maximum. For example, we can use instead,

$$w(i) = \frac{\hat{w}(i)}{\max_{j=1,\ldots,n} \hat{w}(i)}, \quad i = 1, \ldots, n. \tag{2.8}$$

Variations exist in which the weights combine both column and row diagonal dominance ratios.

2.3. ARMS cycles

There are many possible variations of the ARMS preconditioner once the ARMS factorization is available. One of the simplest ways to enhance the robustness of the preconditioner is to perform some form of inner iterations.

The current version of ARMS allows three such variations, which have been implemented and tested. Many other options exist, which have not been explored. All three variations solve the Schur complement system on the last level with GMRES–ILUT, but differ in the ways in which the systems on the intermediate levels are treated:

(VARMS) Descend by using the level structure. Solve system on the last level, then ascend back to current level.

(WARMS) Use a few steps of GMRES to solve the reduced system—utilizing VARMS as a preconditioner.

(WARMS*) Use a few steps of FGMRES to solve the reduced system—utilizing WARMS* (recursively) as a preconditioner.

WARMS* was discussed in the early technical report version of [27]. However, this algorithm turns out to be fairly expensive for most practical situations. The VARMS preconditioning step is shown in the following algorithm.

Algorithm 2.3. $x_l :=$ VARMS-solve(A_l, b_l)—Recursive Multi-Level Solution

0. Let $b_l = \binom{f_l}{g_l}$
1. Solve $L_l \; f_l' = f_l$
2. Descend, i.e., compute $b_{l+1} := g_l - E_l U_l^{-1} \; f_l'$
3. If $l =$ *last_lev* then
4. Solve $A_{l+1} y_l = b_{l+1}$ using GMRES + ILU factors
5. Else
6. $y_l =$ VARMS-solve(A_{l+1}, b_{l+1})
7. EndIf
8. Ascend, i.e., compute $f_l'' = f_l' - L_l^{-1} F_l \; y_l$
9. Back–Substitute $u_l = U_l^{-1} f_l''$
10. Return $x_l := \binom{u_l}{y_l}$

Note that in the analogous WARMS-solve algorithm, line 6 would be simply replaced by the line:

6w. Solve $A_{l+1}y_l = b_{l+1}$ using GMRES preconditioned by VARMS$(A_{l+1}, *)$).

In [27] the question was addressed on how to perform the matrix–vector products with the matrices A_i without having to save these matrices. These operations are required whenever an iterative process is used at the intermediate or last levels. To obviate the need to store Schur complement matrices, we can compute $S_l \times w$ as

$$S_l \times w = \left(C_l - E_l B_l^{-1} F_l\right) w. \tag{2.9}$$

This version uses matrices from the current level instead of the next one and saves storage, most often at the expense of additional arithmetic operations. The inverse of B_l is applied by using its (incomplete) LU factors.

2.4. Comparison with AMG

ARMS can be viewed as a form of Algebraic Multi-Grid (AMG) solver [5] so it is important to comment on the similarities and the differences between the two approaches. Research on AMG is still very active, with current work generally focussed on improving the robustness and applicability of AMG, see, for example, [4,8,10,11,18].

From one viewpoint, the main distinction between a multilevel ILU-based solver and an AMG-based solver resides mainly in how accurately the Schur complements, i.e., the Coarse-mesh problems, are constructed. ILU methods obtain these via an approximate Gaussian elimination process. In contrast, AMG constructs rough approximations to these Schur complements via restriction operators based either on the actual mesh, or a virtual one obtained from considering "strong connections" in the graph, see [5].

ILU-type preconditioners, including ARMS, aim at being "general-purpose". These methods typically sacrifice efficiency by not exploiting any physical information on the problem. In contrast, AMG methods attempt to reproduce the excellent efficiency of multigrid, precisely by taking advantage of physical information or any analogy with physical situations.

For the reasons mentioned above, one should, generally speaking, expect AMG and multigrid solvers to be superior to ILU-type preconditioner (including ARMS) for problems that exhibit a more elliptic nature and arise from single variable problems. AMG can easily exploit knowledge on the physical problem, specifically the geometry (mesh). Thus, the HYPRE package [9] provides several "conceptual interfaces" to enable the user to exploit such knowledge. The most general of these interfaces is the so-called "IJ" interface, which takes a general sparse matrix and a right side, as do ARMS and ILUT. However, note that similar options could as well be provided in ARMS, using a "coarsening" based on the mesh rather than block-independent sets. The inclusion of such options would move ARMS closer to standard AMG solvers.

On the other hand, one should expect ILU-type solvers (ARMS and ILUT) to be more robust than AMG solvers for problems that are highly indefinite and/or correspond to models with several unknowns per grid-point. Here too, AMG can be moved closer to an ILU-type preconditioner by, for example, using more complex smoothers, such as ILUT, to improve robustness. This will likely make AMG behave more like ARMS or ILUT in some cases and multigrid in others.

In the end the capabilities of the two approaches should be similar once more options are added to balance the conflicting needs of robustness on the one hand and efficiency on the other. A complete comparison of the two approaches is beyond the scope of this paper. It way well be the case that eventually the

lines dividing computational codes based on multilevel ILUs and AMGs will get blurred as researchers will add features to improve efficiency, applicability, and robustness.

3. Model problems and solution strategy

Several three-dimensional CFD model problems are used for testing the proposed preconditioners. Incompressible MHD flows and compressible gas flows are solved using stabilized finite element formulations. These considered test problems present some challenges for iterative methods. In this section, the governing equations and the finite element formulations used are briefly presented.

Let Ω be a bounded domain of $\Re^{nd}$ ($nd = 3$) with boundary $\Gamma = \partial\Omega$. The outward unit vector normal to Γ is denoted by $\mathbf{n}$. Throughout this paper, we consider a partition of the domain Ω into elements Ω^e where piecewise continuous approximations for the independent variables are employed. We will use the standard notation: a subscript h to a function denotes a finite element approximation.

3.1. Magnetohydrodynamic flows

Given a flow field, the conservative magnetohydrodynamic system of equations is obtained from the Maxwell equations, and is written as:

$$\frac{\partial \mathbf{B}}{\partial t} - \nabla \times (\mathbf{u} \times \mathbf{B}) - \frac{1}{Re_m}\nabla \times (\nabla \times \mathbf{B}) + \nabla q = 0, \tag{3.1}$$

$$\nabla \cdot \mathbf{B} = 0, \tag{3.2}$$

where Re_m, $\mathbf{B}$, $\mathbf{u}$ and q are respectively the magnetic Reynolds number, magnetic induction field, velocity field and the scalar Lagrange multiplier for the magnetic free divergence constraint [31,30]. For coupled magnetohydrodynamics, this system is solved along with the incompressible Navier–Stokes equations where the body force is $\mathbf{f} = \frac{1}{\mu}(\nabla \times \mathbf{B}) \times \mathbf{B}$. This represents the Lorentz (Laplace) force due to the interaction between the current density $\mathbf{j} = \frac{1}{\mu}(\nabla \times \mathbf{B})$ and the magnetic field, where μ is the magnetic permeability. Since in real applications the magnetic-Reynolds number is small, the discrete variational formulation used for the magnetic problem is a simple stabilized Galerkin formulation [31] and is stated as: find $(\mathbf{B}_h, q_h)$ such that for all weighting functions $(\mathbf{B}_h^*, q_h^*)$ one has,

$$\begin{aligned}
&\int_\Omega \mathbf{B}_h^* \cdot \frac{\partial \mathbf{B}_h}{\partial t}\,\mathrm{d}\Omega + \frac{1}{Re_m}\int_\Omega \nabla \mathbf{B}_h^* \cdot \nabla \mathbf{B}_h\,\mathrm{d}\Omega - \int_\Omega \mathbf{B}_h^* \cdot \nabla \times (\mathbf{u} \times \mathbf{B}_h)\,\mathrm{d}\Omega \\
&\quad + \int_\Omega \mathbf{B}_h^* \cdot \nabla q_h\,\mathrm{d}\Omega + \left(\tau_2 - \frac{1}{Re_m}\right)\int_\Omega (\nabla \cdot \mathbf{B}_h)\big(\nabla \cdot \mathbf{B}_h^*\big)\,\mathrm{d}\Omega \\
&\quad - \frac{1}{Re_m}\int_\Gamma \big((\mathbf{n} \cdot \nabla \mathbf{B}_h) \cdot \mathbf{B}_h^* - \big(\mathbf{n} \cdot \mathbf{B}_h^*\big)(\nabla \cdot \mathbf{B}_h)\big)\,\mathrm{d}\Gamma \\
&\quad + \int_\Omega q_h^*(\nabla \cdot \mathbf{B}_h)\,\mathrm{d}\Omega + \sum_{\Omega_e \in \Gamma_h} \varepsilon_2 \int_{\Omega_e} \nabla q_h \cdot \nabla q_h^*\,\mathrm{d}\Omega = 0.
\end{aligned} \tag{3.3}$$

The parameter ε_2 is a function of the mesh size h and local magnetic Reynolds number ($Re\,m_h = \|\mathbf{u}\|h\,Re_m$) given by $\varepsilon_2 = h^2\,Re_m/\sqrt{Re\,m_h^2+4}$. The parameter τ_2 is a penalty factor given by $\tau_2 = \|\mathbf{u}\|h/2$. The case of fully-coupled MHD problems is described in [30] where a coupling algorithm between magnetic and fluid fields is developed.

3.2. Compressible gas flows

The Navier–Stokes equations are solved using a finite element method [24] in terms of the conservative variables $\boldsymbol{V} = (\rho, \boldsymbol{U}, E)^t$, i.e., the density, the vector momentum and the specific total energy. In the case of turbulent flows we consider the one equation high Reynolds numbers Spalart–Allmaras turbulence model [33], the vector of unknown variables is then $\boldsymbol{V} = (\rho, \boldsymbol{U}, E, \nu_t)^t$ where ν_t is the turbulent kinematic viscosity. The governing system of equations is written in a compact form as

$$\boldsymbol{V}_{,t} + \boldsymbol{F}_{i,i}^{\text{adv}}(\boldsymbol{V}) = \boldsymbol{F}_{i,i}^{\text{diff}}(\boldsymbol{V}) + \boldsymbol{\mathcal{F}}, \tag{3.4}$$

where $\boldsymbol{F}_i^{\text{adv}}$ and $\boldsymbol{F}_i^{\text{diff}}$ are, respectively, the convective and diffusive fluxes in the ith-space direction, and $\boldsymbol{\mathcal{F}}$ is the source vector. Lower commas denote partial differentiation and repeated indices indicate summation. The diffusive fluxes can be written

$$\boldsymbol{F}_i^{\text{diff}} = \boldsymbol{K}_{ij}\boldsymbol{V}_{,j}$$

while the convective fluxes can be represented by diagonalizable Jacobian matrices $\boldsymbol{A}_i = \boldsymbol{F}_{i,V}^{\text{adv}}$. Recall that any linear combination of these matrices has real eigenvalues and a complete set of eigenvectors.

It is well known that the standard Galerkin finite element formulation often leads to numerical instabilities for convective dominated flows. In the Galerkin-Least-Squares (GLS) method, or the generalized Streamline Upwind Petrov Galerkin (SUPG) method, [20,19] the Galerkin variational formulation is modified to include an integral form depending on the local residual $\boldsymbol{\mathcal{R}}(\boldsymbol{V})$ of Eq. (3.4), i.e., $\boldsymbol{\mathcal{R}}(\boldsymbol{V}) = \boldsymbol{V}_{,t} + \boldsymbol{F}_{i,i}^{\text{adv}}(\boldsymbol{V}) - \boldsymbol{F}_{i,i}^{\text{diff}}(V) - \boldsymbol{\mathcal{F}}$, which is identical to zero for the exact solution. The SUPG formulation reads : find $\boldsymbol{V}$ such that for all weighting functions $\boldsymbol{W}$,

$$\sum_e \int_{\Omega^e} \left[\boldsymbol{W}\cdot\left(\boldsymbol{V}_{,t} + \boldsymbol{F}_{i,i}^{\text{adv}} - \boldsymbol{\mathcal{F}}\right) + \boldsymbol{W}_{,i}\boldsymbol{F}_i^{\text{diff}}\right]\mathrm{d}\Omega - \int_{\Gamma} \boldsymbol{W}.\boldsymbol{F}_i^{\text{diff}} n_i\,\mathrm{d}\Gamma + \sum_e \int_{\Omega^e} \left(\boldsymbol{A}_i^t\boldsymbol{W}_{,i}\right)\cdot\boldsymbol{\tau}\cdot\boldsymbol{\mathcal{R}}(\boldsymbol{V})\,\mathrm{d}\Omega = 0. \tag{3.5}$$

In this formulation, the matrix $\boldsymbol{\tau}$ is referred to as *the matrix of time scales*. The SUPG formulation is built as a combination of the standard Galerkin integral form and a perturbation-like integral form depending on the local residual vector [19]. The objective is to reinforce the stability *inside the elements*. The matrix of time scales used here is given by $\boldsymbol{\tau} = (\sum|\boldsymbol{B}_i|)^{-1}$, for $i = 1, nd$ where $\boldsymbol{B}_i = \frac{\partial Re_{mi}}{\partial x_j}\boldsymbol{A}_j$ and $\frac{\partial Re_{mi}}{\partial x_j}$ are the components of the element Jacobian matrix [24]. The SUPG formulation aims at adding an amount of artificial viscosity in the characteristic directions. Since these formulations lead to a high-order scheme, high frequency oscillations in the vicinity of shocks or stagnation points can occur. A shock-capturing viscosity depending on the discrete residual $\boldsymbol{\mathcal{R}}(\boldsymbol{V})$ as proposed in [24] is employed. More dissipation is then added in high gradient zones to avoid any undesirable local oscillations.

3.3. Solution algorithms for nonlinear problems

The solution of partial differential equations of fluid dynamics by any time and space discretization method often leads to solving a nonlinear problem in the Euclidean space $\Re^n$:

$$F(x) = 0. \tag{3.6}$$

The preferred strategy for solving (3.6) is Newton's method which generates a sequence of approximations of the form

$$x_{i+1} = x_i + \delta_i, \tag{3.7}$$

starting from a certain initial approximation, $x_0 \in \Re^n$. The solution update δ_i at step i is the solution of the linear system

$$J(x_i)\delta_i = -F(x_i), \tag{3.8}$$

where $J(x_i)$ denotes the Jacobian associated with F and evaluated at x_i.

A solution strategy often used to solve many CFD problems consists of two nested loops, the first one over time and the second over Newton's iteration. This can be sketched as follows

Algorithm 3.1. General solution strategy

1. Do $i = 1$, *TimeSteps*
2. Compute and factor the Jacobian matrix
3. (or an approximation) when needed
4. Do $k = 1$, *NewtonIterations*
5. Compute the residual $F(x_i)$,
   Solve the system (3.8) and
6. Update the solution by (3.7)
7. Check convergence
8. EndDo
9. EndDo

In the exact Newton method, A is the true Jacobian matrix $J(x_i)$. This matrix, however, is not always available or computable analytically. It may also be expensive to recompute at every iteration, as is generally the case in CFD problems. It is often preferable to compute an approximation of the Jacobian once and use it for a prescribed number of time steps and Newton iterations. In our codes, this matrix is used to construct the preconditioner M. The action of the Jacobian on a vector (i.e., the matrix-by-vector product Az_j required at the jth step of any Krylov-subspace method) is often approximated using a finite difference quotient such as:

$$Az_j \approx \big[F(x_0 + \varepsilon z_j) - F(x_0)\big]/\varepsilon \quad \text{or} \quad Az_j \approx \big[F(x_0 + \varepsilon z_j) - F(x_0 - \varepsilon z_j)\big]/(2\varepsilon),$$

where ε is an appropriate small number. Simple formulas for computing ε are, respectively, $\varepsilon = \varepsilon_0(\|x_0\| + \varepsilon_0)$ (with $\varepsilon_0 = 10^{-6}$) and $\varepsilon = \varepsilon_0/\|z_j\|$ (with $\varepsilon_0 = 10^{-2}$).

Two issues of concern at each time step are the convergence of Newton's iterations and the cost of solving the linearized problem (3.8). For steady-state problems, another concern is the global convergence

of the method. In practice, two different difficulties may be encountered, the divergence of Newton's iterations and the stagnation of the global loop.

As is well known, Newton's method will converge only if the initial guess x_0 is close enough to the solution. To enhance the global convergence characteristics of Newton's method, a number of so-called global convergence strategies are often utilized, see, e.g., [14] for details. The simplest of these, called 'damped' Newton, replaces the update (3.7) by

$$x_{i+1} = x_i + \lambda_i \delta_i, \tag{3.9}$$

in which the damping factor λ_i is less than one and selected so as to achieve sufficient reduction in the form of $\|F(x_i + \lambda_i \delta_i)\|_2$. Trust-region methods are often preferred. Independent of these improvements, the above algorithm may encounter a number of other difficulties related to the time-stepping procedure or the linear solver. Possible cures can be found by using one, or a combination, of the following strategies:

(a) reduce the time step,
(b) select the initial guess carefully,
(c) improve the robustness of the linear system solver by, e.g., increasing the Krylov-space dimension or using a better preconditioner,
(d) use stabilization techniques for discretization,
(e) use multigrid methods or adaptive meshing techniques, etc.

This paper focuses on option (c) which relies on enhancing the preconditioner. In all nonlinear CFD problems considered in this paper the *non-linear version* of FGMRES(m), also referred to as a Jacobian-free nonlinear FGMRES, is employed. Refs. [6,7,13] discuss inexact and Jacobian-free Newton methods, and [32] provides some details on the use of flexible GMRES (FGMRES) within the context of such methods, in CFD.

4. Numerical experiments

The goal of the numerical tests which are reported in this section is to help understand the variations of the ARMS procedure as well as to show the effect of the parameters on performance. Another goal is to compare the performance of the ARMS strategy with a threshold based ILU preconditioner (ILUT) for various problems in CFD. We consider both the performance in the solutions of the linear systems extracted from the nonlinear iterative procedure and for the nonlinear iteration itself.

All experiments have been conducted on the IBM SP at the University of Minnesota using a single processor "Winterhawk" node. In all tests except those of 4.2, the standard version VARMS is used. Also, unless explicitly stated, no inner iterations are performed at the last level (and intermediate levels.)

4.1. Effect of the number of levels

The parameters `bsize` (block-size) and `nlev` (number of levels) are intimately related. The larger the block size, the smaller is the size of the Schur complement relatively to that of the original system.

Therefore, fewer levels will be needed to reach a final Schur complement that is of small enough size. In this experiment we explore the effect of the number of levels on the performance of ARMS.

The linear system considered in this test is extracted from the simulation of a laminar compressible flow at Mach $= 0.5$ and Reynolds $= 10\,000$. The geometry is that of a flat profile and the mesh uses mixed tetrahedras and hexahedras. The matrix size is $n = 122\,321$ and the number of nonzero elements is $nnz = 6\,751\,053$. The parameters used for the experiment are listed in Table 1.

In the table, bsize denotes the block-size selected for the block-independent sets. The actual block-size may differ from the input value of bsize which acts as a threshold—a block is accepted as soon as its size is larger than bsize in the traversal. The subscript I indicates that the parameter is used for the intermediate matrices, while S refers to Schur complements, and LU refers to the incomplete factorization for the last Schur complement. The incomplete factorization process uses two parameters, an integer fill_* which sets the maximum number of nonzeros kept in each row of the matrix being constructed, and a parameter droptol_* which removes entries that are smaller than the threshold droptol_* relative to a certain initial norm of the row. A parameter indexed with I refers to the ILU factorization of each B-matrix at each level as well as for the construction of the temporary matrices $\widetilde{G}$, $\widetilde{W}$, see, e.g., Eqs. (2.7). Thus, the numbers fill_I (intermediate matrices), fill_S (Schur complements) and fill_LU (LU factorization, last level) indicate the maximum number of entries kept when constructing each of the matrices involved. Similarly, the parameter droptol_LU is for the ILU factorization of the last Schur complement. The parameter tol_DD is the threshold value used for the relative diagonal dominance criterion discussed in Section 2.2, see Eq. (2.8). The number of levels nlev is the maximum number of levels allowed. The actual number of levels found may differ from the input value of nlev. Fig. 2 shows the number of iterations and the iteration time required to solve the linear systems when the number of levels varies.

Table 1

bsize	nlev	fill_I	fill_S	fill_LU	droptol_I	droptol_S	droptol_LU	tol_DD
1000	varies	50	40	40	0.001	0.01	0.01	0.1

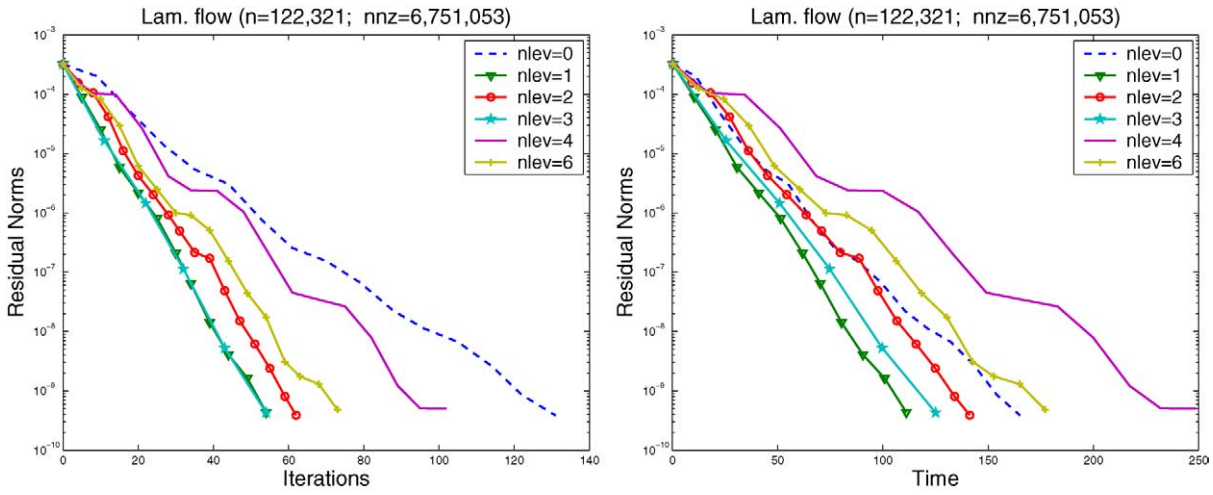


Fig. 2. Residual norm vs number of iterations (left) and iteration time (right) for various number of levels.

These experiments seem to indicate that there is no simple rule to determine whether or not increasing the number of levels will help in reducing the overall run time. This is perhaps due to the fact that the number of levels is tied to other choices. However, it is easy to see that if there is substantial dropping at each level, the Schur complements that are constructed become less accurate and therefore, more levels will harm convergence. In this case it is best to use one level. This is more or less the situation with the method parameters used in this experiment. It is also worth noting that using nlev $= 0$ (equivalent to ILUT) is often not competitive relative to ARMS. In this particular test, ILUT does still fairly well in terms of overall time. It is better than using 4 or 6 levels for example. In terms of memory used, it is remarkable that in all the tests with ARMS (nlev $\geqslant 1$), the fill factor (i.e., the ratio of the number of nonzero elements in the preconditioning over that of the matrix) is about the same, around 1.62. For ILUT this ratio is smaller and close to 1.27. However, as the next experiment shows, ARMS can do well with a much smaller fill-ratio in contrast with ILUT, which seems to require more fill to yield a successful preconditioner.

4.2. Effect of inner iterations and iteration cycles

One of the simplest ways to enhance the robustness of ARMS is to perform some form of inner iterations. The current version of ARMS allows three such variations which were described in Section 2.3. In the next test, we experiment with the WARMS variant (in this variant inner iterations are performed only at the last level and the top level). The number of inner iterations allowed at these levels is varied. Tests have been made with nlev $= 1$ and nlev $= 2$. The other parameters are as follows (see Table 2).

Table 2

bsize	fill_I	fill_S	fill_LU	droptol_I	droptol_S	droptol_LU	tol_DD
1000	20	20	30	0.01	0.01	0.01	0.1

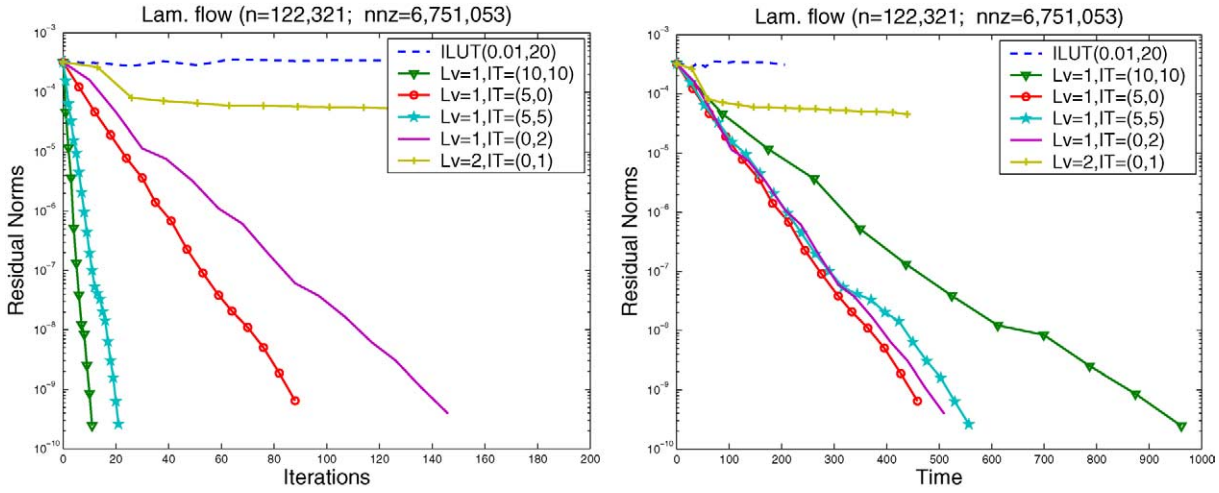


Fig. 3. Residual norm vs number of iterations (left) and iteration time (right) for various inner-outer iteration combinations. The first integer in `IT` indicates the number of GMRES iterations at the top level, while the second is the number if GMRES iterations at the last level.

Table 3

bsize	nlev	fill_I	fill_S	fill_LU	droptol_I	droptol_S	droptol_LU	tol_DD
50	2	50	40	40	0.001	0.01	0.01	varies

As can be seen very loose dropping criteria are selected, so that the resulting factorization is not accurate. The performance of the ILUT factorization using similar parameters is shown for comparison. For the ILUT preconditioner, the fill factor, i.e., the factor of the number of nonzero elements used by the LU factorization over the number of nonzero elements in A, is small, 0.67. For those tests using nlev $= 1$, the fill factor is around 0.79. Thus, the storage requirement in this case is quite modest. The ILUT preconditioner uses less storage, but a slightly more accurate version yielded similar results (stagnation). Results with a few choices of the inner iteration parameters are shown in Fig. 3. The label $IT = (K_1, K_2)$ for each curve shows the number of inner iterations used for the first level (K_1) and the last (K_2).

It is expected that as the number of inner iterations for the first level and the last increases, the number of global FGMRES iterations should decrease. This will tend to enhance robustness of the global scheme. However, because of dropping, the last level Schur complement matrix is not likely to be accurate and, as a result, one expects that solving the last Schur complement system accurately is unlikely to be cost-effective. In the same vein using more levels may become less effective in this situation. This has been verified and it is one of the reasons why there are few plots in Fig. 3 with nlev > 1. Another observation, is that using only one level can be very effective in reducing the overall execution time. It is rather remarkable that convergence can still be reached with a factorization that is so inaccurate.

4.3. Effect of diagonal dominance filtration

It was mentioned earlier that diagonal dominance filtration is a simple yet very effective technique to improve robustness for highly indefinite matrices. In the next experiment we show how the choice of tol_DD can affect performance. It has been our experience that it always pays to take a small positive value for tol_DD, i.e., to reject a (relatively) small percentage of rows with poor diagonal dominance, to the complement set. We consider again the same matrix issued from the laminar flow example seen in the previous two tests. The parameter tol_DD is varied from 0.0 to 0.15. Beyond $\text{tol}_\text{DD} = 0.15$, GMRES preconditioned with ARMS encounters difficulties in achieving convergence. In this test no inner iterations are used at any level. The other parameters employed for the test are shown in Table 3.

The plots in Fig. 4 show a marked improvement as tol_DD increases from zero, reaching the best overall performance for $\text{tol}_\text{DD} = 0.1$. Then, as tol_DD approaches 0.15, the trend reverses itself. For tol_DD larger than 0.15 a poor factorization is obtained. The trend of rapid improvement of the convergence as tol_DD moves away from zero is observed quite often.

4.4. Simulation of Ponomarenko MHD dynamo instability

The Ponomarenko dynamo instability [25] is generated by a solid-body helical motion $\mathbf{u} = (u_r, u_\theta, u_z)$ of an incompressible fluid in a cylinder ($r \leqslant R_1$), surrounded by a cylindrical medium ($R_1 \leqslant r \leqslant R_2$). Both materials are electroconducting and characterized by their electroconductivity (respectively, σ_1 and σ_2) and by their magnetic permeability (respectively, μ_1 and μ_2). The velocity field is defined as $\mathbf{u} = (0, \omega r, U_z)$ with respect to cylindrical coordinates (r, θ and z). The pich of the solid-body helical

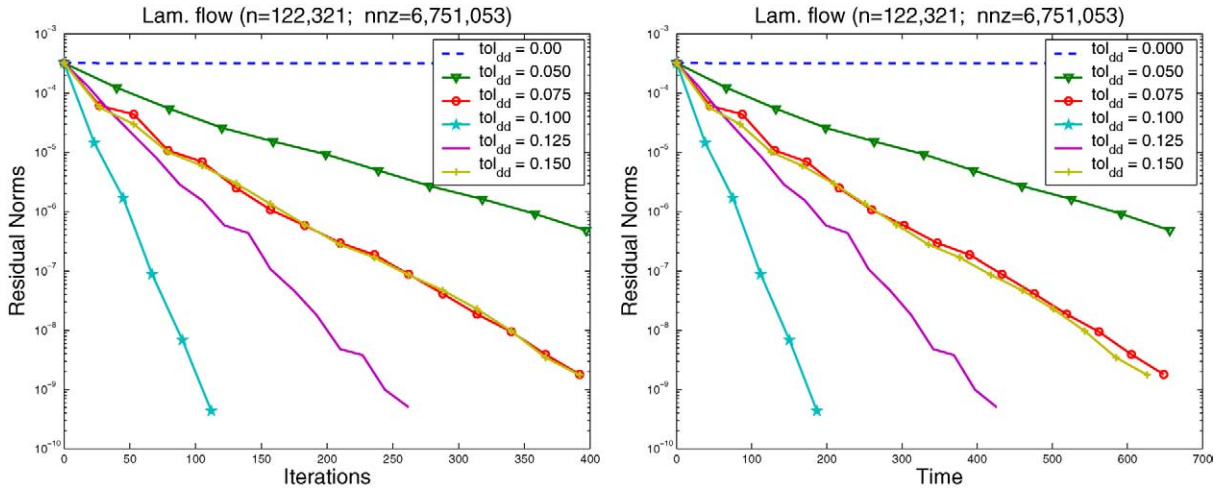


Fig. 4. Residual norm vs number of iterations (left) and iteration time (right) for various values of $\mathrm{tol}_{\mathrm{DD}}$.

Table 4

bsize	nlev	$\mathrm{fill}_{\mathrm{I}}$	$\mathrm{fill}_{\mathrm{S}}$	$\mathrm{fill}_{\mathrm{LU}}$	$\mathrm{droptol}_{\mathrm{I}}$	$\mathrm{droptol}_{\mathrm{S}}$
3000	5	nnz/n	nnz/n	nnz/n	0.0005	0.0005

motion is defined as $\chi = \frac{U_z}{u_\theta(r=R_1)} = \frac{U_z}{R1\omega}$. Here, ω is the angular velocity. For the kinetic MHD problem, the flow is supposed to be steady so ω and U_z are constant. The magnetic field $\mathbf{B}$, is the solution of the MHD equations (3.1), (3.2). There exists a critical magnetic Reynolds number Rem at which the magnetic field becomes unsteady. This critical value can be captured using a direct numerical solution, i.e., solving MHD equations in a time-accurate fashion and for a long period of time. Since this problem is linear, the preconditioner is built only at the first time step. The performance of the preconditioner is measured in terms of the memory and CPU time consumed. The results shown in this section correspond to the physical parameters $\chi = 1.3$ and $\sigma_1 = \sigma_2$. The initial magnetic field is given by: $\mathbf{B}(\mathbf{t} = \mathbf{0}) = (P(r)\cos(\theta), -(P(r) + rP'(r))\sin(\theta), P(r)\cos(\theta))$, where $P(r) = (r - R_2)^2$. No-slip boundary conditions $\mathbf{B} = \mathbf{0}$ are imposed on the lateral surfaces ($r = R_2$) and the periodicity is imposed between the two faces $z = 0$ and $z = H$ with $H = 4\chi\pi$. The mesh used contains 33297 nodes and 158720 tetrahedras, see Fig. 5.

The matrix size and number of nonzero elements are $n = 125\,052$ and $nnz = 6\,354\,832$, respectively. Direct numerical simulations were performed using a nondimensional time step of 0.15 and at least 2000 time steps. A critical magnetic Reynolds number was found at $\mathrm{Rem} = 17.5$ for our simulation, to be compared with the critical value of $\mathrm{Rem} = 17.7$ found by Gailitis [16]. Fig. 6 shows the history of total magnetic energy. There is an instant when this energy starts to increase. This means that the magnetic field extracts energy from the flow field. For this test problem, we compared ILUT and ARMS preconditioners using the following parameters (see Table 4).

In order to obtain a time-accurate solution, the convergence criterion for GMRES iterations is that the residual norm be reduced by a factor of 0.0005. A direct simulation of 2000 time steps takes almost 9 hours of CPU time on a $2Ghz$ PC machine. We found that the ARMS preconditioner saves almost 10%

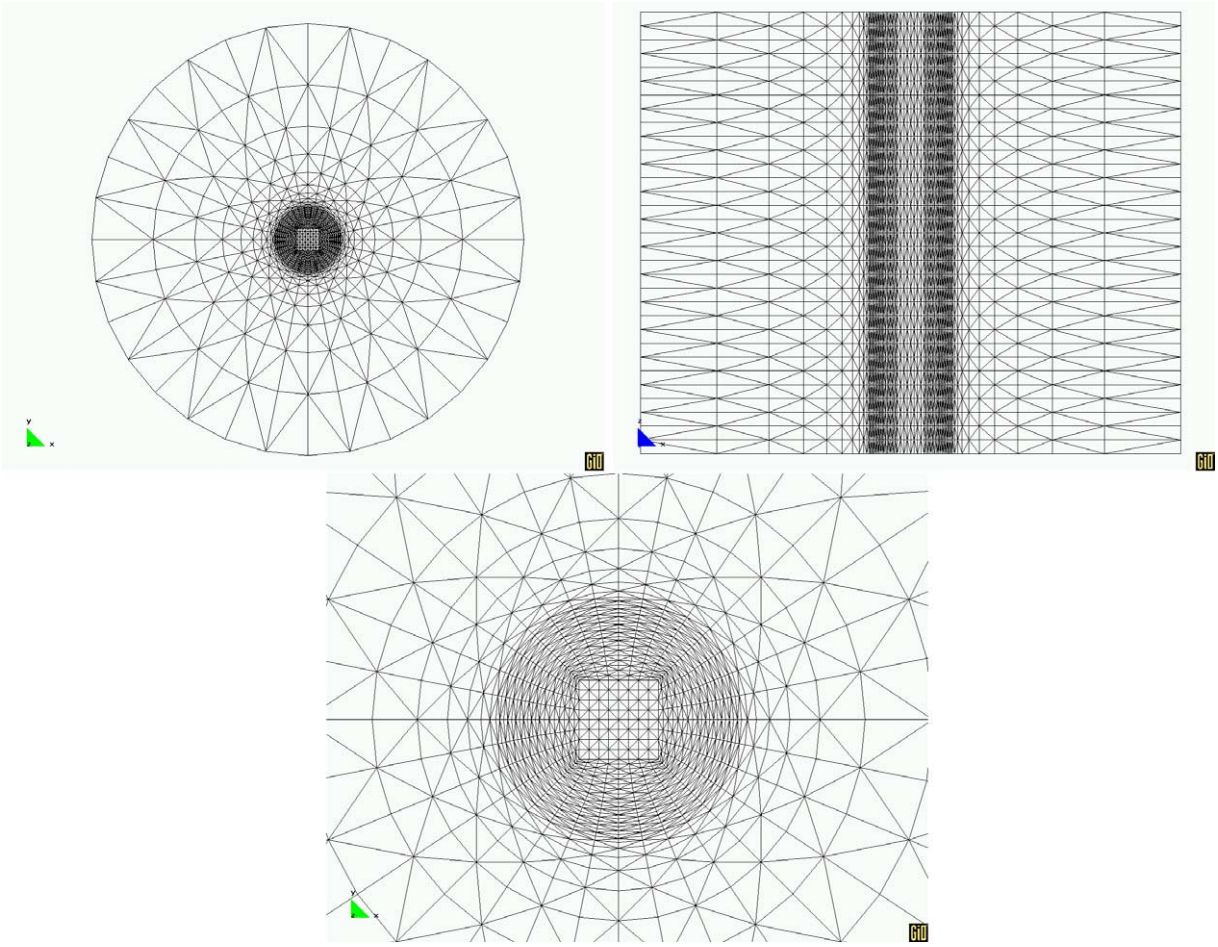


Fig. 5. Mesh used for the Ponomarenko MHD test.

of CPU time as compared to ILUT while using almost the same amount of memory. The histories of the residual norm (Fig. 6) for ARMS and ILUT are undistinguishable. Finally, Fig. 7 shows isocontours of the magnetic energy. This is well concentrated around radius R_1 in a helical shape.

4.5. Tests on compressible flows

We consider the problem of a flow over a flat plate using a relatively fine mesh in the neighborhood of the plate. It is well known that some convergence difficulties can be encountered when looking for a steady solution of a low Mach number (Mach $\leqslant 0.2$), especially when using the conservative variables as the dependent unknowns. In the case of slightly compressible flows, one can use other sets of variables [15] employing the pressure as the dependent variable instead of the density. Another procedure uses what is called 'a preconditioning' of the Navier–Stokes equations where the time dependent term $\boldsymbol{V}_{,t}$ is multiplied by a preconditioning matrix P aimed at damping the spurious sound waves. In this

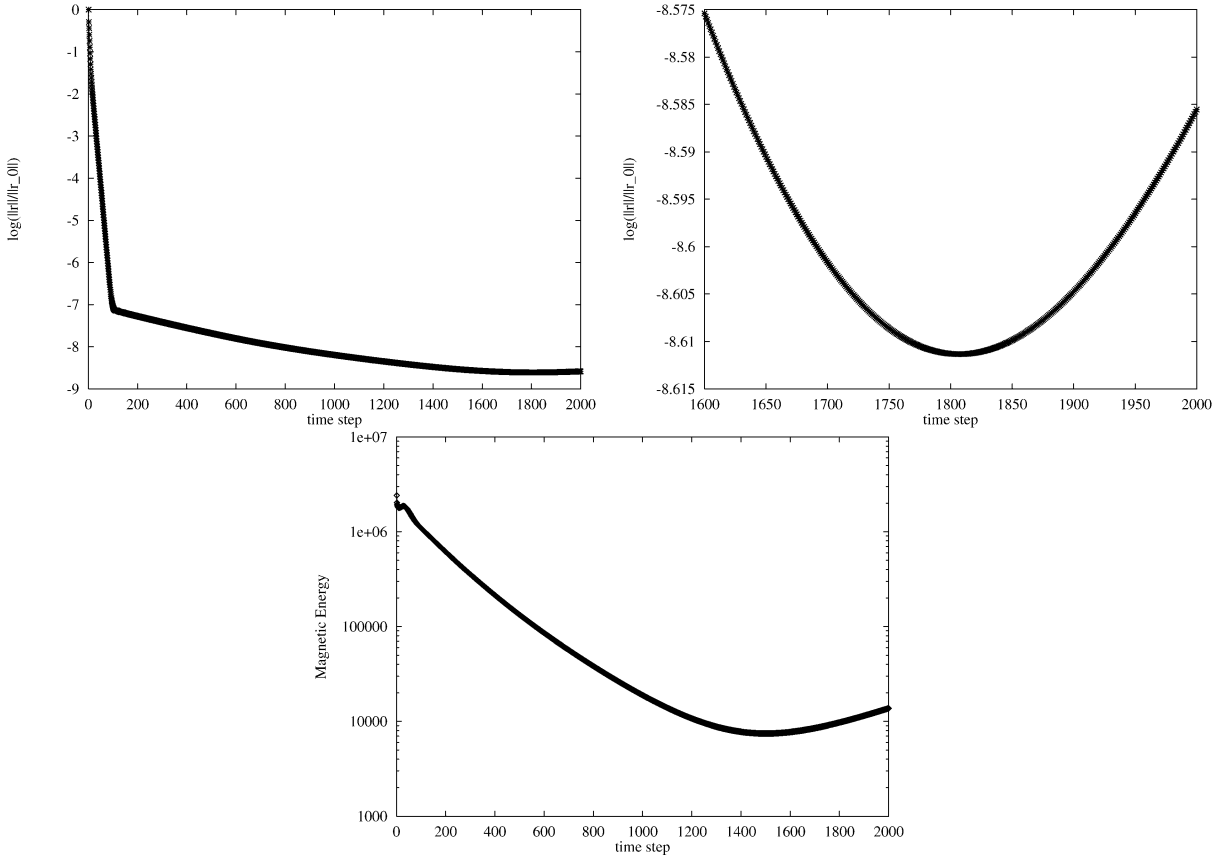


Fig. 6. Ponomarenko MHD test, history of the residual norm with a zoom around the instability (top) and magnetic energy (bottom).

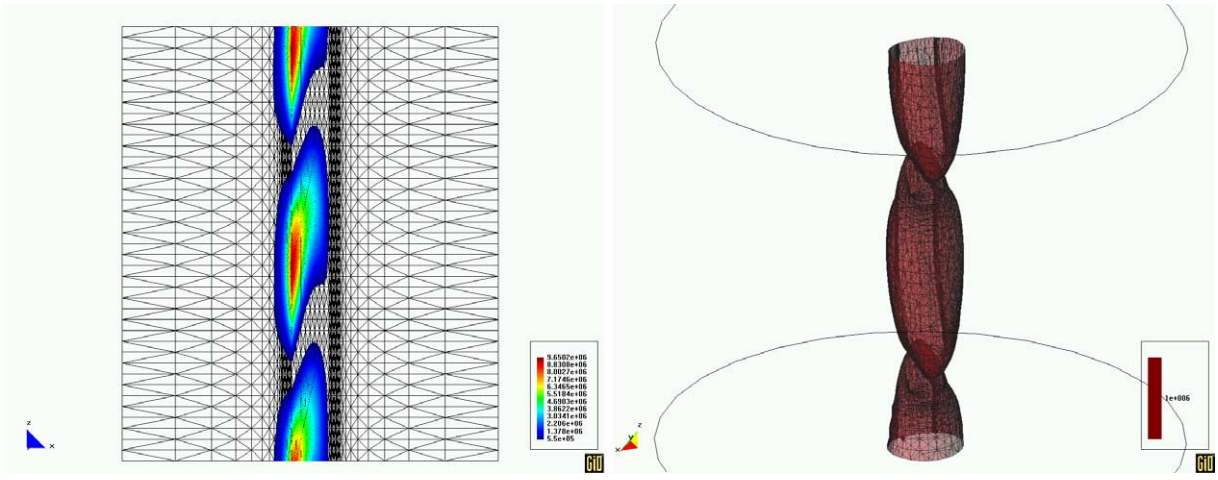


Fig. 7. Ponomarenko MHD test. Isocontours of the magnetic energy.

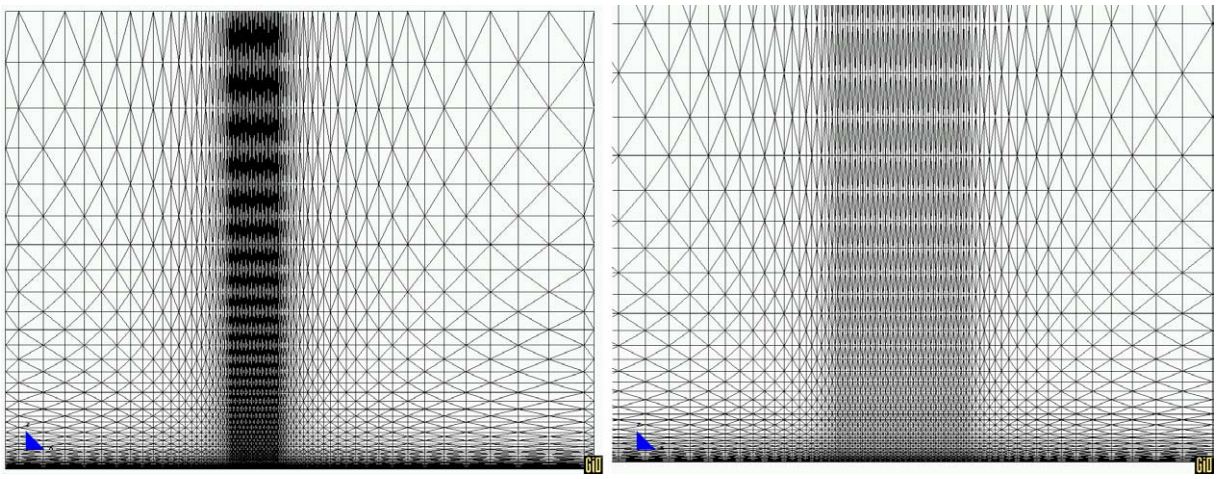


Fig. 8. Mesh over a flat plate, close view (right).

Table 5

bsize	fill_I	fill_S	fill_LU	droptol_I	droptol_S
500	nnz/n or 2 nnz/n	nnz/n or 2 nnz/n	nnz/n or 2 nnz/n	0.0001	0.001

work we prefer to use the same finite element method for all flow regimes and we rely on the algebraic preconditioners and the stabilization techniques to speed up the iterative procedure.

The mesh consists of mixed tetrahedras and hexahedras with a total number of elements equal to 150 064. The first nodes are located at a distance of 5×10^{-4} from the plate (Fig. 8). The resulting matrix has a size $n = 174\,701$ and a total of $nnz = 9\,630\,337$ nonzero elements. We first consider a flow at a Mach number of 0.2 and at a Reynolds number of 10^4. The parameters used for this experiment are listed in Table 5.

As can be seen from this table, the fill-in parameters of the preconditioner have been related to the matrix parameters n and nnz, using some heuristic simple formulas. For the case of nlev $= 0$, the preconditioner corresponds to ILUT. Regarding the ARMS preconditioner, several tests showed that a good value of nlev is 5. A smaller value generates a more stable but much more expensive preconditioner. For this problem, bsize is 500. A larger value makes the preconditioner more expensive to build. It was observed also that the fill-in parameters have a big impact on the quality of the preconditioner and on the memory used. The solution procedure uses the time-stepping approach. After every 5 time steps the preconditioner is rebuilt and the time step is increased by an amount of 0.1 (i.e., we used a variable nondimensional time step Δt according to $\Delta t = \Delta t + 0.1$). Fig. 9 shows typical convergence histories. It is clearly shown that the ILUT preconditioner with a fill-in nnz/n tends to stagnate while for the same fill-in the ARMS based preconditioner behaves fairly well. However, doubling the fill-in parameter makes the ILUT preconditioner behave almost identically to the ARMS preconditioner. This remark generally holds for many tests performed so far on various kind of CFD problems. ILUT can compete with ARMS but requires more fill-in.

On the other-hand, the fill-in parameters have a direct impact on the memory used. Although the ARMS preconditioner requires a smaller fill-in than ILUT, it uses some small additional memory to keep

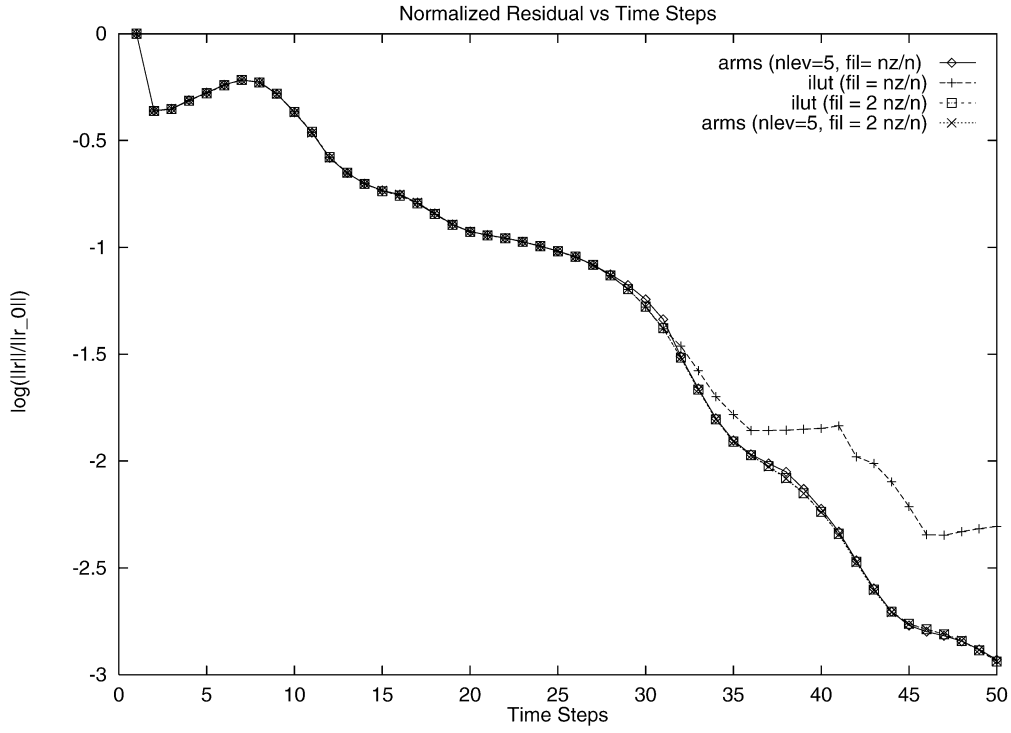


Fig. 9. Residual norm vs number of time steps for various number of levels.

the multilevel structure. As mentioned earlier, a decision was made to sacrifice some computation to save memory, in keeping this structure. Specifically, the matrices G_l and W_l in (2.7) are discarded once the Schur complement given by (2.6) is computed. The products with the matrices W_l and G_l are performed by using L_l, F_l and E_l, U_l, respectively, which are saved. The disadvantage of this approach is that it may entail a nonnegligible additional operation count when the matrices L_l and U_l are large. For the ARMS preconditioner, we found that a fill-in of nnz/n is sufficient but for ILUT we recommend $2nnz/n$. In terms of CPU time, ARMS is found to be almost 15 percent more expensive than ILUT for the same convergence quality. However, this slight superiority of ILUT over ARMS is not always observed. Other tests show that ARMS can be faster than ILUT depending on how often the preconditioner is rebuilt, and on the difficulty of the problem. Indeed, some CPU time saving can be obtained in favor of ARMS if the number of Krylov directions used per time step is found much lower than that required by ILUT. Fig. 10 shows the profile of the nondimensional longitudinal velocity at few stations on the plate ($5.0 \leqslant x \leqslant 6.0$) and at station $x = 7$. They are compared to the Blasius profile which is the solution for an incompressible flow at zero-pressure gradient.

We performed another test using the same geometry and the same mesh, in the transonic regime, i.e., the Mach number was set to 0.85 and the Reynolds number was $Re = 10^4$. The parameters used for this test are listed in Table 6:

The time marching procedure is used with a variable *local time step* (i.e., the element-time step is computed according to $\Delta t = CFL\frac{h^2}{h\lambda+2/Re}$, where h measures the element size, λ the maximum wave speed and *CFL* the Courant number which increases after every 10 time steps $CFL = CFL + 1$). The

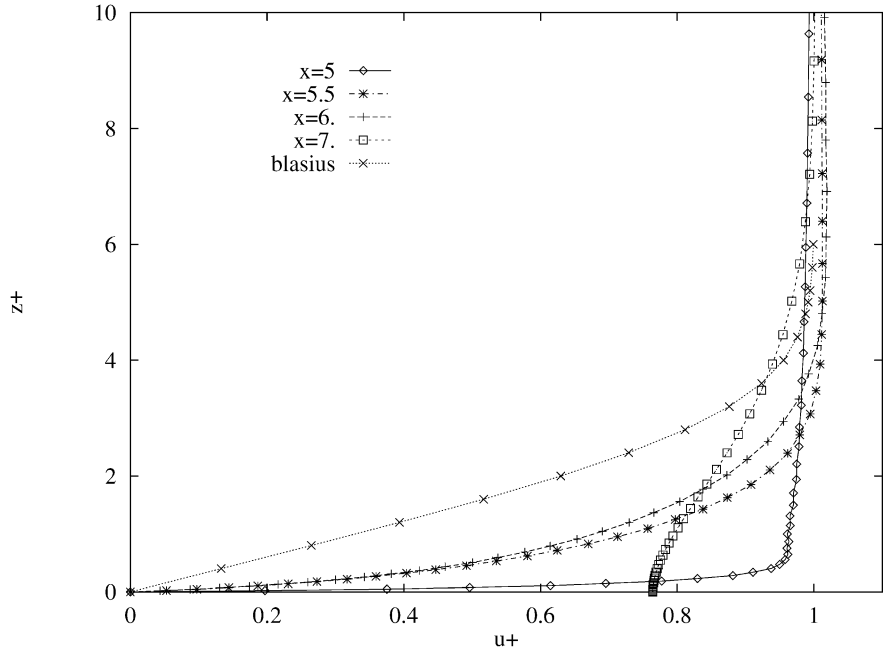


Fig. 10. Nondimensional longitudinal velocity profile at different stations.

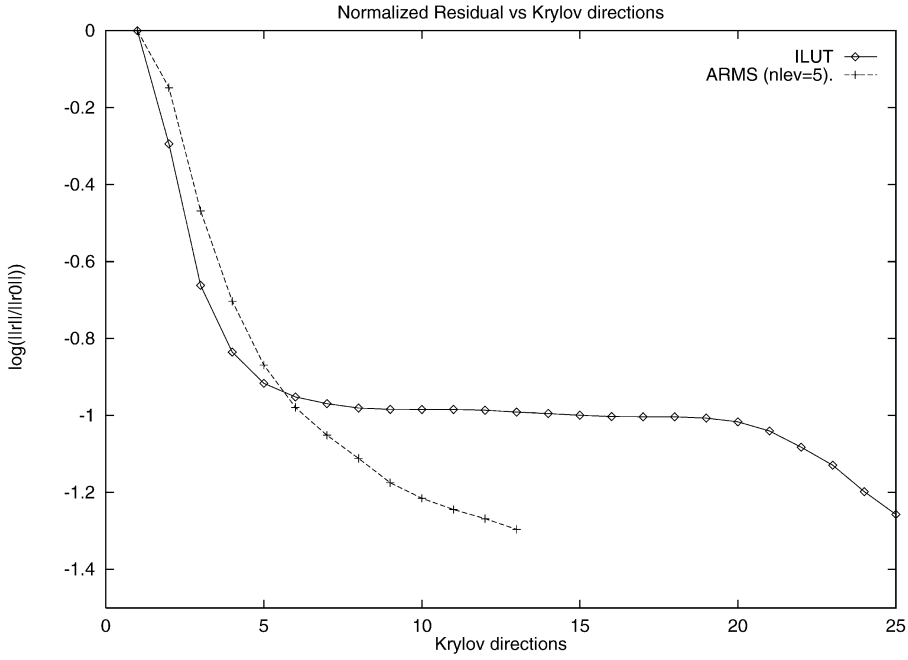


Fig. 11. FGMRES convergence for time step number 20 (Free-stream Mach number = 0.85).

Table 6

bsize	nlev	fill_I	fill_S	fill_LU	droptol_I	droptol_S
2000	nlev	$2^*nz/n$	$2^*nz/n$	$2^*nz/n$	0.0001	0.0001

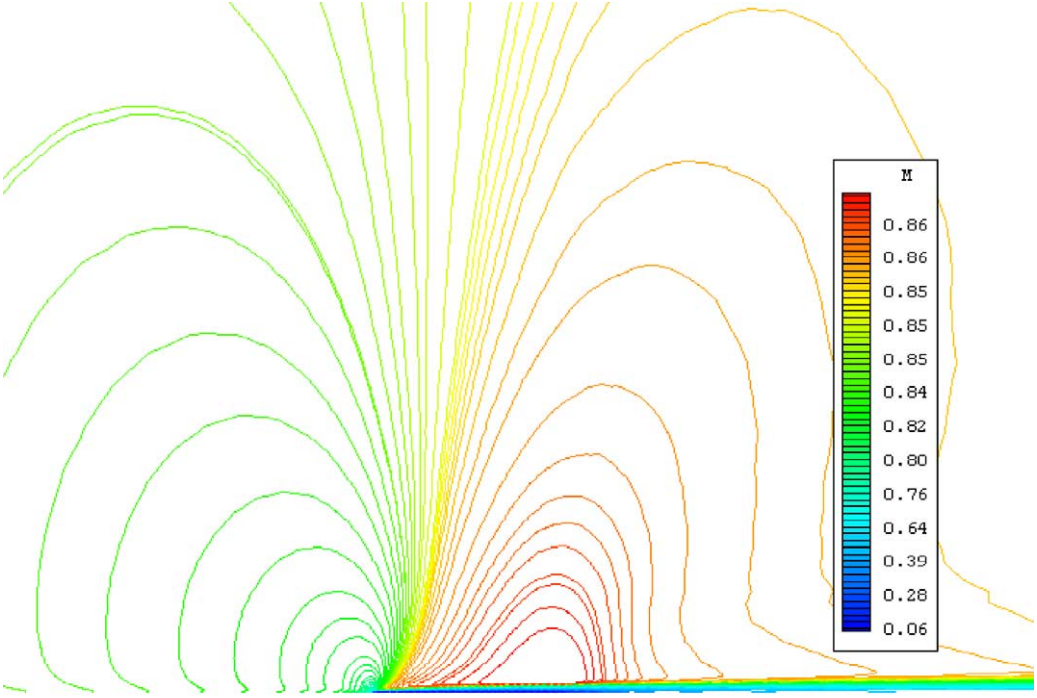


Fig. 12. Iso-Mach contours (Free-stream Mach number = 0.85).

initial flow field is obtained for a converged solution at a Reynolds number of $Re = 10^2$. At each time step, only one Newton iteration is allowed, the matrix-free FGMRES algorithm is used with a maximum Krylov subspace dimension of 40 and the inner iteration is stopped when the Euclidean norm of the residual is reduced by a factor of 0.05. We observed that this test case is particularly demanding on the iterative solver, so the fill-in parameters were set to $2nnz/n$. The global convergence (up to 200 time steps where the residual was reduced to three-order of magnitude) for ILUT and ARMS are almost identical, but the FGMRES algorithm takes many more directions to converge with ILUT as compared to when the ARMS preconditioner is used. Fig. 11 shows an example of the convergence history for time step number 20. Finally, Fig. 12 shows the iso-Mach contours for this test case.

Conclusion

We have shown various ways of using preconditioning techniques derived from the Algebraic Recursive Multilevel Solver framework, for dealing with linear systems arising in realistic flow problems. There is virtually no limit in the number of possibilities available in this framework. The main ingredi-

ents are a means of selecting nodes to leave for the coarse level, a reordering of the nodes that are kept for the fine level, and various dropping techniques for constructing the next Schur complement. We have seen that performing inner iterations can make up for an inaccurate preconditioner and reestablish convergence in some difficult cases. Another important strategy used in ARMS is to filter the least diagonal dominant equations to the end of the system. We have illustrated the importance of this strategy.

The comparison with ILUT on a set of physical problems reveals that ARMS is often superior for low-memory preconditioners but that it yields a similar performance when accurate factorizations are sought.

Acknowledgements

The work of Azzeddine Soulaimani and Rida Touihri has been supported by the Natural Sciences and Engineering Research Council of Canada (NSERC). Yousef Saad acknowledges support from the Army Research Office under contract DA/DAAD19-00-1-0485, the NSF under grants ACI-0305120, INT-0003274, and the Minnesota Supercomputing Institute. The authors would like to thank Masha Sosonkina for her help in comparing ARMS with the Algebraic Multigrid method (AMG) on the IBM SP.

References

[1] R.E. Bank, C. Wagner, Multilevel ILU decomposition, Numer. Math. 82 (4) (1999) 543–576.
[2] E.F.F. Botta, A. van der Ploeg, F.W. Wubs, A fast linear-system solver for large unstructured problems on a shared-memory computer, in: O. Axelsson, B. Polman (Eds.), Proceedings of the Conference on Algebraic Multilevel Methods with Applications, 1996, pp. 105–116.
[3] E.F.F. Botta, F.W. Wubs, Matrix Renumbering ILU: An effective algebraic multilevel ILU, SIAM J. Matrix Anal. Appl. 20 (1999) 1007–1026.
[4] M. Brezina, A. Cleary, R. Falgout, V.E. Henson, J. Jones, T. Manteuffel, S. McCormick, J. Ruge, Algebraic multigrid based on element interpolation (AMGe), SIAM J. Sci. Comput. 22 (5) (2000) 1570–1592.
[5] W.L. Briggs, V.E. Henson, S.F. Mc Cormick, A Multigrid Tutorial, second ed., SIAM, Philadelphia, PA, 2000.
[6] P.N. Brown, A.C. Hindmarsh, Matrix-free methods for stiff systems of ODEs, SIAM J. Numer. Anal. 23 (1986) 610–638.
[7] X.-C. Cai, W.D. Gropp, D.E. Keyes, R.G. Melvin, D.P. Young, Parallel Newton Krylov Schwarz algorithms for thr transonic full potential equation, SIAM J. Sci. Statist. Comput. 19 (1998) 246–265.
[8] E. Chow, An unstructured multigrid method based on geometric smoothness, Numer. Linear Algebra Appl. 10 (2003) 401–422.
[9] E. Chow, A. Cleary, R. Falgout, HYPRE User's manual, version 1.6.0, Technical Report UCRL-MA-137155, Lawrence Livermore National Laboratory, Livermore, CA, 1998.
[10] E. Chow, P.S. Vassilevski, Multilevel block factorizations in generalized hierarchical bases, Numer. Linear Algebra Appl. 10 (2002) 105–127.
[11] A.J. Cleary, R.D. Falgout, V.E. Henson, J.E. Jones, T.A. Manteuffel, S.F. McCormick, G.N. Miranda, J.W. Ruge, Robustness and scalability of algebraic multigrid, SIAM J. Sci. Comput. 21 (2000) 1886–1908.
[12] T.A. Davis, A Parallel Algorithm for Sparse Unsymmetric LU Factorizations, Ph.D. Thesis, University of Illinois at Urbana Champaign, Urbana, IL, 1989.
[13] R.S. Dembo, S.C. Eisenstat, T. Steihaug, Inexact Newton methods, SIAM J. Numer. Anal. 18 (2) (1982) 400–408.
[14] J.E. Dennis, R.B. Schnabel, Numerical Methods for Unconstrained Optimization and Nonlinear Equations, Prentice-Hall, Englewood Cliffs, NJ, 1983.
[15] N.E. Elkadri, A. Soulaïmani, C. Deschenes, A finite element formulation of compressible flows using various sets of independent variables, Comput. Methods Appl. Mech. Engrg. 181 (2000) 161–189.

[16] A.K. Gailitis, The helical mhd dynamo, in: Moffatt, Tsinober (Eds.), Topological Fluid Mechanics, Proceeding of the IUTAM Symposium, Cambridge, August 13–18, 1989.
[17] J.A. George, J.W.-H. Liu, Computer Solution of Large Sparse Positive Definite Systems, Prentice-Hall, Englewood Cliffs, NJ, 1981.
[18] M. Griebel, D. Oeltz, M.A. Schweitzer, An algebraic multigrid method for linear elasticity, SIAM J. Sci. Comput. 25 (2003) 385–407.
[19] T.J.R. Hughes, L.P. Franca, Hulbert, A new finite element formulation for computational fluid dynamics: VIII. The Galerkin/least-squares method for advective–duffusive equations, Comput. Methods Appl. Mech. Engrg. 73 (1989) 173–189.
[20] T.J.R. Hughes, P. Mallet, A new finite element formulation for computational fluid dynamics: III. The generalized streamline operator for multidimensional advective–diffusive systems, Comput. Methods Appl. Mech. Engrg. 58 (1986) 305–328.
[21] R. Leuze, Independent set orderings for parallel matrix factorizations by Gaussian elimination, Parallel Comput. 10 (1989) 177–191.
[22] Z. Li, Y. Saad, M. Sosonkina, pARMS: A parallel version of the algebraic recursive multilevel solver, NLAA 10 (2003) 485–509.
[23] M. Luby, A simple parallel algorithm for the maximum independent set problem, SIAM J. Comput. 14 (4) (1986) 1036–1053.
[24] A. Soulaïmani, M. Fortin, Finite element solution of compressible viscous flows using conservative variables, Comput. Methods Appl. Mech. Engrg. 118 (1994) 319–350.
[25] Y.B. Ponomarenko, On the theory of hydromagnetic dyanmo, Zh. Prikl. Mekh. Tekhn Fiz (USSR) 6 (1973) 47–51.
[26] Y. Saad, ILUM: a multi-elimination ILU preconditioner for general sparse matrices, SIAM J. Sci. Comput. 17 (4) (1996) 830–847.
[27] Y. Saad, B. Suchomel, ARMS: An algebraic recursive multilevel solver for general sparse linear systems, Numer. Linear Algebra Appl. 9 (2002).
[28] Y. Saad, J. Zhang, BILUM: Block versions of multi-elimination and multi-level ILU preconditioner for general sparse linear systems, SIAM J. Sci. Comput. 20 (1999) 2103–2121.
[29] Y. Saad, J. Zhang, BILUTM: A domain-based multi-level block ILUT preconditioner for general sparse matrices, SIAM J. Matrix Anal. Appl. 21 (1999) 279–299.
[30] N. Ben Salah, A. Soulaïmani, W.G. Habashi, A finite element method for magneto-hydrodynamics equations, Comput. Methods Appl. Mech. Engrg. 190 (2001) 5867–5892.
[31] N. Ben Salah, A. Soulaïmani, W.G. Habashi, M. Fortin, A conservative stabilized finite element method for the magneto-hydrodynamics equations, Internat. J. Numer. Methods Fluids 29 (1999) 535–554.
[32] A. Soulaimani, N.B. Salah, Y. Saad, Enhanced GMRES acceleration techniques for some CFD problems, Internat. J. CFD 16 (1) (2002) 1–20.
[33] P.R. Spalart, S.R. Allmaras, A one-equation turbulence model for aerodynamic flows, Rech. Aérosp. 1 (1994) 5–21.