Journal of Computational Physics 303 (2015) 222–237

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

Recycling Krylov subspaces for CFD applications and a new hybrid recycling solver

Amit Amritkar [a,b,*], Eric de Sturler [b], Katarzyna Świrydowicz [b], Danesh Tafti [a], Kapil Ahuja [c,1]

[a] *Department of Mechanical Engineering, Virginia Tech, United States*
[b] *Department of Mathematics, Virginia Tech, United States*
[c] *Computer Science and Engineering, IIT Indore, India*

A R T I C L E   I N F O

Article history:
Received 5 January 2015
Received in revised form 4 September 2015
Accepted 22 September 2015
Available online 28 September 2015

Keywords:
Linear solver
Krylov subspace recycling
CFD
Preconditioner
Recycling GCROT
Recycling BiCGStab

A B S T R A C T

We focus on robust and efficient iterative solvers for the pressure Poisson equation in incompressible Navier–Stokes problems. Preconditioned Krylov subspace methods are popular for these problems, with BiCGStab and GMRES(m) most frequently used for nonsymmetric systems. BiCGStab is popular because it has cheap iterations, but it may fail for stiff problems, especially early on as the initial guess is far from the solution. Restarted GMRES is better, more robust, in this phase, but restarting may lead to very slow convergence. Therefore, we evaluate the rGCROT method for these systems. This method recycles a selected subspace of the search space (called recycle space) after a restart. This generally improves the convergence drastically compared with GMRES(m). Recycling subspaces is also advantageous for subsequent linear systems, if the matrix changes slowly or is constant. However, rGCROT iterations are still expensive in memory and computation time compared with those of BiCGStab. Hence, we propose a new, hybrid approach that combines the cheap iterations of BiCGStab with the robustness of rGCROT. For the first few time steps the algorithm uses rGCROT and builds an effective recycle space, and then it recycles that space in the rBiCGStab solver.
We evaluate rGCROT on a turbulent channel flow problem, and we evaluate both rGCROT and the new, hybrid combination of rGCROT and rBiCGStab on a porous medium flow problem. We see substantial performance gains for both the problems.

1. Introduction

The current study focuses on iterative solvers for computational fluid dynamics (CFD) applications on structured grids, solving the Navier–Stokes (N–S) and energy equations. For very large problems direct methods are impractical, and even for moderate three-dimensional problems they are typically more expensive than iterative methods in work and storage. This is the case even for systems with many right hand sides; see, e.g., [1].

* Corresponding author at: Department of Mechanical Engineering, Virginia Tech, 445 Goodwin Hall, 635 Prices Fork Road - MC 0238, Blacksburg, VA 24061.
E-mail addresses: amritkar@vt.edu (A. Amritkar), sturler@vt.edu (E. de Sturler), kswirydo@vt.edu (K. Świrydowicz), dtafti@exchange.vt.edu (D. Tafti), kahuja@iiti.ac.in (K. Ahuja).
1 Worked on this project when graduate student at Virginia Tech. Currently Assistant Professor at IIT Indore, India.

http://dx.doi.org/10.1016/j.jcp.2015.09.040
0021-9991/© 2015 Elsevier Inc. All rights reserved.

The N–S equations are nonlinear partial differential equations (PDEs) that are commonly discretized in space using the Finite Volume Method (FVM), and occasionally the Finite Element Method (FEM). The pressure–velocity coupling frequently requires special consideration. This has given rise to various coupled and decoupled solution algorithms, where the choice of nonlinear solution algorithm leads to particular systems of linear equations. In terms of computational cost, the solution of the pressure Poisson equation is the most critical, and incompressible N–S solvers deal with this equation in several ways. The pressure Poisson equation is elliptic in nature but leads to nonsymmetric matrices for non-Cartesian meshes or due to certain boundary conditions. Such systems are typically solved by preconditioned Krylov subspace methods [2] with a suitable preconditioner. For symmetric positive definite systems on regular meshes, multigrid preconditioners or multigrid methods are advised, but for nonsymmetric systems the case is less clear. In this paper, we use relatively simple multilevel preconditioners that are particularly suited for implementation on graphical processing units (GPUs); see below for further details. More general multilevel preconditioners or multigrid methods are interesting next steps. However, these are not considered here.

We solve the N–S equations at discrete time steps to capture the transient flow characteristics in unsteady flows or to converge to a steady state solution, leading to a sequence of linear systems of equations. For these systems, originating from a Newton-type iteration, there is little or no change in the coefficient matrix (A), and only the right hand side (b) changes for subsequent time steps. In this case, recycling a judiciously selected subspace of the search space typically reduces the number of iterations substantially. This is referred to as *Krylov subspace recycling* [3]. There are several algorithms that take advantage of recycling for subsequent systems as well as for restarts of generalized minimal residual (GMRES) type methods, such as the generalized conjugate residual method with inner orthogonalization (GCRO) [4,5], GCRO with deflated restarting (GCRODR) [3], GMRES with deflated restarting (GMRES-DR) [6] (only for a single system) and GCRO with optimal truncation (GCROT) [7]. For a sequence of nonsymmetric linear systems, bi-Lanczos-based recycling algorithms, like recycling biconjugate gradient stabilized (rBiCGStab) [8], have the advantage of a short-term recurrence, and hence low storage requirements and fewer orthogonalizations compared with GMRES variants.

Krylov subspace recycling for sequences of systems has been sporadically applied to structured grid CFD calculations. Several studies in the area of aerodynamic shape optimization show that Krylov subspace recycling with simplified GCROT(m, k) [9] improves convergence [10,11]. In a study by Carpenter et al. [12], using an enriched GMRES method with subspace recycling for a long sequence of linear systems for steady convection–diffusion problems and flow over a wind turbine, solver parameter tuning yields improved convergence and even eliminates stagnation in some cases. For CFD matrices from the Harwell—Boeing Sparse Matrix Collection, the GCRO-DR algorithm is accelerated by avoiding small skip angles [13]. A study by Meng et al. [14] shows that block GCROT(m, k) performs better than block GMRES(m) for most of the Poisson and convection–diffusion problems. There are some cases where recycling has not worked particularly well. A study by Mohamed et al. [15] observed poor performance of GCRO-DR compared with GMRES(m) for both restarting and sequences of systems for a variety of aerodynamic flows. These previous efforts on Krylov recycling in CFD applications all concentrate on variants of GMRES. Here, we extend recycling for CFD applications to the bi-Lanczos-based method rBiCGStab, and we propose a new hybrid method as well. The new hybrid method exploits a simplified and more efficient version of the rBiCGStab algorithm derived in [8]. This more efficient version made possible by restricting the type of recycling. In this paper, we first demonstrate the advantages of Krylov subspace recycling for a turbulent channel flow problem by comparing the results of BiCGStab, GMRES(m) and recycling GCROT (rGCROT). Next, we introduce a novel method for building the outer recycle space for the rBiCGStab algorithm. We use rGCROT to build the recycle space and then switch to rBiCGStab using the same recycle space for subsequent systems. This hybrid method combines the robustness of rGCROT in the initial phase of the simulation, when the starting residual (r_0) is large, with the economy of BiCGStab iterations when r_0 drops as the flow develops. We test our hybrid method on the pressure Poisson equation for a porous media flow problem with 2.56 million unknowns. We believe this is the first application of preconditioned Bi-Lanczos based recycling algorithms to CFD.

Below, we use rGCROT(m, k) to denote the main parameters, m and k, giving the dimensions of the inner Krylov space (as for GMRES(m)) and the outer recycle space, respectively. In the next section, we briefly outline the CFD applications and the GenIDLEST package, and we discuss the preconditioners and linear solvers. Section 3 describes our CFD test problems. Results from numerical experiments are discussed in Section 4 and conclusions in Section 5.

2. Methodology

In this section, we present a brief overview of the GenIDLEST package and the CFD applications it is used for, along with details of the linear solvers used. We briefly discuss Krylov subspace recycling and we suggest an improved algorithm for rBiCGStab.

2.1. GenIDLEST

The GenIDLEST (**Gen**eralized **I**ncompressible **D**irect and **L**arge-**E**ddy **S**imulations of **T**urbulence) code used in this study solves the incompressible, time-dependent, Navier–Stokes and energy equations in a generalized structured body-fitted multiblock framework. It has been extensively used in propulsion, biological flows, and energy related applications that involve complex multi-physics flows [16]. GenIDLEST has various turbulence modeling capabilities including Reynolds Average Navier–Stokes (RANS), Detached Eddy Simulation (DES) and Large Eddy Simulation (LES) with subgrid stresses modeled using the dynamic Smagorinsky eddy viscosity model [17]. The code uses a finite volume formulation with fractional step algorithm using the semi-implicit Adams–Bashforth/Crank–Nicolson method or a fully-implicit Crank–Nicolson method for a predictor step. The corrector step solves a pressure Poisson equation to satisfy mass continuity [18]. The non-dimensional pressure Poisson equation in generalized coordinates has the following form in GenIDLEST [18],

$$\frac{\partial}{\partial \xi_j}\left(\sqrt{g}g^{jk}\langle\frac{\partial}{\partial \xi_k}p^{n+1}\rangle\right) = \frac{1}{\Delta t}\frac{\partial\langle\sqrt{g}\tilde{U}^j\rangle}{\partial \xi_j}, \tag{1}$$

where $\sqrt{g}$ is the Jacobian of the spatial transformation, g^{jk} is the contravariant metric tensor, p^{n+1} is the pressure at time level $n+1$ (with density absorbed in the pressure), Δt is the time step and $\langle\sqrt{g}\tilde{U}^j\rangle$ is the conserved contravariant flux based on an intermediate velocity $\tilde{u}_i$. Using finite volume discretization, this Poisson equation is transformed into a linear system, $Ax = b$, with pressure as the unknown. The pressure coefficient matrix (A) consists of geometric quantities from the left hand side of (1). The right hand side (b) is evaluated from a local balance of the intermediate volume fluxes at cell faces surrounding a finite volume. The current study focuses on the solution of linear system arising from (1).

The GenIDLEST code spans over 300,000 lines and more than 600 subroutines, and it has modules for the Arbitrary Lagrangian–Eulerian (ALE) Method [19], the Discrete Element Method (DEM) [20], the Immersed Boundary Method (IBM) [21], and for Fluid Structure Interaction (FSI) [22]. The computational algorithms are optimized to take maximum advantage of state-of-the-art hierarchical memory and parallel architectures, with a focus on Message Passing Interface (MPI) and OpenMP-based codes for central processing units (CPUs) [23] and more recently for GPUs [24].

Data layout

GenIDLEST uses a non-staggered grid topology; so, all the unknowns are defined and calculated at the computational cell center except the contravariant volume fluxes, which are defined and calculated at cell faces. The coefficient matrix is stored in structured banded sparse storage format (also diagonal storage) [25, Section 11.5]. For every unknown, only the coefficients for the 7 or 19 point stencil, for orthogonal or non-orthogonal grids respectively, are stored. Although this format (potentially) stores coefficients more than once, it avoids building the stencil more than once and has contiguous memory storage patterns for efficient utilization of CPU cache memory and vectorization. If orthogonal meshes are used, and the domain decomposition leads to matching interfaces at mesh block boundaries, the matrices from (1) are symmetric positive definite and conjugate gradient method (CG) is well-suited. In all other cases, the system matrices are nonsymmetric and nonsymmetric linear solvers must be used. The sparsity pattern of the matrices depends on the boundary conditions. In GenIDLEST, the matrix from the pressure Poisson equation changes only when there is ALE/IBM grid movement or for IBM related boundary flux correction.

Preconditioners in GenIDLEST

Preconditioners are essential for Krylov subspace methods, as they greatly improve the rate of convergence for many problems. In some cases, they are required for convergence. Preconditioners are the most expensive operation in the GenIDLEST framework. Thus a good choice for the preconditioner is imperative.

GenIDLEST uses domain decomposition-based preconditioners that have been optimized for parallel [26] execution on CPUs and, more recently, on GPUs [27,24]. Optimizations for serial implementations have also been done [28], but here we focus on preconditioners that run fast in parallel. In particular for high performance on GPUs, it is important to have a preconditioner tuned for the architecture [29]. For preconditioning, we partition the domain into many overlapping subdomains, and we use an additive or multiplicative Schwarz method.

The subdomain equations are solved inexactly using a few (local) iterations of Jacobi or SSOR methods. The substructured overlapping subdomain blocks are well-suited to exploit SIMD parallelism and the hierarchical memory of CPUs. Global coupling between distant subdomains is provided by Jacobi iteration on the global problem [26].

Since the forward- and backward substitution in SSOR is less suited to GPUs, we focus on Jacobi iterations for preconditioning in this paper. Five sweeps of the Jacobi method with a relaxation parameter (under-relaxation) are applied on the local sub-domains, followed by a sweep of point Jacobi smoothing on the global domain. This preconditioner is referred to as the *Jacobi preconditioner* in this paper. Additional details of the preconditioners can be found in Wang et al. [26]. The preconditioners are applied from the left of the coefficient matrix, except for BiCGStab, where right preconditioning is used.

2.2. Linear solvers

In GenIDLEST, linear solvers are needed for the solution of decoupled momentum equations and the pressure equation. We need to solve three linear systems in 2D or four in 3D for every time step. CG is used for symmetric systems, and BiCGStab or GMRES(m) are used for non-symmetric systems. In this paper, we introduce and test in GenIDLEST the recycling Krylov subspace solvers rGCROT [3,7] and rBiCGStab [8], with a new hybrid approach for the latter. In the applications considered here, the momentum equations converge rapidly (≤ 2 iterations) and are always solved with BiCGStab. However, the pressure Poisson equation is more difficult to solve, and we consider various linear solvers described in this section.

Since the quantity of interest is the pressure correction from (1), at the start of each time step an initial zero pressure field, $x_0 = 0$, is used for all the solvers. Thus, the initial residual is $r_0 = b - Ax_0 = b$ for every time step.

GMRES/GMRES(m)

Algorithm 1 *(GMRES(m)).*

```
1.  Choose $x_0$, compute $r_0 = b - Ax_0$, $\|r_0\|_2$, and set $i = 0$.
2.  Choose tol, $m$, and max_itn.
3.  while $\|r_i\|_2 > \mathtt{tol} * \|b\|_2$ and $i \leq$ max_itn
4.      $v_1 = r_i / \|r_i\|_2$
5.      for $j = 1 \dots m$
6.          $v_{j+1} = Av_j$;   $i = i + 1$
7.          for $k = 1 \dots j$ do
8.              $h_{k,j} = v_k^T v_{j+1}$;   $v_{j+1} = v_{j+1} - v_k h_{k,j}$
9.          end
10.         $h_{j+1,j} = \|v_{j+1}\|_2$;   $v_{j+1} = h_{j+1}^{-1} v_{j+1}$
11.     end                                              {Arnoldi: $AV_m = V_{m+1}\underline{H}_m$ }
12      Solve $y = \arg\min_{\tilde{y}\in\mathbb{R}^m} \left\| e_1 \|r_0\|_2 - \underline{H}_m \tilde{y} \right\|_2$          { where $\underline{H}_m = (h_{k,j})$ }
13.     $x_i = x_{i-m} + V_m y$;                         {where $V_m = [v_1\, v_2\, \dots\, v_m]$, analogous for $V_{m+1}$ }
14.     $r_i = b - Ax_i$;                                {alternatively $r_i = r_{i-m} + V_{m+1}(\underline{H}_m y)$ }
15. end
```

Note: For ease of exposition, a few simplifications are made in this presentation. Our actual implementation handles all these cases properly. First, the algorithm given above does not check convergence in the inner cycle of m iterations. An estimate of the residual is available at each iteration, and the algorithm can be stopped at an arbitrary iteration. Second, the algorithm can stop early if at any point $\|v_{j+1}\|_2 = 0$; this is not checked in the algorithm above.

The Generalized Minimum Residual Method (GMRES) [30], given in Algorithm 1, is an iterative method for linear systems. At iteration m, the method minimizes the residual norm, $\|b - Ax_m\|_2$, over all vectors x in the Krylov subspace, $K_m(A, b) = \text{span}\{b, Ab, \dots, A^{m-1}b\}$, where A and b can be considered as the preconditioned matrix and right hand side, respectively. The Arnoldi recurrence, lines 4–11, generates a sequence of orthogonal vectors that span the Krylov space [31]. These vectors satisfy the (standard) Arnoldi recurrence relation

$$AV_m = V_{m+1}\underline{H}_m. \tag{2}$$

The solution to the least squares system, line 12, provides an approximate solution by orthogonal projection. For a non-symmetric system, each new Arnoldi vector must be explicitly orthogonalized against all previous ones. Therefore storage increases linearly with iteration count, until convergence or a restart (line 15), and work increases quadratically with iteration count. To limit high costs in storage and computation, *restarted* versions of the method, referred to as GMRES(m) [30], are used. However, restarting often results in a significant increase in the total number of iterations. The storage cost associated with GMRES(m) in our current implementation is $(m + 1)N + 4N$, in addition to the system matrix (here $19N$), where N is the number of unknowns and m is the restart frequency.

BiCGStab

Algorithm 2 *(BiCGStab).*

```
1.  Choose $x_0$, compute $r_0 = b - Ax_0$, $\|r_0\|_2$, and set $i = 0$. Choose $\tilde{r}$.
2.  Choose tol and max_itn
3.  while $\|r_i\|_2 > \mathtt{tol} * \|b\|_2$ and $i \leq$ max_itn
4.      $\rho = \tilde{r}^T r_i$
5.      if $\rho == 0$ then breakdown occurred; exit gracefully
6.      if $i == 0$ then $p = r_i$ else $\beta = (\rho/\rho_{\text{old}})(\alpha/\omega)$;  $p = r_i + \beta(p - \omega v)$ end
7.      $v = Ap$;
8.      $\alpha = \rho/(\tilde{r}^T v)$; $s = r_i - \alpha v$;
9.      if $\|s\|_2 \leq \mathtt{tol} * \|b\|_2$ then
10.         $x_{i+1} = x_i + \alpha p$;  $r_{i+1} = s$;  exit triumphantly (converged)
11.     end
12.     $t = As$;
13.     $\omega = (t^T s)/(t^T t)$
14.     $x_{i+1} = x_i + \alpha p + \omega s$;
```

```
15.     $r_{i+1} = s - \omega t$;
16.     $\rho_{\text{old}} = \rho$; $i = i + 1$;
17.  end
```

Note: Typically, $\tilde{r}$ is either chosen as a random vector or equal to r_0.

Unlike GMRES, BiCGStab does not need to store the entire Krylov subspace, nor does it perform a full orthogonalization (Gram–Schmidt process). Instead, BiCGStab relies implicitly on a (non-optimal) oblique projection to define its iterations [32], which requires only a short recurrence. The BiCGStab method was developed by Henk A. van der Vorst [33] and quickly gained popularity due to its robustness and low computational cost. The BiCGStab algorithm is given in Algorithm 2. The short term recurrence in BiCGStab, in iteration i, builds the residual as the product of two polynomials,

$$r_i = Q_i(A)P_i(A)r_0. \tag{3}$$

As a result, BiCGStab requires two (preconditioned) matrix-vector products per iteration, with the possible exception of the last iteration if convergence is attained at line 9 (Algorithm 2). The storage cost of BiCGStab is $8N$, in addition to the system matrix (here $19N$).

2.3. Recycling Krylov subspaces

Algorithm 3 *(rGCROT(m, k)/GCRODR(m, k)).*

```
1.   Choose $\hat{x}$; compute $\hat{r} = b - A\hat{x}$, $\|b\|_2$, and set $i = 0$.
1a.  Choose/Given $U$, compute $\tilde{C} = AU$ and $\tilde{C} = CR$ (thin QR decomposition)
1b.  $\xi = C^T\hat{r}$; $r_0 = \hat{r} - C\xi$; $\rho_0 = \|r_0\|_2$; $x_0 = \hat{x} + U(R^{-1}\xi)$
2.   Choose tol, $m$, and max_itn.
3.   while $\rho_i >$ tol $* \|b\|_2$ and $i \le$ max_itn
4.       $v_1 = r_i/\rho_i$
5.       for $j = 1 \dots m$
6.           $v_{j+1} = Av_j$;  $i = i + 1$
6a.          for $\ell = 1 \dots k$ do
6b.              $b_{\ell,j} = c_\ell^T v_{j+1}$;  $v_{j+1} = v_{j+1} - c_\ell b_{\ell,j}$
6c.          end                                              { $v_{j+1} = v_{j+1} - CC^T v_{j+1}$ }
7.           for $\ell = 1 \dots j$ do
8.               $h_{\ell,j} = v_\ell^T v_{j+1}$;  $v_{j+1} = v_{j+1} - v_\ell h_{\ell,j}$
9.           end
10.          $h_{j+1,j} = \|v_{j+1}\|_2$;  $v_{j+1} = h_{j+1,j}^{-1} v_{j+1}$
11.      end                                  { augmented Arnoldi: $AV_m = CB + V_{m+1}\underline{H}_m$ }
12       Solve $y = \arg\min_{\tilde{y}\in\mathbb{R}^m} \|e_1\rho_{i-m} - \underline{H}_m\tilde{y}\|_2$
12a.     $z = -By$                                                { where $B = (b_{\ell,j})$ }
13.      $x_i = x_{i-m} + V_m y$
13a.     $x_i = x_i - U(R^{-1}z)$
14.      $r_i = b - Ax_i$; $\rho_i = \|r_i\|_2$      { alternatively $r_i = r_{i-m} + V_{m+1}(\underline{H}_m y)$ }
14a.     Update $U$ and $C$ if desired                          { for details see [7,3] }
15.  end
```

Note: The changes from the GMRES(m) algorithm are given in lines 1a–1b, 6a–6c, 12a, 13a, and 14a.

When solving a sequence of slowly changing linear systems, $A_j x_j = b_j$, the Krylov subspaces $K_m(A_j, b_j)$ typically contain smaller subspaces that are all close to each other and can be approximated by some *recurring* subspace (which need not be a Krylov space). The idea of Krylov subspace recycling is (1) to compute this recurring subspace efficiently while solving subsequent linear systems and (2) to improve the rate of convergence of the iterative solves by iterating orthogonally to this space.

To get an idea of the effectiveness of this approach, consider restarted GMRES for a single system. If GMRES(m) restarts while little progress has been made, the method restarts with almost the same residual as before. Hence GMRES(m) explores a search space very close to the search space from the previous m steps. Since it has already computed the optimal solution from that space, this leads to very slow convergence or even stagnation. By keeping a judiciously selected subspace after restarting (or for the next linear system), recycling methods compute a new search space orthogonal to this recycled space, which prevents searching near a previous space and often leads to much faster convergence [3,4,7].

Next, we discuss how to implement recycling for the GMRES-like methods rGCROT and GCRODR and for the short recurrence method BiCGStab.

Recycling GMRES: rGCROT and GCRODR

The rGCROT(m, k) (or GCRODR(m, k)) algorithm is given in Algorithm 3. Let $U \in \mathbb{R}^{N\times k}$ and range(U) be a subspace we want to recycle. If U has been updated or the matrix has changed, we compute the (thin) QR-decomposition

$$CR = AU,$$

where $C \in \mathbb{R}^{N\times k}$ has orthonormal columns and $R \in \mathbb{R}^{k\times k}$ is upper triangular; see line 1a. In line 1b, we orthogonalize the initial residual $\hat{r}$ against C, and update the initial (solution) guess accordingly. This ensures

$$r_0, v_1 \perp C.$$

After this initialization, recycling is implemented by an augmented Arnoldi recurrence, lines 4–11 in Algorithm 3, the solution of a modified least squares problem, lines 12–12a, and a modified update for the approximate solution, lines 13–13a. Otherwise, the method proceeds like GMRES(m); compare corresponding lines in Algorithms 1 and 3. Lines 6a–6c extend the standard Arnoldi recurrence given in lines 4–11 of Algorithm 1, orthogonalizing the new v_{j+1} against C before orthogonalizing against $v_1, \ldots, v_j$. This gives

$$V_{m+1} \perp C \tag{4}$$

and the augmented Arnoldi recurrence relation, see (2) for comparison,

$$AV_m = CB + V_{m+1}\underline{H}_m, \qquad \text{where} \quad B = C^T AV_m. \tag{5}$$

Hence, we approximate the residual in a space, range($V_{m+1}\underline{H}_m$), that is indeed orthogonal to the (previously used) space range(C). The optimal update to the solution from the space range(U) + range(V_m) leads to a modified least squares problem. We have

$$x_m = x_0 + V_m y + UR^{-1}z \quad \Rightarrow \quad r_m = r_0 - AV_m y - Cz, \tag{6}$$

then substituting the augmented Arnoldi recurrence (5) gives

$$(y, z) = \arg \min_{\tilde{y}\in\mathbb{R}^m, \tilde{z}\in\mathbb{R}^k} \left\| V_{m+1}(e_1\|r_0\|_2 - \underline{H}_m\tilde{y}) - C(B\tilde{y} + \tilde{z}) \right\|_2 . \tag{7}$$

Using (4), we can minimize separately for the V_{m+1} component (cf. Algorithm 1, line 12) and the C component. The latter is minimized by taking $z = -By$ (line 12a).

If the recurring subspace approximately persists from one system to the next, recycling is very effective. The overhead consists mostly of the additional orthogonalizations in lines 6a–6c. Compared with GMRES(m) recycling variants may even reduce memory requirements, because m can be much smaller while the number of columns in U and C is often modest; see [7]. Nevertheless, the memory requirements are typically substantially higher than for short recurrence methods like BiCGStab.

GCROT and GCRODR differ only in how they select the space range(U). The GCRO-DR algorithm uses approximate invariant subspaces for U, whereas GCROT measures angles between successive search spaces to approximate the recurring subspace directly [3]. We refer to GCROT with recycling for a sequence of systems as rGCROT here. For our applications of interest, rGCROT gives faster convergence and is a more robust solver for the initial time steps. The additional storage cost of the rGCROT solver over GMRES(m) is $2kN$, where k is the (maximum) dimension of the recycle space.

rBiCGStab

Algorithm 4 *(rBiCGStab with one recycle space (used for the hybrid algorithm)).*

```
1.    Choose $\hat{x}$, compute $\hat{r} = b - A\hat{x}$, $\|b\|_2$, and set $i = 0$.
1a.   Choose/Given $U$ and possibly $C$ and $R$ (provided by rGCROT for the hybrid algorithm).
1b.   **if** $C$ and $R$ not given **then** compute $\tilde{C} = AU$ and $\tilde{C} = CR$ (thin QR decomposition) **end**
1c.   Compute $\eta_1 = C^T\hat{r}$; $r_0 = \hat{r} - C\eta_1$; and $\xi = -\eta_1$   {*accumulate U-updates for x in* $\xi$ }
2.    Choose tol, max_itn, and $\tilde{r}$.
3.    **while** $\|r_i\|_2 >$ tol $* \|b\|_2$ **and** $i \le$ max_itn
4.        $\rho = \tilde{r}^T r_i$
5.        **if** $\rho == 0$ **then** breakdown occurred; **exit** gracefully
6.        **if** $i == 0$ **then** $p = r_i$ **else** $\beta = (\rho/\rho_{\text{old}})(\alpha/\omega)$;  $p = r_i + \beta(p - \omega v)$ **end**
7.        $v = Ap$;
7a.       $\eta_1 = C^T v$;  $v = v - C\eta_1$
8.        $\alpha = \rho/(\tilde{r}^T v)$;  $s = r_i - \alpha v$
9.        **if** $\|s\|_2 \le$ tol $* \|b\|_2$ **then**
10.           $x_{i+1} = x_i + \alpha p$;  $r_{i+1} = s$
10a.          $\xi = \xi + \alpha\eta_1$;  **exit** triumphantly (converged)
```

```
11.        **end**
12.        $t = As$
12a.       $\eta_2 = C^T t;\ t = t - C\eta_2$
13.        $\omega = (t^T s)/(t^T t)$
13a.       $\xi = \xi + \alpha\eta_1 + \omega\eta_2$   {accumulate $U_k$ updates for $x$ }
14.        $x_{i+1} = x_i + \alpha p + \omega s$
15.        $r_{i+1} = s - \omega t$
16.        $\rho_{\text{old}} = \rho;\ i = i + 1$
17.    **end**
17a.   $x_i = x_i - U(R^{-1}\xi)$   {add accumulated updates to solution }
```

Note: As the solution is needed only at the very end, all updates with the U vectors (updates of the type $x = x - UR^{-1}z$) are postponed, reducing #its $* 4Nk$ work to $2Nk$ work. The length k vector ξ accumulates these postponed updates. The changes from the standard BiCGStab algorithm are given in lines 1a–1c, 7a, 10a, 12a, 13a, and 17a.

As we saw above, BiCGStab builds polynomials of the type (3). In general, for recycling in BiCGStab we would recycle two subspaces, each augmenting one of the two Krylov subspaces in the underlying bi-Lanczos recurrence [8]. However, as shown in [8, Section 5, Example 1], this is not always necessary. If only one recycle space is used, range(U), the algorithm simplifies substantially. To see how such a recycling BiCGStab can be implemented, we rearrange the augmented Arnoldi recurrence relation (5) for A as a standard Arnoldi recurrence relation for $A_C = (I - CC^T)A$. Moving the term CB to the left and using $B = C^T AV_m$, we get

$$AV_m = CB + V_{m+1}\underline{H}_m \quad \Leftrightarrow \quad (I - CC^T)AV_m = V_{m+1}\underline{H}_m. \tag{8}$$

Hence, as for rGCROT, we can generate the Krylov subspace $K_m(A_C, r_0)$ with the operator A_C and with r_0 orthogonalized against C; see lines 1a, 7a, and 12a in Algorithm 4. This leads to polynomials of the type

$$r_i^{\text{rBiCGStab}} = Q_i(A_C)P_i(A_C)r_0, \tag{9}$$

with starting residual $r_0 = (I - CC^T)\hat{r}$. So, we run BiCGStab with a modified initial residual, A_C replacing A, and special updates for the solution as for rGCROT. The special updates for the solution can be derived from the fundamental relation between solution updates and residual updates in Krylov methods,

$$x_{i+1} = x_i + z \quad \Rightarrow \quad r_{i+1} = r_i - Az, \tag{10}$$

which implies

$$z = A^{-1}(r_i - r_{i+1}). \tag{11}$$

Note that (11) still holds when the residual is computed as in (9). Hence, the residual update $s = r_i - \alpha A_C p$ (lines 7–8) corresponds to the solution update

$$x_{i+1} = x_i + A^{-1}(\alpha A_c p) = x_i + \alpha p - U(\alpha R^{-1}\eta_1) \quad \text{(see lines 10–10a)}, \tag{12}$$

where the U-component of the update, $U(\alpha R^{-1}\eta_1)$, is postponed till the end; see line 17a. The residual update in line 15 leads to an analogous update for the approximate solution.

This implementation also avoids some expensive updates if possible. The algorithm for the general case of rBiCGStab and its derivation can be found in [8,34]. The additional storage for rBiCGStab compared with BiCGStab is $2kN$, where again k is the (maximum) dimension of the recycle space.

2.4. Hybrid approach

Although the rGCROT method is more robust and often converges in fewer iterations than BiCGStab, its iterations are relatively expensive. As a result, for later time steps, the run time for rGCROT might be higher than for (r)BiCGSTab. In our application, the additional robustness and fast convergence of rGCROT are important mainly in the first few time steps. For later time steps, we can use BiCGStab or, for faster convergence and improved robustness, rBiCGStab. Therefore, we propose a hybrid strategy that starts with rGCROT and switches to rBiCGStab, recycling the subspace computed by rGCROT for previous systems. The recycle spaces constructed in rGCROT, range(U) and range(C), remain constant during the rBiCGStab calculations. Thus a large recycle space can be used efficiently in this approach. An alternative approach of intermittent solves with rGCROT to update the recycle space can be used when the system matrix changes for certain time steps. Moreover, if convergence is poor at an intermediate phase, we can switch back to rGCROT. This was not necessary in the applications discussed in this paper. We list the advantages and disadvantages of the various solvers in Table 1. The hybrid approach is designed to combine the advantages of the robust rGCROT algorithm and the cheaper rBiCGStab iterations.

Table 1
Advantages and disadvantages of various linear solvers considered.

Solver	Advantages	Disadvantages
BiCGStab	· Cheap iterations (in time) with low storage requirements · Often fast in run time	· Irregular and/or slow convergence for stiff problems, possible accuracy problems
rBiCGStab	· Relatively cheap iterations (in time)	· Improved but irregular convergence · Higher storage requirement
GMRES(m)	· Min. residual over *m* iterations · Monotonic residual norm decrease	· Large storage requirement · Expensive iterations due to orthogonalization
rGCROT	· Fast convergence (in iterations) because of recycling · Monotonic residual decrease	· Higher storage requirement · Expensive iterations due to additional orthogonalizations

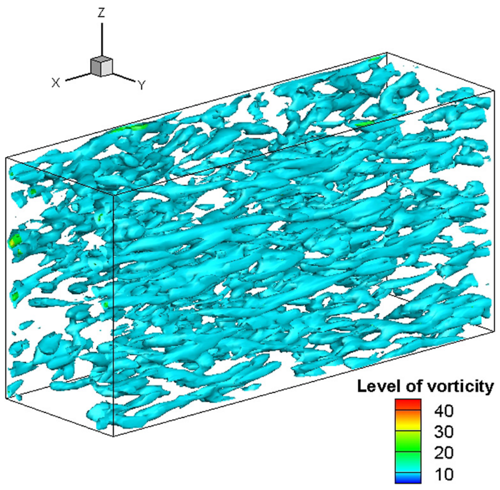

Fig. 1. Turbulent channel flow showing vorticity.

3. CFD problem description

In this study, we consider two problems where the pressure coefficient matrix remains constant for all subsequent time steps,

1. Turbulent channel flow,
2. Flow through porous media.

3.1. *Turbulent channel flow*

Turbulent channel flow has been used extensively as a canonical problem that embodies most of the physical complexities of wall bounded turbulent flow. For this reason, turbulent channel flow, depicted in Fig. 1, is a good problem to evaluate the linear solvers. In this study, we use a three-dimensional, orthogonal computational grid, leading to a seven point stencil, with a single mesh block of $64 \times 64 \times 64$ computational cells. A higher grid density near the wall is used to resolve the boundary layer, and periodic boundary conditions are used in the flow direction to mimic an infinitely long turbulent channel. We specify a fixed mean pressure gradient in the flow direction to balance the wall friction. Prescribed perturbations in the flow, based on the mean turbulent channel flow profile, are given as the initial conditions to trigger the onset of turbulence. The flow is allowed to evolve in time until the solution reaches a stationary state. A skin friction Reynolds number (Re_τ) of 180, based on channel half width and wall friction velocity, is used to compare with the study of Moser et al. [35]. First, the simulation is run using the rGCROT algorithm until the flow turbulence has become statistically stationary in time (stationary flow). For performance comparison, 30 time steps of 5×10^{-5} second are run at the start and end (after 2.5 seconds) of the calculations with various solvers. An absolute tolerance of 10^{-6} is used as the convergence criterion on the L^2 norm of the residual. The Jacobi preconditioner with 5 inner iterations is used for all the solvers.

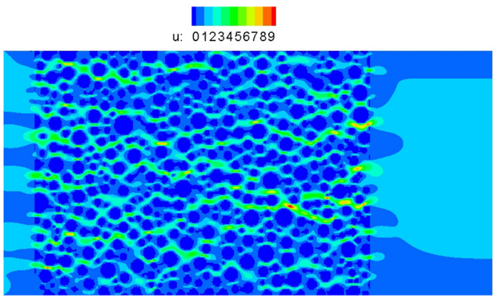

Fig. 2. Flow through porous media with u velocity contours.

3.2. Flow through porous media

Porous media flow has a wide variety of applications. In the current study, we model the porous medium using IBM to resolve the porous structure, which is constructed using the stochastic reconstruction procedure, as shown in Fig. 2. The IBM scheme uses indirect forcing of sharp interfaces on the background mesh [21]. We use a 2D background mesh consisting of 2.56 million orthogonal cells and 16 mesh blocks ($100 \times 800 \times 2 \times 16$) to simulate a bulk flow Reynolds number (Re_b) of 10^{-4}. In the current IBM framework, two computational cells are needed in the Z direction for 2D calculations. The porous medium has wall boundaries on the top and the bottom, with inlet and outlet boundaries on the left and right, respectively. The pressure coefficient matrix remains constant for every time step, as the IBM related flux corrections at the immersed boundaries are not applied. For the performance comparison, ten time steps of 10^{-8} second are run at the start of the calculations with various solvers. A relative tolerance of 10^{-10} on the L^2 norm of the residual is used as the convergence criterion, as the initial residual norm is large and no solver would converge to an absolute tolerance of 10^{-6}, since this is too close to the relative machine precision. The preconditioner uses 5 inner iterations of either Jacobi iterative smoothing or SSOR smoothing. Although the SSOR algorithm is more appropriate for symmetric matrices, it can be applied to nonsymmetric coefficient matrices as a preconditioner, as this only requires an approximate solution.

4. Results

The turbulent channel flow simulation is performed serially on a local system (Dual Intel®Xeon®CPU X5650 @ 2.67 GHz & 48 GB memory), whereas the flow through porous media study is performed using 16 CPU cores (Dual Intel®Xeon®CPU E5-2670 @ 2.60 GHz & 64 GB memory) with MPI parallelism on the BlueRidge HPC system at Virginia Tech.

In the experiments discussed in this section, we check the convergence for BiCGStab and rBiCGStab only at the end of one complete execution of the while-loop, that is, going once through lines 3–17, in Algorithms 2 and 4. Therefore, one iteration of (r)BiCGStab includes two (preconditioned) matrix-vector products. Similarly, for GMRES(m) and rGCROT(m, k) we check convergence only at the end of one complete execution of the while-loop, that is, going once through lines 3–15, in Algorithms 1 and 3. In the context of restarted GMRES(m) and rGCROT(m, k), this is often called a cycle. A cycle involves m steps of the (augmented) Arnoldi recurrence, and hence involves m matrix-vector products. Notice that the number of matrix-vector products determines the dimension of the Krylov space from which a solution is computed. Therefore, in comparing how fast methods converge, we compare the number of matrix-vector products until a specified tolerance is reached. Since the number of flops (and hence run time) per GMRES(m) or rGCROT(m, k) iteration, typically defined as one iteration of the (augmented) Arnoldi recurrence, is not constant, we provide some of the timing/performance information below per cycle (m matrix-vector products). Similarly, we provide some of the timing/performance information for (r)BiCGStab per iteration (2 matrix-vector products).

In our comparisons of the various solvers, we have to make a choice of either running each solver independently in GenIDLEST for the specific problem, or choosing one solver as the master and make the solutions computed by that solver determine the linear systems solved by all the other solvers. The first approach has the advantage that the performance numbers reported are the actual results obtained for running GenIDLEST with that solver. It has the disadvantage that the solvers being compared all solve slightly different problems, potentially impacting the comparison. In our case, only the right hand sides are slightly different. The second approach avoids this problem, but will compute performance characteristics that would not actually be observed using these solvers independently for GenIDLEST.

Since we compare the solvers over a fairly large number of time steps (linear systems), we prefer the first approach. The effect of a few potentially unfortunate right hand sides for, say, one solver will mostly be averaged out. Moreover, we are careful to ensure that the right hand sides for each time step don't vary much among the solvers. The starting point for all the solvers is the same, and we use a relatively high convergence criterion to ensure that the solutions of linear systems (which determine subsequent right hand sides), are very close. So, although the right hand sides are not exactly the same across all solvers, we don't expect this to have a large impact on the average and total matrix-vector product counts and on the run time required for convergence. As will be clear from the reported results, for most solvers the number of

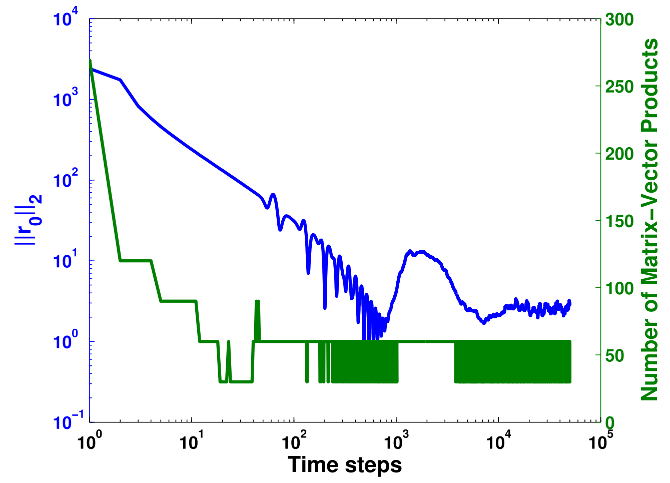

Fig. 3. Number of matrix-vector products for rGCROT(30, 130) to converge for each time step (right y-axis) and the initial $\|r_0\|_2$ for each time step (left y-axis).

matrix-vector products for convergence does not fluctuate much from time step to time step; see Figs. 4 and 5. This suggests the convergence rates are not particularly sensitive to the variations in typical right hand sides.

4.1. Turbulent channel flow

There are 6 parameters associated with the rGCROT algorithm [3]: the maximum number of steps of the augmented Arnoldi recurrence (lines 5–11 in Algorithm 3), m[2]; the (max) dimension of the recycle space, or outer search space, k; the number of outer vectors after truncation of the outer space, here $k - 10$; the number of inner vectors to select outer vectors from, here $m/2$; the number of inner vectors selected to extend the outer space, here 1; and, the number of latest inner vectors kept to extend the outer space, here 0.

In this study, we focus on the two most influential parameters, m and k. The other parameter values are given as constants or in terms of m and k (as stated above). To analyze optimal performance, k is varied from 20 to 170, and m is varied from 20 to 100, both with increments of 10, at the start of the simulations and also at the end of simulations. At the start of the simulations, rGCROT(30, 130) is optimal with respect to solution time, whereas rGCROT(20, 130) is optimal after the flow becomes stationary. Additional details regarding the rGCROT parameters and selection guidelines can be found in [3].

Fig. 3 gives the number of matrix-vector products needed to converge for a long sequence of time steps with the rGCROT(30, 130) solver. It also shows the L^2 norm of the initial residual ($\|r_0\|_2 = \|b\|_2$) for every time step. The variation in $\|r_0\|_2$ is directly associated with the flow physics. At the first time step, the initial guess for the velocity with prescribed perturbations in the flow is farthest from the solution, so the $\|r_0\|_2$ value is maximum. As the flow starts to develop, $\|r_0\|_2$ goes down and drops by 3 orders of magnitude. After about 1000 time steps, $\|r_0\|_2$ goes back up by an order of magnitude, because rapid changes in the flow rate cause relatively large changes in subsequent right hand sides (b). Once the flow becomes stationary, the flow rate and consequently $\|r_0\|_2$ settle down to quasi-steady values. The number of matrix-vector products needed to converge for a given time step is influenced by the variations in $\|r_0\|_2$, since an absolute convergence tolerance of 10^{-6} is used. For the rGCROT algorithm, the dimension and quality of the recycle space also influences the number of matrix-vector products needed to converge. Initially, the dimension of the recycle space is small and the recurring subspace has not been constructed yet, and thus the number of matrix-vector products to converge is higher. As a recurring subspace is discovered and captured by the recycle space, the number of matrix-vector products drops to 30 or one cycle. When the flow develops, the right hand side changes considerably with each time step due to rapid changes in flow rate. This warrants intermittent updates to the recycle space, and the number of cycles (each 30 matrix-vector products) to converge fluctuates between one and two. As the $\|r_0\|_2$ values goes back up, the number of cycles increases to two, since the relative reduction required in the residual norm also increases. After this phase of fluctuations, the flow settles and the number of cycles is again one or two.

Tables 2 and 3 give the time to solution and the average number of matrix-vector products to converge. From the results it is clear that the parameter-tuned rGCROT solver takes the least number of matrix-vector products per time step and has the lowest time to solution. Notice that the time for a BiCGStab iteration, which takes 2 matrix-vector products and a few orthogonalizations, is very small compared with a rGCROT(30, 130) cycle, which takes 30 matvecs and many

[2] The parameter m also gives the maximum dimension of the inner search space, span$(v_1, \ldots, v_m)$, and it is similar to the restart frequency of GMRES.

Table 2
Solver performance at the start of the calculations.

First 30 time steps	BiCGStab	GMRES(50)	rGCROT(30, 130)
Total time to solution (s)	365	12 680	323
Average number of matrix-vector products per time step	184	4350	69
Average number of iterations (its) or cycles (c) per time step	92 (its)	87 (c)	2.3 (c)
Solver time per iteration (s/it) or per cycle (s/c)	0.132 (s/it)	4.84 (s/c)	4.68 (s/c)

Table 3
Solver performance when the flow is stationary.

30 time steps at end	BiCGStab	GMRES(50)	rGCROT(20, 130)
Total time to solution (s)	242.7	336.7	164.8
Average number of matrix-vector products per time step	112	100	30
Average number of iterations (its) or cycles (c) per time step	56	2	1.5
Solver time per iteration (s/it) or per cycle (s/c)	0.144	5.52	3.66

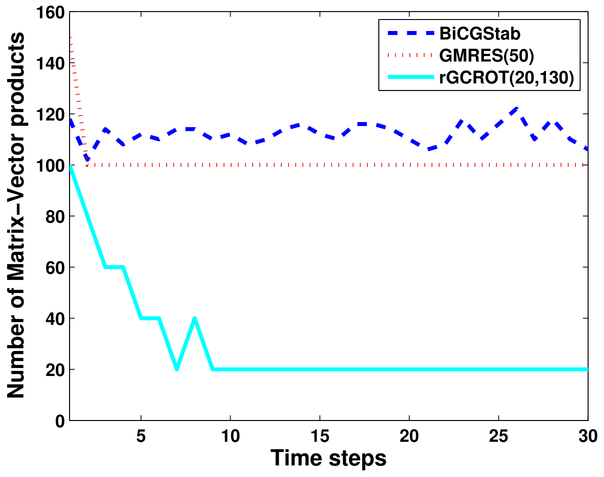

Fig. 4. Number of matrix-vector products for each solver to converge to an absolute tolerance of 10^{-6} for each time step when the flow is stationary.

Table 4
Solver storage cost comparison for the turbulent channel case.

Solver	Additional storage cost
BiCGStab	$8N$
GMRES(50)	$55N$
rGCROT(30, 130)	$293N$

orthogonalizations. However, it takes many more BiCGStab iterations to converge than rGCROT(30, 130) cycles. At the start of the calculations, GMRES(m) did not converge in 1000 matrix-vector products if the restart frequency, m, was chosen less than 50. This prolonged stagnation in convergence is a typical issue with GMRES(m) for small m. All the solvers converge faster when the flow becomes stationary, because $\|r_0\|_2$ is smaller.

The number of matrix-vector products to converge, after the flow becomes stationary, is shown in Fig. 4 for a sequence of 30 time steps. The number of matrix-vector products for rGCROT(20, 130) remains constant once the recycle space has captured the dominant subspace. The method convergences in one cycle (20 matvecs). The number of matrix-vector products to converge for GMRES(m) also remains constant (100 matvecs or 2 cycles), but the time to solution is much larger as we have to build the search space from scratch after every restart. The number of matrix-vector products for BiCGStab fluctuates only modestly.

The storage costs of the various solvers for the turbulent channel flow problem are listed in Table 4. We only list storage costs beyond the standard costs in every solver, for the system matrix, the right hand side, and the approximate solution, and that are at least $O(N)$, where N is number of unknowns. Required auxiliary and temporary vectors used in the solvers are counted towards the storage overhead. As mentioned in Section 2.2, the BiCGStab solver has the lowest storage requirements, and the rGCROT solver often has substantially higher storage requirements. For the same amount of storage, it has been shown that the rGCROT algorithm tends to outperform GMRES(m) [3]. When storage is cheap, the rGCROT solver typically shows superior performance compared with other solvers.

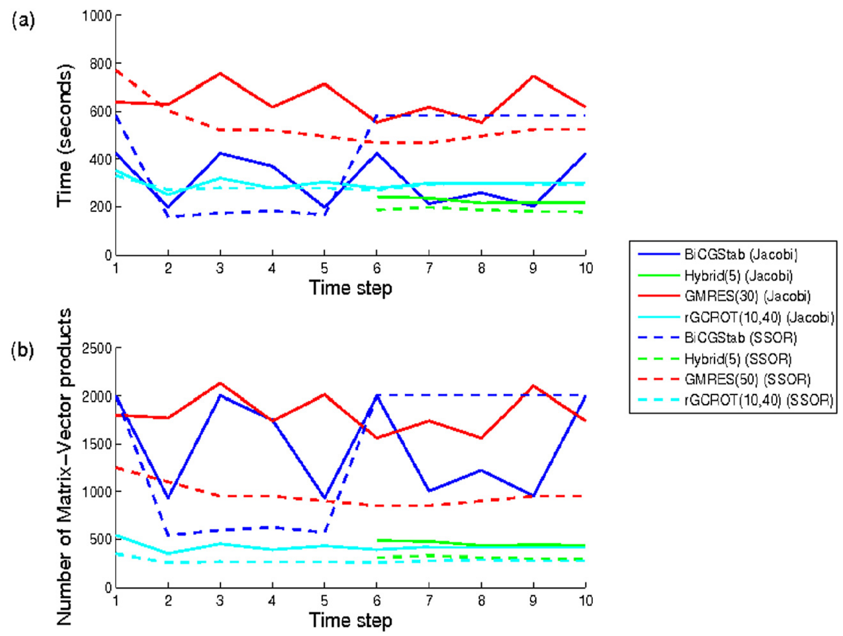

Fig. 5. Comparison of solvers using the Jacobi preconditioner (solid lines) and using the SSOR preconditioner (dashed lines): (a) Time to converge (seconds), (b) Number of matrix–vector products to convergence to a relative tolerance of 10^{-10}.

In summary, rGCROT is a robust solver that is capable of converging in very small numbers of cycles (or matrix-vector products) by exploiting subspace recycling. However, an rGCROT cycle may be relatively expensive in time and in storage. On the other hand, BiCGStab has cheap iterations but may take many more iterations (or matrix-vector products) to converge. This suggests a hybrid approach that combines the robustness of rGCROT with the cheap iterations of BiCGStab by using the rGCROT recycle space in rBiCGStab as discussed below. We also tested this hybrid approach for the turbulent channel flow problem, but the method was not faster (in time) than rGCROT. However, for the porous media application in the next section the hybrid method does produce shorter runtimes.

4.2. Flow through porous media

Next, we analyze several solvers for the simulation of porous media flow. We determine optimal parameters for rGCROT(m, k) and GMRES(m) and compare the solvers and preconditioners for those parameters. Finally, the tuning of the hybrid approach is discussed.

Parameter study for rGCROT(m, k) and GMRES(m)

For a fair comparison, the parameters of rGCROT and GMRES(m) are optimized independently for each solver. For rGCROT, we consider the performance with the Jacobi preconditioner while varying the restart frequency (m) and the dimension of the recycle space (k). The parameter k is varied from 10 to 210, with increments of 10, and m is varied from 10 to 80, with increments of 5. For the other parameters, we take the following values: $k - 10$ for the number of outer vectors after truncation of the outer space; $m/2$ for the number of inner vectors to select outer vectors from; 2 for the number of inner vectors selected to extend the outer space; and, 0 for the number of latest inner vectors kept. We obtained the minimum runtime for rGCROT(10, 40), and this choice is used throughout this section.

The restart frequency for GMRES(m) with the Jacobi preconditioner is varied from 15 to 55, with increments of 5, and with the SSOR preconditioner from 30 to 60, with increments of 10. GMRES(30) with the Jacobi preconditioner and GMRES(50) with the SSOR preconditioner gave the optimal performance in terms of time to solution for the first 10 time steps, and this choice is used for further comparison.

Comparison of solvers and preconditioners

We compare the average number of matrix-vector products and total time taken for the first 10 time steps in Table 5. Note that the average numbers of matrix-vector products for BiCGStab are based on a maximum of 1000 iterations (2000 matrix-vector products) for a time step, and for several time steps this method did not reach the convergence tolerance (see Fig. 5). We continued the simulation for the purpose of comparison. The average number of matrix-vector products for the hybrid approach is taken only over the time steps that were solved using rBiCGStab. For the hybrid approach, hybrid(n) indicates n time steps with rGCROT(10, 40) and $(10 - n)$ time steps with rBiCGStab.

Table 5
Average number of Matrix-Vector products per time step and total time for 10 time steps of the porous media flow problem for various solvers and preconditioners. The dashes below for GMRES(m) indicate that the convergence for the other m-value was faster (we aim to compare only for the optimal parameters).

Solver	Average number of matrix-vector products		Total time (s)	
	Jacobi	SSOR	Jacobi	SSOR
BiCGStab	1480 (max 2000)	1432 (max 2000)	3185	4202
Hybrid(5)	454	310	2682	2076
GMRES(30)	1800	–	6497	–
GMRES(50)	–	950	–	5539
rGCROT(10, 40)	420	280	3028	2943

Table 6
Convergence comparison for the tenth time step of the porous media flow problem. The dashes below for GMRES(m) indicate that the convergence for the other m-value was faster (we aim to compare only for the optimal parameters).

Solver	Number of matrix-vector products		Solution time (s)	
	Jacobi	SSOR	Jacobi	SSOR
BiCGStab	(2000)	(2000)	423.1	582.1
rBiCGStab	436	298	215.9	178
GMRES(30)	1740	–	617.5	–
GMRES(50)	–	950	–	523.2
rGCROT(10, 40)	420	280	298.5	293.4

The results in Table 5 show that the hybrid method has the lowest time to solution with both preconditioners and requires only a moderate average number of matrix-vector products for convergence. We also see that rGCROT(10, 40) itself is competitive, only losing to the hybrid solver in run time while having the lowest average number of matrix-vector products required for convergence. However, the difference in matrix-vector products is small, and as rBiCGStab is cheaper in terms of orthogonalizations it reduces the runtime substantially, especially with the SSOR preconditioner.

Using the SSOR preconditioner reduces the average number of matrix-vector products for all solvers, indicating that it is the more effective preconditioner for this problem. For all solvers except BiCGStab, the total time also decreases with the SSOR preconditioner.

Fig. 5 shows the number of matrix-vector products to converge and the time to solution per time step for the various solvers. This complements Table 5, which only gives the average results. The results for the hybrid solver are given only for the time steps solved with rBiCGStab, as the results for the initial steps with rGCROT(10, 40) are already given. The number of iterations remains almost constant for all the solvers except for BiCGStab. As can be seen, BiCGStab (with either preconditioner) did not converge to the required relative tolerance for several time steps. Clearly, BiCGStab is not robust enough for this problem.

Table 6 shows the number of matrix-vector products for convergence and the solution time for the tenth time step. We focus on this time step, because the hybrid approach is tuned based on this time step as discussed in the next subsection. The hybrid approach is represented by rBiCGStab, since the tenth time step is solved by rBiCGStab using the recycle space from 5 initial time steps of rGCROT(10, 40). The rBiCGStab solver with the SSOR preconditioner has by far the lowest time to solution.

Fig. 6 plots the convergence of the relative residual norm for each solver, $\|r_k\|_2/\|r_0\|_2$, against the number of matrix-vector products, for the tenth time step. The residual norms of both GMRES(m) and rGCROT decrease monotonically, whereas the residual norm of BiCGStab shows irregular convergence towards the end and the method does not converge. The residual norm of rBiCGStab also displays some (modest) irregular convergence towards the end. When comparing the preconditioners, for all solvers the initial residual ($\|r_1\|_2/\|r_0\|_2$) and initial slope are about the same, except for GMRES(m) since the restart frequencies differ.

The L^2 norm convergence curves in Fig. 6 are given for the *true* residuals ($b - Ax_i$), except for the rBiCGStab solver, where it is given for the *updated* residual ($r_{i+1} = s - \omega t$, line 15 in Algorithm 4). The update from the outer projection space to the solution vector in rBiCGStab is performed only after convergence (see line 17a Algorithm 4). So, the calculation of the true residual for every iteration would introduce additional overhead, and is therefore avoided.

The storage costs for the various solvers for the porous media flow problem are listed in Table 7. Similar to the discussion in Section 4.1, the storage cost for the BiCGStab solver is the lowest, and it is the highest for the rGCROT algorithm. The hybrid approach with rBiCGStab has slightly lower persistent storage requirements than rGCROT(10, 40) but higher than for the other solvers. However, it has the lowest time to solution. So, there is a trade-off between performance and storage, and the selection of the best solver may depend on the availability of storage and data access costs.

Hybrid approach tuning

We analyze and tune the performance of the hybrid approach to obtain the best parameter choices. This analysis provides the parameter choices for the experiments discussed in the previous subsection. We vary the dimension of the recycle

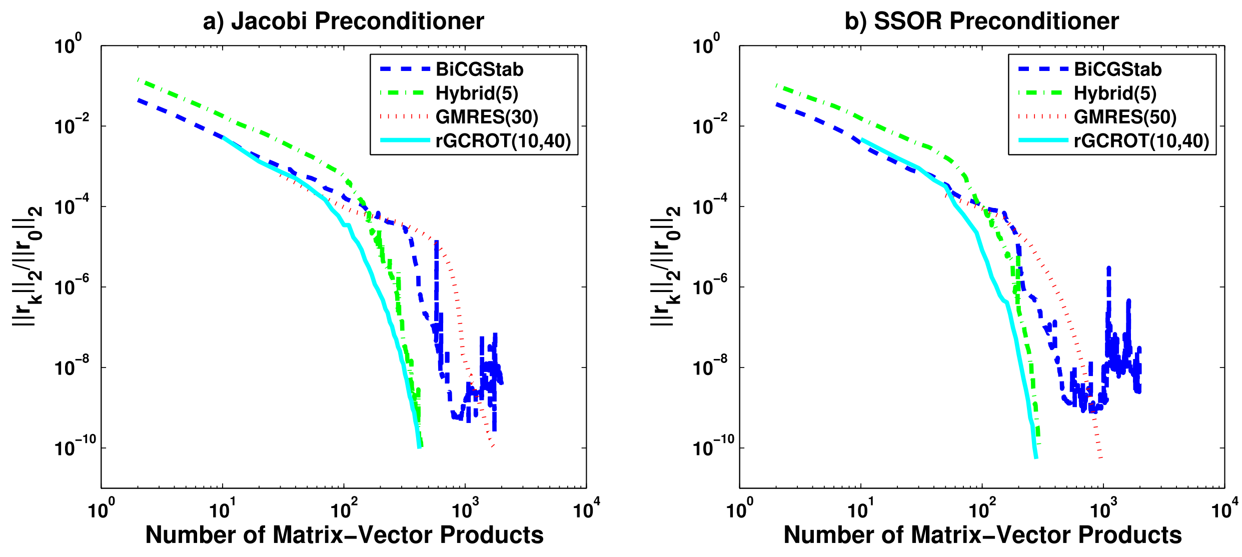

Fig. 6. Convergence comparison for the tenth time step using BiCGStab, Hybrid, GMRES and rGCROT solvers with (a) the Jacobi preconditioner and (b) the SSOR preconditioner.

Table 7
Solver storage cost comparison for flow through porous media case.

Solver	Additional storage cost
BiCGStab	8N
GMRES(50)	55N
rGCROT(10, 40)	93N
Hybrid(rBiCGStab)	88N

Table 8
Effect of the number of initial time steps with rGCROT(10, 40) on the solution time for the hybrid approach.

Number of initial time steps with rGCROT(10, 40)	Solution time for the tenth time step (s)	Total time (s)
1	Unstable	–
2	Unstable	–
3	239.5	3089
4	248.3	3003
5	215.9	2682
6	232.8	2792
7	233.9	2945
8	252.3	2928
9	234.4	2967

space, k, and the time step, n, at which we switch the solver from rGCROT(10, 40) to rBiCGStab. We vary k from 10 to 210, with a step of 10, while using six time steps to build the recycle space using rGCROT. The optimal run time for the hybrid approach is also obtained with rGCROT(10, 40). For a larger recycle space, the cost of projection becomes too high, whereas for a smaller recycle space, the number of matrix-vector products for convergence becomes too large.

Table 8 lists the time to solution for the tenth time step and the total time for several n in the hybrid(n) approach. We use the Jacobi preconditioner for the tuning. If we run 1 or 2 time steps with rGCROT(10, 40) and then switch to rBiCGStab, the rBiCGStab solver becomes unstable, generating ever larger residual norms. This suggests that the recycle space is not yet of good quality. However, with further initial time steps by rGCROT(10, 40), the recycle space becomes better, and subsequent systems are solved faster. With the minimum runtime as the criterion, the optimal switching point is after 5 time steps of rGCROT(10, 40).

The convergence histories for the tenth time step (with rBiCGStab) for several n are shown in Fig. 7. The convergence histories for all $n > 2$ are fairly close to each other, suggesting that once a good recurrent subspace is discovered, the need to update this space further is minimal. These results also show that the choice of time step to switch from rGCROT to rBiCGStab is not particularly sensitive, although care should be taken not to switch too early. For some problems, it might be useful to update the recycle space after a certain number of time steps, based on the changes to the system matrix and/or right hand side. This was not investigated in the current study.

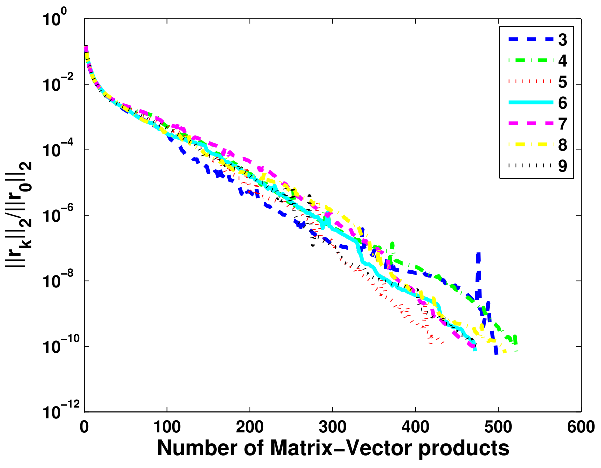

Fig. 7. Effect of the number of initial time steps with rGCROT(10, 40) on the convergence of the tenth time step with rBiCGStab.

The hybrid approach is appropriate for problems where the condition number of the preconditioned system matrix is high or the spectrum is otherwise unfavorable, requiring a large number of iterations to converge, and there is a sequence of such systems to be solved. In this case, typically, performance gains from reducing the number of iterations outweighs the overhead per iteration associated with orthogonalization against the recycle space. Finally, as our results show, the hybrid approach with rBiCGStab significantly reduces overall runtime for certain CFD applications at the cost of a moderate increase in storage costs.

5. Conclusions and future work

The Krylov subspace recycling linear solvers rGCROT and rBiCGStab are tested for two important CFD applications that lead to a sequence of linear systems, and they are compared with the standard solvers GMRES(m) and BiCGSTab. For a turbulent channel flow problem, the parameter tuned rGCROT solver performs better than both BiCGStab and GMRES(m) in terms of time to solution and number of matrix-vector products to converge. A novel approach to build the recycle space for recycling BiCGStab is proposed and tested for a porous media flow simulation. This hybrid approach has the fastest time to solution with a moderate number of matrix-vector products for convergence compared with BiCGSTAB, GMRES(m) and rGCROT.

Several other solver aspects provide further important directions of research. We should consider a similar study to analyze which preconditioners provide the best performance, both in terms of convergence and robustness, including convergence on a sequence of finer meshes, and in terms of high performance implementations. In particular, it will be interesting to analyze what type of multilevel preconditioners are most effective for the CFD problems considered in this paper and allow high performance on modern architectures, for example, multiple GPU cards. The optimal interaction between Krylov recycling and preconditioning would be an important aspect of such a study. Analogously, it would be important to consider how finer discretizations might affect the benefits of recycling. Of course, such an analysis cannot be separated from choices how to precondition for a sequence of finer meshes.

Acknowledgements

This work was funded in part by the Air Force Office of Scientific Research (AFOSR) Computational Mathematics Program via Grant Number FA9550-12-1-0442 and in part by the National Science Foundation (USA) under Grant Numbers DMS-1025327 and DMS-1217156. The authors also acknowledge Advanced Research Computing at Virginia Tech for providing computational resources.

Finally, we thank the anonymous reviewers for their careful and helpful suggestions, which greatly helped us to improve this paper.

References

[1] S. Wang, E. de Sturler, G.H. Paulino, Large-scale topology optimization using preconditioned Krylov subspace methods with recycling, Int. J. Numer. Methods Eng. 69 (2007) 2441–2461.
[2] R. Barrett, M. Berry, T.F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, H.A. van der Vorst, Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd edition, SIAM, Philadelphia, PA, 1994.
[3] M.L. Parks, E. de Sturler, G. Mackey, D.D. Johnson, S. Maiti, Recycling Krylov subspaces for sequences of linear systems, SIAM J. Sci. Comput. 28 (5) (2006) 1651–1674.
[4] E. de Sturler, Nested Krylov methods based on GCR, J. Comput. Appl. Math. 67 (1) (1996) 15–41.

[5] E. de Sturler, Inner-outer methods with deflation for linear systems with multiple right-hand sides, in: Householder Symposium XIII, in: Proceedings of the Householder Symposium on Numerical Algebra, Pontresina, Switzerland, 1996, pp. 193–196.
[6] R.B. Morgan, GMRES with deflated restarting, SIAM J. Sci. Comput. 24 (1) (2002) 20–37.
[7] E. de Sturler, Truncation strategies for optimal Krylov subspace methods, SIAM J. Numer. Anal. 36 (3) (1999) 864–889.
[8] K. Ahuja, P. Benner, E. de Sturler, L. Feng, Recycling BiCGStab with an application to parametric model order reduction, SIAM J. Sci. Comput. (2015), in press, available from http://arxiv.org/abs/1406.2831.
[9] J.E. Hicken, D.W. Zingg, A simplified and flexible variant of GCROT for solving nonsymmetric linear systems, SIAM J. Sci. Comput. 32 (3) (2010) 1672–1694.
[10] J. Hicken, M. Osusky, D. Zingg, Comparison of parallel preconditioners for a Newton–Krylov flow solver, in: A. Kuzmin (Ed.), Computational Fluid Dynamics 2010, Springer, Berlin, Heidelberg, 2011, pp. 457–463.
[11] T.M. Leung, D.W. Zingg, Aerodynamic shape optimization of wings using a parallel Newton–Krylov approach, AIAA J. 50 (3) (2012) 540–550.
[12] M.H. Carpenter, C. Vuik, P. Lucas, M. van Gijzen, H. Bijl, A general algorithm for reusing Krylov subspace information. I. Unsteady Navier–Stokes, NASA/Technical Memorandum 2010-216190, 2010.
[13] Q. Niu, L.-Z. Lu, G. Liu, Accelerated GCRO-DR method for solving sequences of systems of linear equations, J. Comput. Appl. Math. 253 (2013) 131–141.
[14] J. Meng, P.-Y. Zhu, H.-B. Li, A block method for linear systems with multiple right-hand sides, J. Comput. Appl. Math. 255 (2014) 544–554.
[15] K. Mohamed, S. Nadarajah, M. Paraschivoiu, Krylov recycling techniques for unsteady simulation of turbulent aerodynamic flows, in: 26th International Congress of the Aeronautical Sciences, Anchorage, Alaska, International Council of the Aeronautical Sciences, 2008.
[16] D.K. Tafti, GenIDLEST – A Scalable Parallel Computational Tool for Simulating Complex Turbulent Flows, 2001, New York 10016–5990.
[17] D.K. Tafti, S.P. Vanka, A numerical study of the effects of spanwise rotation on turbulent channel flow, Phys. Fluids A: Fluid Dyn. (1989–1993) 3 (4) (1991) 642–656.
[18] D.K. Tafti, Time-accurate techniques for turbulent heat transfer analysis in complex geometries, in: R.S. Amano, B. Sundén (Eds.), Computational Fluid Dynamics and Heat Transfer: Emerging Topics, WIT, Southampton, UK, ISBN 978-1-84564-144-3, 2010, pp. 215–264.
[19] P. Gopalakrishnan, D.K. Tafti, A parallel boundary fitted dynamic mesh solver for applications to flapping flight, Comput. Fluids 38 (8) (2009) 1592–1607.
[20] A.R. Amritkar, S. Deb, D.K. Tafti, Efficient parallel CFD-DEM simulations using OpenMP, J. Comput. Phys. 256 (2014) 501–519.
[21] K. Nagendra, D.K. Tafti, K. Viswanath, A new approach for conjugate heat transfer problems using immersed boundary method for curvilinear grid based solvers, J. Comput. Phys. 267 (2014) 225–246.
[22] P. Gopalakrishnan, D.K. Tafti, Effect of wing flexibility on lift and thrust production in flapping flight, AIAA J. 48 (5) (2010) 865–877.
[23] A.R. Amritkar, D.K. Tafti, R. Liu, R. Kufrin, B. Chapman, OpenMP parallelism for fluid and fluid–particulate systems, Parallel Comput. 38 (9) (2012) 501–517.
[24] A. Amritkar, D. Tafti, Computational fluid dynamics computations using a preconditioned Krylov solver on graphical processing units, J. Fluids Eng. 138 (1) (2015) 011402.
[25] Y. Saad, Iterative Methods for Sparse Linear Systems, SIAM, 2003.
[26] G. Wang, D.K. Tafti, Performance enhancement on microprocessors with hierarchical memory systems for solving large sparse linear systems, Int. J. High Perform. Comput. Appl. 13 (1) (1999) 63–79.
[27] A.R. Amritkar, D.K. Tafti, CFD computations using a preconditioned Krylov solver on GPUs, in: ASME 2014 4th Joint US–Europe Fluids Engineering Division Summer Meeting and the 12th International Conference on Nanochannels, Microchannels, and Minichannels, American Society of Mechanical Engineers, 2014.
[28] G. Wang, D.K. Tafti, Uniprocessor performance enhancement with additive Schwarz preconditioners on Origin 2000, Adv. Eng. Softw. 29 (3) (1998) 425–431.
[29] K. Świrydowicz, E. de Sturler, C.J. Roy, X. Xu, Efficient solvers and preconditioners for CFD applications on GPUs, Parallel Comput. (2015), in preparation.
[30] Y. Saad, M.H. Schultz, GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM J. Sci. Stat. Comput. 7 (3) (1986) 856–869.
[31] W.E. Arnoldi, The principle of minimized iterations in the solution of the matrix eigenvalue problem, Q. Appl. Math. 9 (1) (1951) 17–29.
[32] V. Simoncini, D.B. Szyld, Interpreting IDR as a Petrov–Galerkin method, SIAM J. Sci. Comput. 32 (4) (2010) 1898–1912.
[33] H.A. van der Vorst, BI-CGSTAB: a fast and smoothly converging variant of BI-CG for the solution of nonsymmetric linear systems, SIAM J. Sci. Stat. Comput. 13 (2) (1992) 631–644.
[34] K. Ahuja, Recycling Krylov subspaces and preconditioners, Ph.D. thesis, Virginia Tech, Adviser: Eric de Sturler, 2011.
[35] R.D. Moser, J. Kim, N.N. Mansour, Direct numerical simulation of turbulent channel flow up to $Re = 590$, Phys. Fluids 11 (4) (1999) 943–945.