SIAM J. SCI. COMPUT.
Vol. 32, No. 3, pp. 1672–1694

A SIMPLIFIED AND FLEXIBLE VARIANT OF GCROT FOR SOLVING NONSYMMETRIC LINEAR SYSTEMS*

JASON E. HICKEN† AND DAVID W. ZINGG†

Abstract. There is a need for flexible iterative solvers that can solve large-scale ($> 10^6$ unknowns) nonsymmetric sparse linear systems to a small tolerance. Among flexible solvers, flexible GMRES (FGMRES) is attractive because it minimizes the residual norm over a particular subspace. In practice, FGMRES is often restarted periodically to keep memory and work requirements reasonable; however, like restarted GMRES, restarted FGMRES can suffer from stagnation. This has led us to develop a flexible variant of the Krylov linear solver GCROT (generalized conjugate residual with inner orthogonalization and outer truncation). Unlike the original GCROT algorithm, the proposed GCROT variant uses a simplified truncation strategy similar to loose GMRES (LGMRES). This modification is motivated by numerical experiments that suggest the specific subspace retained in the outer iteration of GCROT is less important than its size. The flexible GCROT variant appears to be well suited for advection-dominated problems. In particular, when applied to an adjoint problem from computational aerodynamics, the proposed GCROT variant is robust and efficient compared with several popular truncated Krylov subspace methods. Finally, a flexible version of LGMRES is easily constructed by recognizing algorithmic similarities to GCROT.

Key words. nonsymmetric linear systems, Krylov subspace, iterative methods, truncation, flexible iterations, GMRES, LGMRES, GCROT

AMS subject classification. 65F10

DOI. 10.1137/090754674

1. Introduction and motivation.

Iterative Krylov solvers are now commonly used to solve sparse linear systems that arise from the discretization of partial differential equations (PDEs). In particular, Krylov subspace methods have found success in inexact Newton strategies for solving nonlinear systems of equations; see, for example, [11, 15, 6]. When a Krylov iterative method is used to inexactly solve the linear Newton subsystems, the residual norm is typically reduced by a modest factor $\eta \in [10^{-2}, 10^{-1}]$, relative to the initial residual norm. These large tolerances require few iterations when a good preconditioner is available; hence, optimal methods such as GMRES [18] are often favored in the Newton–Krylov context, since memory requirements remain modest.

Researchers are increasingly interested in optimization problems that are constrained by PDEs. Such problems are often solved using gradient-based optimization algorithms, which require the sensitivities (i.e., gradients) of the functional of interest. These sensitivities can be determined efficiently using the adjoint variables, ψ, which are solutions to linear equations of the form

$$A^T \psi = b,$$

where A is the Jacobian matrix of the discretized PDE. In contrast to the linear systems that arise during the Newton–Krylov solution of the discretized PDE, the

*Received by the editors April 1, 2009; accepted for publication (in revised form) March 29, 2010; published electronically June 9, 2010. This work was supported by the Natural Sciences and Engineering Research Council (NSERC), the Canada Research Chairs program, Bombardier Aerospace, Mathematics of Information Technology and Complex Systems (MITACS), and the University of Toronto.

http://www.siam.org/journals/sisc/32-3/75467.html

†Institute for Aerospace Studies, University of Toronto, Toronto M3H 5T6, ON, Canada (jehicken@oddjob.utias.utoronto.ca, dwz@oddjob.utias.utoronto.ca).

adjoint system must be solved to a much smaller tolerance to ensure that the gradient is accurate. For example, a tolerance of $\eta = 10^{-8}$ is necessary to keep the norm of the gradient error below 10^{-6} in aerospace applications [8, 10].

The tighter tolerance demanded by the adjoint equations requires significantly more iterations, making full GMRES prohibitively expensive for three-dimensional problems involving millions of degrees of freedom. In such situations, when many iterations are needed, restarted GMRES is often employed to limit memory usage, despite its tendency to stall. BiCGStab [21] offers an alternative to restarted GMRES, but this method is not well suited to advection-dominated steady-state problems arising in many of the physical sciences [7]. Researchers have proposed numerous variants of BiCGStab and GMRES(m) to address the above shortcomings; see Simoncini and Szyld [20] for a review. One of the more promising Krylov subspace algorithms is GCROT [4] (generalized conjugate residual with inner orthogonalization and outer truncation).

Despite its promise, in the decade following the original GCROT paper very few application-based articles have been published that use this solver. We speculate that there are two reasons for this. First, GCROT is, in its full generality, a complex algorithm to implement. Second, users must supply five input parameters to GCROT, the choice of which is nontrivial and requires some knowledge of the method.

In an effort to reduce the burden of determining optimal parameters, we present a variant of GCROT, GCROT(m, k), that uses a simplified truncation strategy. The proposed method requires two parameters: an inner subspace size, m, and an outer subspace size, k. In addition to requiring fewer input parameters, GCROT(m, k) is straightforward to implement, and we hope that this will encourage others to experiment with GCROT.

Together with the simplified GCROT algorithm, we also present flexible versions of GCROT and loose GMRES (LGMRES) [1]. Flexible iterative methods allow the preconditioner to vary with each step. Thus, the preconditioner can use relaxation-based methods, or even nested Krylov methods. An example of a nonstationary preconditioner, which we have found useful in computational aerodynamics, is the parallel approximate-Schur preconditioner of Saad and Sosonkina [19].

The paper is organized as follows. We begin in section 2 with a review of generalized conjugate residual with inner orthogonalization (GCRO), the method underlying GCROT, and present a variant that uses flexible GMRES (FGMRES) as the inner method. In section 3, we investigate the residual error analysis proposed by de Sturler [4], by conducting some numerical experiments. These experiments lead to the truncation strategy used in GCROT(m, k). In section 4, we briefly discuss the flexible variant of LGMRES and its properties. Finally, in section 5, we assess the new variants by comparing their performance with that of GMRES(m), FGMRES(m), BiCGStab, and flexible BiCGStab (FBiCGStab) [23]. Conclusions can be found in section 6.

2. GCRO with FGMRES nested. In this section, we present a version of GCRO [3] that uses FGMRES as the inner Krylov method; this version of GCRO will subsequently be used to develop GCROT(m, k). We also use this as an opportunity to review GCRO in general and establish an optimality result for the flexible version of GCRO.

GCRO is essentially a generalization of the generalized conjugate residual (GCR) method [5]. Consider the linear system $Ax = b$ with $A \in \mathbb{R}^{n\times n}$. Let $U_k, C_k \in \mathbb{R}^{n\times k}$

be given matrices that satisfy

$$AU_k = C_k, \tag{2.1}$$

$$C_k^T C_k = I_k, \tag{2.2}$$

where I_k is the $k \times k$ identity matrix. In addition, let x_k and $r_k \in \mathbb{R}^n$ be the approximate solution and residual, respectively, at iteration k; throughout this paper we will assume that the initial iterate is $x_0 = 0$. Suppose we look for $x_k \in \text{range}\,(U_k)$ such that the residual norm is minimized. This assumption, together with (2.1) and (2.2), leads to

$$r_k = b - AU_k y_k = b - C_k y_k,$$

where $y_k \in \mathbb{R}^k$ is given by

$$y_k = \underset{y}{\text{argmin}}\, \|b - C_k y\|_2 = C_k^T b.$$

Thus, we obtain $x_k = U_k C_k^T b$ and $r_k = b - C_k C_k^T b$.

The question remains of how to generate u_{k+1} and c_{k+1} for the subsequent iteration. Suppose, for the moment, that it would be possible to take $c_{k+1} = r_k$ for iteration $k+1$. Then $C_{k+1} = [C_k\ r_k]$, and if we take $y_{k+1}^T = [y_k^T\ 1]$, the residual at iteration $k+1$ is given by

$$\begin{aligned} b - AU_{k+1}y_{k+1} &= b - \begin{bmatrix} C_k & r_k \end{bmatrix} \begin{bmatrix} y_k \\ 1 \end{bmatrix} \\ &= b - C_k C_k^T b - r_k \\ &= 0. \end{aligned}$$

This shows that taking $c_{k+1} = r_k$ is optimal; however, this choice implies $u_{k+1} = A^{-1} r_k = e_k$ (the error), which requires inverting the matrix. Nevertheless, it does suggest that we take $u_{k+1} \approx e_k$ and $c_{k+1} = Au_{k+1}$. This strategy is applied in the GMRESR [22] algorithm by using GMRES to find a u_{k+1} that approximates the error.

GCRO improves upon GMRESR by requiring that the inner method maintain orthogonality to C_k. Orthogonality to C_k is ensured by replacing A in the Arnoldi iteration with $A_{C_k} = (I - C_k C_k^T)A$.

In general, GCRO can use any Krylov-based iterative solver for the inner method. We consider FGMRES since this method is flexible and because it leads to the following global optimality result.

THEOREM 1. *Let Z_m and V_{m+1} be the matrices generated from FGMRES(m) using an Arnoldi iteration with $A_{C_k} = (I - C_k C_k^T)A$ and $v_1 = r_k / \|r_k\|_2$; hence, $A_{C_k} Z_m = V_{m+1} \bar{H}_m$ and $V_{m+1}^T V_{m+1} = I_{m+1}$ [16]. Let y_m be the solution of the inner, FGMRES(m), iteration:*

$$y_m = \underset{y}{\text{argmin}}\, \|r_k - A_{C_k} Z_m y\|_2 = \underset{y}{\text{argmin}}\, \|r_k - V_{m+1} \bar{H}_m y\|_2.$$

Then the minimal residual solution of the (inner) FGMRES method gives the outer approximation

$$x_{k+1} = x_k + (I - U_k C_k^T A) Z_m y_m,$$

which is also the solution of the global minimization problem

$$\min\{\|b - A\tilde{x}\|_2 \mid \tilde{x} \in \text{range}\,(U_k) \oplus \text{range}\,(Z_m)\}.$$

Proof. The proof is analogous to the proof for Theorem 2.2 from [3], with Z_m replacing V_m and $A_{C_k} Z_m = V_{m+1}\bar{H}_m$ replacing $A_{C_k} V_m = V_{m+1}\bar{H}_m$. ∎

Theorem 1 explains why GCRO is preferable to GMRESR: the former finds the solution with the minimal residual norm in the space $\text{range}\,(U_k) \oplus \text{range}\,(Z_m)$.

To make these ideas more concrete, the pseudocode for GCRO using FGMRES as the inner method is presented in Algorithm 1 in the appendix. As is the case with GCRO nested with GMRES [3], more efficient implementations of FGMRES-nested GCRO are possible, but many of the changes are useful only for full GCRO and must be discarded when we consider truncation.

3. Truncation strategies for GCRO. The memory requirements of GCRO grow linearly with each outer iteration, since the vectors c_{k+1} and u_{k+1} are added to the matrices C_k and U_k. In [4], de Sturler addresses this by developing GCRO truncated, or GCROT. The truncation strategy in GCROT examines the residual error to determine which subspace was most important to convergence at outer iteration k. The strategy assumes that a subspace that was important for convergence will continue to be important, and that a subspace that did not contribute to convergence will continue to be unimportant. These assumptions will be tested below. However, we begin by reviewing the theory presented in [4] in the context of using FGMRES as the inner method in GCRO.

The following analysis is motivated by the question, Which subspace from $\text{range}\,(C_k)$ was most important for convergence during the inner FGMRES iteration of GCRO? Recall that the inner FGMRES method produces the relation (see Theorem 1)

$$\begin{aligned} V_{m+1}\bar{H}_{m+1} &= A_{C_k} Z_m \\ &= AZ_m - C_k C_k^T AZ_m \\ \Leftrightarrow \qquad AZ_m &= C_k B + (V_{m+1} G_m)\bar{R}_m, \end{aligned} \tag{3.1}$$

where $B \equiv C_k^T AZ_m \in \mathbb{R}^{k\times m}$. We have also introduced $G_m \in \mathbb{R}^{(m+1)\times(m+1)}$, which is the product of the m Givens rotations[1] that reduce the upper Hessenberg matrix $\bar{H}_m$ to an upper triangular matrix, denoted here by $\bar{R}_m \in \mathbb{R}^{(m+1)\times m}$. Dropping the last row of zeros from $\bar{R}_m$ to obtain R_m and the last column of G_m to obtain $\bar{G}_m$, we arrive at the QR-decomposition $V_{m+1}\bar{H}_{m+1} = Q_m R_m$, where $Q_m \equiv V_{m+1}\bar{G}_m$. Using this decomposition in (3.1), we have

$$AZ_m = C_k B + Q_m R_m. \tag{3.2}$$

Recall that V_{m+1} is constructed to satisfy $C_k^T V_{m+1} = 0$, so $C^T Q_m = 0$ as well. Consequently, (3.2) shows us how to express AZ_m using the orthonormal basis $[C_k\ Q_m]$. Alternatively, if we ignore orthogonality to the subspace $\text{range}\,(C_k)$, we can obtain the following QR-decomposition of AZ_m:

$$AZ_m = W_m S, \tag{3.3}$$

where $S \in \mathbb{R}^{m\times m}$ is upper triangular and $W_m^T W_m = I_m$. It follows from (3.2) and (3.3) that the best approximation to the residual r_k in $(\text{range}\,(AZ_m) \oplus \text{range}\,(C_k))^{\perp}$

[1] These rotations are available in most implementations of GMRES and FGMRES.

is (recall $C_k^T r_k = 0$)

$$\begin{aligned} r_k^{(1)} &= (I - C_k C_k^T - Q_m Q_m^T) r_k \\ &= (I - Q_m Q_m^T) r_k, \end{aligned}$$

and the best approximation to r_k in range $(AZ_m)^\perp$, ignoring orthogonality to range (C_k), is

$$r_k^{(2)} = (I - W_m W_m^T) r_k.$$

The residual error is defined as the difference between $r_k^{(1)}$ and $r_k^{(2)}$:

$$\boldsymbol{\varepsilon} \equiv r_k^{(2)} - r_k^{(1)}.$$

We now paraphrase Theorem 2.2 from [4], which provides an inexpensive means of orthogonally decomposing the residual error and calculating its norm.

THEOREM 2. *Let $D = BR_m^{-1}$, and let the singular value decomposition of D be given by*

$$D = \Lambda_D \Sigma_D \Gamma_D^T,$$

where the columns of the unitary matrices $\Lambda_D = [\lambda_1 \ \lambda_2 \ \cdots \ \lambda_k] \in \mathbb{R}^{k \times k}$ and $\Gamma_D = [\gamma_1 \ \gamma_2 \ \cdots \ \gamma_m] \in \mathbb{R}^{m \times m}$ are ordered such that the singular values in Σ_D satisfy

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p,$$

$p = \min(k, m)$. Then the residual error satisfies

$$\boldsymbol{\varepsilon} = \sum_{i=1}^{p} \boldsymbol{\varepsilon}_i, \qquad \text{where} \qquad \boldsymbol{\varepsilon}_i = \left(\frac{\nu_i \sigma_i^2}{1 + \sigma_i^2} Q_m v_i - \frac{\nu_i \sigma_i}{1 + \sigma_i^2} C_k y_i \right)$$

and $\nu_i = \gamma_i^T Q_m^T r_k$. The norm of the residual error is given by

$$\|\boldsymbol{\varepsilon}\|_2 = \left(\sum_i^p \|\boldsymbol{\varepsilon}_i\|_2^2 \right)^{\frac{1}{2}}, \qquad \text{where} \qquad \|\boldsymbol{\varepsilon}_i\|_2^2 = \frac{\nu_i^2 \sigma_i^2}{1 + \sigma_i^2}.$$

We consider Theorem 2 as it applies to the residual error in neglecting subspaces from C_k. When the inner method of GCRO is GMRES, the theorem can be applied much more generally [4]. For example, by identifying the first s iterations of the inner method with C_s, we can, with the help of the residual error, determine how restarting at iteration $s+1$ would affect convergence in the last $m-s$ iterations. Unfortunately, it does not appear possible to extend this more general analysis to the case where the inner method of GCRO is FGMRES.[2] Even if such an analysis were possible, we could keep only a small subspace from V_{m+1} before the potential residual reduction in subsequent iterations would be outweighed by the CPU cost of moving the subspace into C_k. For these reasons, we favor a truncated version of flexible GCRO that keeps only the residual updates, c_k, in the matrix C_k.

If the dimension of C_k is limited to k, then the subspace range (C_k) must be truncated for outer iterations greater than $k+1$. We consider the following three truncation strategies here.

[2] In particular, the matrix S in (3.3) cannot be constructed using an Arnoldi iteration based on $\bar{H}_m$, in contrast to the case where GMRES is the inner method; see page 873 of [4].

TABLE 1
Summary of cases used to study the truncation strategy for GCRO.

Case	Name	Size	RHS?	Tolerance
1	add20	2395	yes	10^{-10}
2	osreg_1	2205	no	10^{-10}
3	orsirr_1	1030	no	10^{-5}
4	cdde2	961	no	10^{-10}
5	pde900	900	no	10^{-10}
6	sherman1	1000	yes	10^{-5}
7	sherman4	1104	yes	10^{-10}
8	rdb1250	1250	no	10^{-5}
9	cavity05	1182	yes	10^{-5}
10	nos3	960	no	10^{-5}
11	watt_2	1856	no	10^{-5}
12	fs_760_1	760	no	10^{-10}
13	er05r0000	236	yes	10^{-5}
14	steam2	600	no	10^{-5}
15	cavity10	2597	yes	10^{-5}
16	example6 ($D = 1$)	1600	yes	10^{-10}
17	example6 ($D = 41$)	1600	yes	10^{-10}
18	example6 ($D = 41^2$)	1600	yes	10^{-10}

(1) *Smallest singular value.* The first strategy is the one proposed in [4]; namely, we discard the vector associated with the smallest singular value. Thus, we keep the subspace

$$C_k[\lambda_1\ \lambda_2\ \cdots\ \lambda_{k-1}\,].$$

If we had been limited to using a $(k-1)$-dimensional subspace from range (C_k) during the inner iteration, Theorem 2 tells us that the above choice is the one that minimizes the residual error. Note, however, that there is no way of knowing a priori that this will continue to be the optimal choice. This is an assumption that we must test.

(2) *Largest singular value.* In this strategy, we discard the vector associated with the largest singular value, keeping the subspace

$$C_k[\lambda_2\ \lambda_3\ \cdots\ \lambda_k\,].$$

This choice may seem counterintuitive, given Theorem 2. It provides a test for the assumptions used for the first choice.

(3) *Oldest vector in C_k.* The final strategy we consider is modelled on the one used in LGMRES. Specifically, we discard the oldest vector in C_k and keep

$$C_k\left[e_2\ e_3\ \cdots\ e_k\right],$$

assuming the vectors are stored columnwise, oldest to newest, in C_k.

Following [1], we assess the truncation strategies using a set of 15 problems from the Matrix Market Collection [2, 12], as well as the three convection-diffusion problems from Example 6 in [14]. All 18 cases are summarized in Table 1. The table indicates the names of the matrices, their size, and whether or not the right-hand-side vector is provided. If the right-hand side is not available, we pseudo-randomly generate the entries in b such that $b_i \in [0, 1]$.

The convection-diffusion problems, denoted example6 in Table 1, are based on a finite-difference discretization of the PDE

$$u_{xx} + u_{yy} + Du_x = -41^2.$$

The parameter D determines the relative importance of convection over diffusion and takes on values $D = 1$, 41, and 41^2. The domain consists of a unit square, and it is discretized using a uniform mesh with $\Delta x = \Delta y = 1/41$. Second-order centered schemes are used to discretize the derivatives, and Dirichlet boundary conditions of $u = 0$ are applied along the edges of the domain.

Table 1 also lists the tolerances that determine when the iterative solution process terminates. A particular problem is considered solved when the relative residual, $\|r\|_2/\|b\|_2$, is reduced below the given tolerance. The tolerance for each case is chosen to ensure that the number of matrix-vector products falls approximately in the range $[10^2, 10^4]$ and that truncation of C_k is necessary.

Figures 1(a)–(c) compare the three truncation strategies for $m + k = 10$, 20, and 30, respectively, where m is the size of the inner subspace and k is the dimension of C_k (the outer subspace). The dimension of range (C_k) is allowed to vary from $k = 1$ to $k = m$. Note that if $k > m$, there will always be a subspace of range (C_k) that is orthogonal to the residual error, so the upper bound on k is required for the first truncation strategy to remain well defined. The data points in the figures indicate the average number of matrix-vector products over all k, while the upper and lower "error" bars indicate the maximum and minimum number of products required.

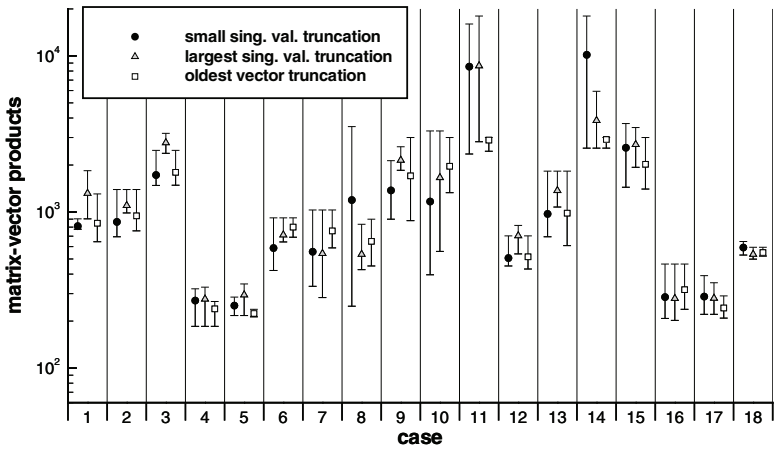
The results suggest that truncating C_k based on the smallest singular value does not offer a significant improvement over the other strategies. Indeed, for $m + k = 20$ this strategy performs the worst on average for five of the cases. Similarly, if the singular value analysis can predict which subspace should be kept, then truncating based on the largest singular value should perform rather poorly. Instead, the overall performance of this strategy is very similar to those of the other two.

3.1. GCROT(m, k). Despite its simplicity, the strategy of discarding the oldest vector in C_k is competitive with truncation based on the singular-value analysis. Discarding the oldest vector is also attractive because there is no need to apply Givens rotations to eliminate a column from C_k to make space for the new residual update, unlike the singular-value-based truncation.

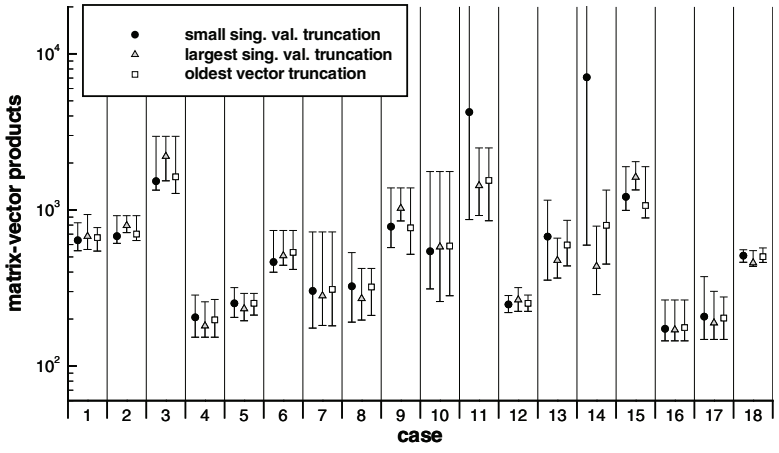
This simplified truncation strategy is a variant of GCROT that we refer to as GCROT(m, k). This variant can use either GMRES or FGMRES as the inner method; when FGMRES is used we refer to the solver as flexible GCROT(m, k) or F-GCROT(m, k). A listing for F-GCROT(m, k) is provided in the appendix.[3] Note the use of FGMRES$(m + \max(k - l, 0))$ as the inner method, where l is the outer iteration. Thus, the method begins with FGMRES$(m+k)$ and progressively decreases the number of inner iterations as the size of C_k grows, to keep memory requirements fixed. A similar strategy is used in LGMRES [1].

Table 2 lists the computational costs of F-GCROT(m, k) in terms of the number of matrix-vector products (matvecs), scalar-vector-plus-vector operations (daxpys), and inner products (ddots). The costs for the first k outer iterations differ from those of the subsequent iterations, so these are listed separately. The total memory requirements are also listed in the table in terms of the number of vectors of length n. The memory requirements are based on an actual implementation, which modifies Algorithm 2 such that the residual is stored in v_1, one work array is used for both w and $\hat{u}$, and $\hat{c}$ is stored directly in C_k. The nonflexible GCROT(m, k) algorithm has identical costs but uses only $m + 2k + 3$ vectors.

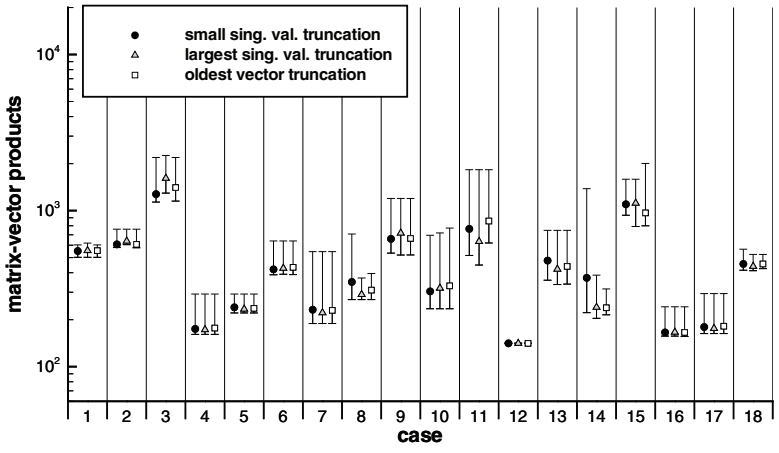
[3]The nonflexible version can be obtained by replacing FGMRES with GMRES in the inner method, which makes the z_i vectors redundant.



(a) $m + k = 10$, $k = 1, \ldots, 5$



(b) $m + k = 20$, $k = 1, \ldots, 10$



(c) $m + k = 30$, $k = 1, \ldots, 15$

FIG. 1. *Number of iterations used by the three GCRO truncation strategies to converge the test cases for various values of m and k.*

TABLE 2
Computational costs and memory requirements of flexible GCROT(m, k) and flexible LGMRES.

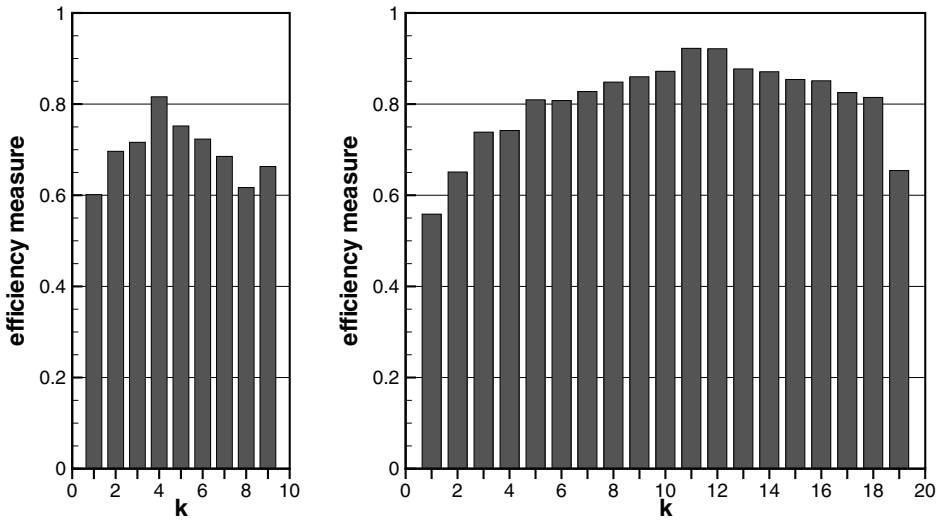
Flexible GCROT(m,k)	
Memory	$2m+2k+3$
Cost (first k outer iterations)	
matvecs	$mk+\frac{1}{2}k(k+1)$
daxpys	$\frac{1}{2}k(m+k+1)(m+k+6)-\frac{1}{6}k(k+1)(k+5)+4k$
ddots	$\frac{1}{2}k(m+k+1)(m+k+2)-\frac{1}{6}k(k+1)(k+2)+3k$
Cost (every subsequent outer iteration)	
matvecs	m
daxpys	$\frac{1}{3}(m+k)(m+k+7)-\frac{1}{2}k(k+5)+5$
ddots	$\frac{1}{2}(m+k)(m+k+3)-\frac{1}{2}k(k+3)+3$
Flexible LGMRES(m,k)	
Memory	$2m+3k+3$
Cost (first k outer iterations)	
matvecs	$mk+\frac{1}{2}k(k+1)$
daxpys	$\frac{1}{2}k(m+k+1)(m+k+6)$
ddots	$\frac{1}{2}k(m+k)(m+k+1)+k$
Cost (every subsequent outer iteration)	
matvecs	m
daxpys	$\frac{1}{2}(m+k)(m+k+7)+3$
ddots	$\frac{1}{2}(m+k)(m+k+3)+1$

Although GCROT(m,k) is a variant of GCROT, the algorithm is sufficiently distinct that some discussion regarding appropriate values for the parameters m and k is warranted. The parameter m determines the size of the Krylov subspace used in the inner (F)GMRES iterations. In our applications, we typically use values of m between 10 and 50, with the larger values giving more robust convergence at the expense of memory. To determine appropriate values for k, we performed a parameter study using the same cases from the truncation strategy study. For each case, we fixed $s = m + k$ and varied the value of k from 1 to $s - 1$, and we recorded the number of matrix-vector products. We define an efficiency measure for each value k using

$$\eta_k = \frac{1}{N_{\text{cases}}} \sum_{i=1}^{N_{\text{cases}}} \frac{\min_k \left(L_{i,k}\right)}{L_{i,k}},$$

where $L_{i,k}$ is the number of matrix-vector products used by GCROT(m,k) on case i and $N_{\text{cases}} = 18$ is the total number of cases. Hence, $\eta_k \leq 1$, with equality if and only if GCROT(m,k) used the fewest products in all cases. If GCROT(m,k) stalled or failed to converge in fewer than 30000 products for case i, the ratio $\min_k \left(L_{i,k}\right)/L_{i,k}$ was set to zero.

Figure 2 plots the efficiency measure η_k for $m + k = 10$ and $m + k = 20$; similar results were obtained for other values of $m + k$. The results suggest that $k \approx m$ is a good choice on average, at least in terms of matrix-vector products. When the computational cost of the daxpys and ddots are taken into consideration, lower values of k may also provide good performance in terms of CPU time. In section 5.2, we will revisit the choice of k in the context of an adjoint problem.



(a) $m + k = 10$ (b) $m + k = 20$

FIG. 2. *Performance of GCROT(m, k) for fixed $m + k$ and various values of k.*

4. Flexible LGMRES(m, k). A flexible variant of LGMRES is obtained with only a few changes to the original algorithm in [1]. See Algorithm 3 in the appendix for a listing of flexible LGMRES (F-LGMRES). As with F-GCROT, F-LGMRES is built on FGMRES as the inner Krylov method. Thus, since an additional vector of storage is required for each iteration of the inner method, the total memory requirements increase from $m + 3k + 3$ to $2m + 3k + 3$. The memory and computational costs of F-LGMRES(m, k) are summarized in Table 2.

Several theorems from [1] that apply to GMRES or LGMRES can be extended to their flexible variants; these theorems are repeated below for completeness. In fact, the theorems apply more broadly to any Petrov–Galerkin projection method for which $u_i \equiv x_i - x_{i-1} \in \mathcal{K}$ and $r_i \perp \mathcal{L} = A\mathcal{K}$.

THEOREM 3. *Let r_{i+1} and r_i be the residuals from the consecutive FGMRES cycles $i + 1$ and i, respectively. Then the angle between these residuals is given by*

$$\cos \angle(r_{i+1}, r_i) = \frac{\|r_{i+1}\|_2}{\|r_i\|_2}.$$

THEOREM 4. *Let r_{i+1} and r_{i-1} be the residuals from the FGMRES restart cycles $i + 1$ and $i - 1$, respectively. Then the angle between these residuals is*

$$\cos \angle(r_{i+1}, r_{i-1}) = \frac{\|r_{i+1}\|_2}{\|r_{i-1}\|_2} - \frac{(Au_{i+1}, Au_i)}{\|r_{i+1}\|_2 \|r_{i-1}\|_2},$$

where $u_{i+1} = x_{i+1} - x_i$ and $r_{i+1} = r_i - Au_{i+1}$ for all $i \geq 1$.

THEOREM 5. *Let r_{i+1} and r_{i-1} be the residuals from the F-LGMRES restart cycles $i + 1$ and $i - 1$, respectively. Then the angle between these residuals is given by*

$$\cos \angle(r_{i+1}, r_{i-1}) = \frac{\|r_{i+1}\|_2}{\|r_{i-1}\|_2}.$$

Proofs of the above theorems are easily obtained by generalizing the proofs of the corresponding theorems in [1].

Theorems 4 and 5 suggest that F-LGMRES will improve upon FGMRES in much the same way that LGMRES improves upon GMRES: by increasing the angle between r_{i+1} and r_{i-1}, thereby reducing the tendency for these residuals to point in the same direction.

5. Results. In this section, we consider two problems that illustrate the performance of (F)GCROT(m,k) and, to a lesser extent, F-LGMRES(m,k). The problems reflect our interest in the advection-dominated physics that typically arises in computational fluid dynamics. In particular, we choose problems that result in ill-conditioned, indefinite sparse matrices.

5.1. Advection-diffusion. The first example is a two-dimensional steady advection-diffusion problem governed by the PDE

$$\partial_x u - \nu \nabla^2 u = 0, \tag{5.1}$$

where ∇^2 is the Laplacian and ν is the diffusion coefficient. The domain consists of the unit square $[0,1] \times [0,1]$, which is discretized uniformly with 46 nodes in both the x- and y-directions. The PDE (5.1) is discretized using a second-order accurate finite-volume scheme.

Dirichlet boundary conditions of $u = 0$ are applied along the edges $y = 0$ and $y = 1$. Dirichlet boundary conditions are also applied at the inlet, $x = 0$, where we set

$$u(x=0) = \begin{cases} 1 & \text{if } |y - \frac{1}{2}| \leq \frac{1}{6}, \\ 0 & \text{otherwise.} \end{cases}$$

Neumann boundary conditions of $\partial_x u = 0$ are enforced at the outlet, $x = 1$. To improve equation scaling, the boundary equations are multiplied by ν. The diffusion coefficient is set to $\nu = 0.2(\Delta x)^2 \approx 10^{-4}$. All solvers are given an initial guess of $u = 0$.

To investigate the flexible solvers, we use GMRES(5) as a preconditioner and measure performance in terms of matrix-vector products. Thus, each iteration requires six matrix-vector products when the GMRES(5) preconditioner is used—five products for the preconditioner and one for the solver itself. We consider three subspace sizes, $m + k = 16$, 20, and 24, and we compare FGMRES$(m + k)$ with the flexible variants of GCROT(m,k) and LGMRES(m,k).

Table 3 lists the number of matrix-vector products required to reduce the relative residual of the advection-diffusion problem below 10^{-10}. Runs that failed to reach this tolerance in 10^5 products were terminated and are indicated with a dash in the table. The truncated solvers were run with all values of k between 1 and m, although only three representative values are shown in Table 3, specifically, $k = 1, m/3$, and m.

When no preconditioning is used, GCROT is the only method of the three that consistently solves the problem in fewer than 10^5 products. Admittedly, Krylov solvers are rarely used without a preconditioner; nevertheless, these results suggest that GCROT(m,k) is a robust solver. Indeed, $(m,k) = (15,1)$ is the only parameter combination considered that results in the failed convergence of GCROT.

As expected, all the methods perform significantly better with GMRES(5) preconditioning. The improvement is particularly dramatic for FGMRES$(m + k)$ and F-LGMRES(m,k), which both failed to converge in 10^5 products without the preconditioner. F-GCROT(m,k) has the best performance, with $k = m$ requiring the fewest

TABLE 3
Matrix-vector products required to reach convergence tolerance 10^{-10} for the advection-diffusion problem.

		Matrix-vector products					
		No preconditioner			GMRES(5)		
Solver	$m+k$	16	20	24	16	20	24
FGMRES($m+k$)		—	—	—	3846	2046	1188
F-GCROT($m, k=1$)		—	32862	5686	1746	1362	1092
F-GCROT($m, k=\frac{1}{3}m$)		3787	1829	2246	942	912	696
F-GCROT($m, k=m$)		2803	2562	2784	756	780	696
F-LGMRES($m, k=1$)		—	—	—	4002	2184	1230
F-LGMRES($m, k=\frac{1}{3}m$)		—	—	—	—	2520	924
F-LGMRES($m, k=m$)		—	—	—	—	1098	924

products. The performance of F-LGMRES(m, k) is similar to FGMRES$(m + k)$ on this particular problem.

We observe the following trends regarding the choice of k for the present advection-dominated problem. GCROT(m, k) and its flexible variant are relatively insensitive to this parameter. For all $k \leq m$, only $k = 1$ leads to convergence difficulties. For the preconditioned cases, $k = m$ was the optimal choice for F-GCROT for each value of $m + k$. For the unpreconditioned cases, there was more variability in the optimal choice of k for GCROT; $k = 8$, 3, and 3 produced the fewest products (2803, 1628, and 1116) for $m + k = 16$, 20, and 24, respectively.

In [1], LGMRES(m, k) was found to perform well with values of k less than or equal to 3. This is consistent with the present results for F-LGMRES(m, k) when $m + k = 16$ ($k = 1$ gives the fewest matrix-vector products in this case). However, for the subspace sizes $m + k = 20$ and 24, larger values of k performed better. In particular, the value $k = m$ was optimal, using 25%–50% fewer matrix-vector products relative to values $k \leq 3$.

Finally, we emphasize that the present comparison is based on the subspace size, $m + k$, and not on memory. This distinction is important, because the memory for flexible LGMRES grows as $2m + 3k + 3$, so fixing the subspace size at $m + k$ ignores a nonconstant k term. A comparison based on fixed memory is considered in the next section.

5.2. Adjoint equation for a compressible flow. In this example, we investigate the performance of F-GCROT(m, k) on an adjoint problem from computational aerodynamics. Such problems frequently arise in aerodynamic optimization during the evaluation of functional gradients. The problem is an appropriate test case for two reasons. First, many iterations are necessary to achieve an accurate gradient, so full GMRES and full FGMRES are not practical. Second, as we shall see, a nonstationary preconditioner may be optimal in some applications, so a flexible variant is useful.

The matrix for the adjoint problem is the transpose of a Jacobian matrix, denoted by A. The Jacobian matrix arises from the linearization of the discrete Euler equations. The Euler equations are discretized using second-order-accurate finite differences; see [9] for further details. The right-hand-side vector is the derivative of a functional (e.g., lift or drag) with respect to the flow variables at each node.

The properties of the adjoint matrix are strongly influenced by the Mach num-

ber. The Mach number is defined by $M \equiv u_\infty/a_\infty$, where u_∞ is the far-field flow velocity and a_∞ the far-field speed of sound. For Mach numbers below 0.3, the flow is essentially incompressible. For larger Mach numbers, compressibility effects become important—for the geometry considered here, a shock wave appears at $M \approx 0.7$ and grows in strength as the Mach number is increased. To cover both incompressible and compressible regimes, we consider a range of Mach numbers between $M = 0.2$ and $M = 0.8$.

There are 5791500 unknowns, so a parallel solution strategy is necessary for reasonable CPU times. The matrix and vectors (solution and right-hand side) are partitioned across 12 processors. The distributed matrix is stored in the format described in [17] and [19] such that the equations and unknowns corresponding to the same node are assigned to the same processor.

We consider two parallel preconditioners for the adjoint problem: an additive-Schwarz preconditioner with no overlap (block Jacobi), and an approximate-Schur preconditioner [19]. The Schwarz preconditioner consists of a single application of an incomplete lower-upper factorization [13] to the diagonal blocks of A^T. It does not require a flexible iterative solver. The Schur-based preconditioner couples the entire domain by finding an approximate solution to the Schur complement corresponding to interface unknowns, i.e., unknowns coupled between processors. This approximate solution uses GMRES, and, consequently, the approximate-Schur preconditioner requires a flexible iterative solver.

For the same nested subspace size, the flexible variants of GCROT and LGMRES require more memory than their GMRES-based counterparts. Therefore, we need to demonstrate that the flexible variants can perform better using the same amount of memory and, hence, smaller subspaces. For example, suppose we pair the approximate-Schur preconditioner with F-GCROT(m, k). The flexible variant is unnecessary for the additive-Schwarz preconditioner, so FGMRES can be replaced with GMRES in Algorithm 2. Consequently, the set of vectors $\{z_i\}_{i=1}^m$ becomes redundant, and the additional memory can be used to expand the subspace sizes. Thus, to provide a fair comparison in terms of memory, the additive-Schwarz preconditioner should be paired with GCROT$(2m, k)$. Similar arguments can be applied to LGMRES.

We have implemented the solvers and preconditioners in an in-house Fortran95 library, which we have used in the following studies. In addition, the results were obtained on the resources of SciNet, an IBM iDataPlex cluster based on Intel's Nehalem architecture. The cluster is interconnected with nonblocking 4X-DDR InfiniBand.

5.2.1. The choice of k in GCROT(m, k). In section 3.1 we investigated appropriate values for the parameter k in GCROT(m, k). Here, we use the adjoint problem to explore this issue further. In addition, we compare the Schur and Schwarz preconditioners applied to GCROT(m, k), which provides an opportunity to assess the benefits of a flexible version of this solver.

We consider the set of Mach numbers

$$M_i = 0.2 + (i-1)0.5, \qquad i = 1, 2, \ldots, 13, \tag{5.2}$$

and solve the adjoint problem using F-GCROT$(20-k, k)$ paired with the approximate-Schur preconditioner and GCROT$(40-2k, k)$ paired with the additive-Schwarz preconditioner. The parameter k is varied from 1 to 19. This covers the full range of k values for F-GCROT$(20-k, k)$ and approximately half the range for GCROT$(40-2k, k)$.

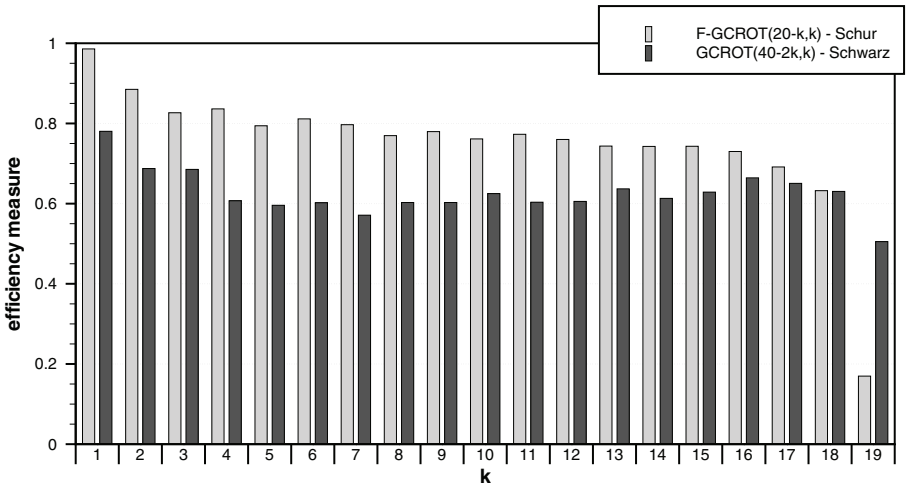


FIG. 3. *Efficiency measure, over a range of k values, for F-GCROT$(20-k,k)$ and for GCROT$(40-2k,k)$ paired with the Schur- and Schwarz-based preconditioners, respectively.*

For each k value, we define an efficiency measure as follows:

$$\eta_k = \frac{1}{13}\sum_{i=1}^{13}\frac{T_i^{\min}}{T_{i,k}},$$

where $T_{i,k}$ is the time required by GCROT or F-GCROT, using the parameter value k, to reduce the relative residual below 10^{-8} for Mach number M_i. $T_i^{\min}$ is the minimum time for Mach number M_i and is taken over all k values and both preconditioner-solver combinations; consequently, η_k permits a direct comparison of the two combinations. If the solver fails to converge in 3000 matrix-vector products, $T_i^{\min}/T_{i,k}$ is set to zero. The above measure is similar to the one used in section 3.1, except that CPU time is used rather than matrix-vector products. CPU time is preferable in the present context, since it accounts for the unequal costs of the two preconditioners.

Figure 3 plots the efficiency measure for the F-GCROT$(20-k,k)$-Schur combination and the GCROT$(40-2k,k)$-Schwarz combination. For $k \le 15$, F-GCROT outperforms GCROT by 15%–28% (for a fixed k), despite the smaller inner subspace size used by the flexible variant. On this set of problems, GCROT(m,k) and F-GCROT$(m,k,)$ perform best using small values of k. This does not necessarily contradict the findings of section 3.1, since those results were obtained by averaging over a range of distinct problems; thus, $k = m$ may still be optimal on average. Moreover, the present efficiency measure is based on the CPU time, which includes the advantage that smaller values of k have in terms of daxpys.[4] Both the earlier and present results agree that the largest k value (i.e., $m = 1$) should be avoided. Finally, while $k = 1$ performs best, the figure shows that the performance for other values of k is reduced by only a modest amount (approximately 10%–30% for $k \le 17$).

5.2.2. The choice of k in LGMRES(m,k). In this section, we adapt the methodology of section 5.2.1 to LGMRES and its flexible variant to determine values

[4]The $k = 1$ case is also the most efficient in terms of matrix-vector products, which is why fewer daxpys offers only a partial explanation here.

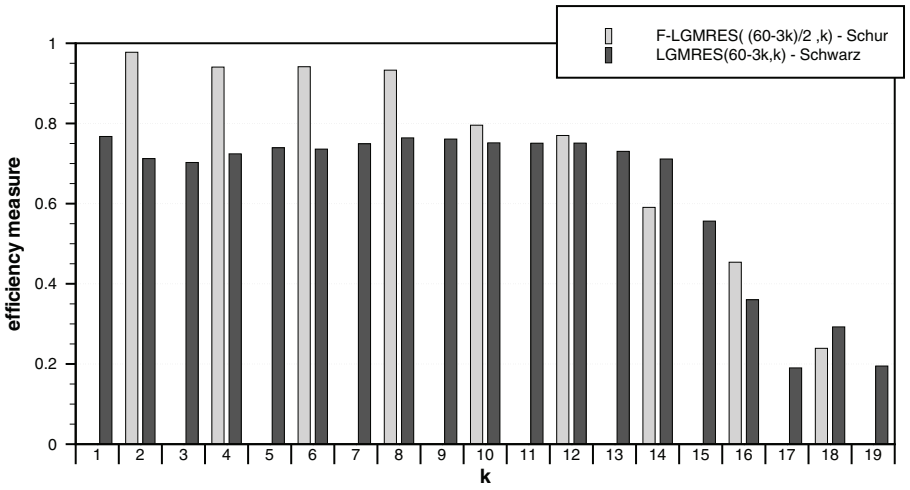


FIG. 4. *Efficiency measure, over a range of k values, for F-LGMRES((*$60-3k)/2, k$*) and LGMRES(*$60-3k$*,k) paired with the Schur- and Schwarz-based preconditioners, respectively.*

of k that work well for these solvers when applied to the adjoint problem. This study will inform our choice of k in the next section, where the performance of LGMRES and F-LGMRES is compared with the performance of other solvers.

We note two changes to the methodology of the previous GCROT study. First, the subspace-dependent memory is increased from 40 to 60, because F-LGMRES($(40-3k)/2, k$) was found to stall for Mach numbers $M_i \geq 0.45$, making it difficult to determine suitable values of k. Therefore, we compare LGMRES($60-3k, k$) with F-LGMRES($(60-3k)/2, k$): recall that the subspace-dependent memory of LGMRES(m, k) and F-LGMRES(m, k) is $m+3k$ and $2m+3k$, respectively. The second change we make is to consider only even values of k in F-LGMRES(m, k) to ensure that $m = (60-3k)/2$ is a whole number.

Figure 4 shows η_k for Schur-preconditioned F-LGMRES($(60-3k)/2, k$) and for Schwarz-preconditioned LGMRES($60-3k, k$). We see that standard LGMRES is approximately 20% less efficient than flexible LGMRES for $k \leq 8$, with the preconditioners considered. Both solvers consistently stall when $M_i > 0.6$ and $k \geq 14$. Consequently, we observe degraded efficiency for larger values of k in the figure.

In reference [1] lower values of k are recommended for LGMRES, and this is consistent with the performance of Schur-preconditioned F-LGMRES applied to the present adjoint problem. However, while $k = 2$ yields the best efficiency for flexible LGMRES, values $k \leq 8$ give similar performance. Schwarz-preconditioned LGMRES appears to be even less sensitive to the choice of k, provided $k \leq 14$. In general, given the present results and those of section 5.1, we can only conclude that the optimal value of k for advection-dominated problems likely falls in the broad range $[1, m]$.

The efficiency in Figure 4 is based on the best LGMRES or F-LGMRES run time, so we cannot use this figure and Figure 3 to compare the performance of GCROT and LGMRES. Such a comparison is included in the next section.

5.2.3. Solver comparison over a range of Mach numbers. Finally, we compare GCROT(m, k) with restarted GMRES, LGMRES(m, k), and BiCGStab over a range of Mach numbers. We pair each solver with the additive-Schwarz precon-

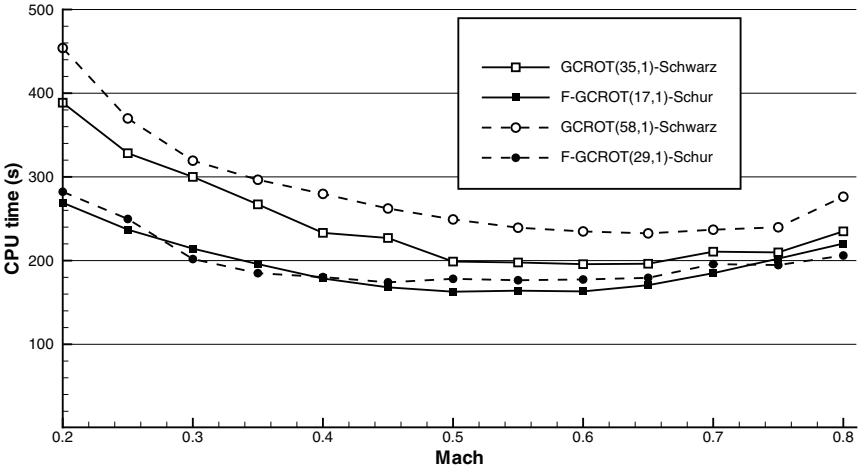


FIG. 5. *CPU time (in seconds) required by GCROT and F-GCROT to solve the adjoint problem for* $0.2 \leq M_i \leq 0.8$. *Both memory scenarios* ($s = 36$ *and* $s = 60$) *are plotted.*

ditioner, and the corresponding flexible variants with the approximate-Schur preconditioner. We consider two memory scenarios for the truncated subspace solvers: $s = 36$ and $s = 60$, where s is the size of the subspace used in restarted GMRES. To have the same subspace-dependent memory requirements as GMRES(s), we use FGMRES($s/2$), GCROT($s-2k, k$), F-GCROT($(s-2k)/2, k$), LGMRES($s-3k, k$), and F-LGMRES($(s-3k)/2, k$). Based on the results of sections 5.2.1 and 5.2.2, we adopt $k = 1$ for GCROT, F-GCROT, and LGMRES, and we use $k = 2$ for F-LGMRES. As in the previous two studies, we consider the set of Mach numbers defined by (5.2) and solve the adjoint problem to a relative tolerance of 10^{-8}.

We begin by comparing the performance of GCROT with that of F-GCROT, so that we can identify the better variant to compare with the remaining solvers. Although we expect Schur-preconditioned F-GCROT(m, k) to perform better—based on the results of section 5.2.1—we need to confirm this for the memory scenarios $s = 36$ and $s = 60$. Figure 5 plots the CPU time, in seconds, required by GCROT and F-GCROT to solve the adjoint problem for the selected set of Mach numbers. The two memory scenarios are plotted together. Note that F-GCROT(m, k) is relatively insensitive to the two subspace sizes; this observation is consistent with the results from the advection-diffusion problem. As expected, the F-GCROT(m, k)-Schur combination outperforms GCROT(m, k)-Schwarz in both memory scenarios, so we restrict our comparisons below to F-GCROT.

To compare the remaining solvers, we use a relative efficiency measure defined at each Mach number:

$$\eta_{\text{solver},i} = \frac{T_i^{\min}}{T_{\text{solver},i}}, \tag{5.3}$$

where $T_{\text{solver},i}$ is the CPU time required by the solver to fully converge the adjoint problem at the Mach number M_i, and $T_i^{\min}$ is the minimum time over all solver-preconditioner pairs at Mach number M_i. Recall that the solvers considered are GMRES(m), LGMRES(m, k), BiCGStab, GCROT(m, k), and their flexible variants. Cases that failed to converge in 4000 matrix-vector products are assigned a relative efficiency of zero.

The results for the $s = 36$ memory scenario are summarized in the plots of Figure 6. Figure 6(a) compares the efficiency of F-GCROT(17,1) with GMRES(36) and FGMRES(18). The most striking result is that FGMRES(18) stalls for all Mach numbers considered, yet F-GCROT(17,1) has relative efficiencies between 0.9 and 1. This dramatic improvement is achieved by simply keeping one residual update during a restart. F-GCROT(17,1) outperforms GMRES(36) using half the number of subspace vectors.

The efficiencies of LGMRES(30,2) and F-LGMRES(15,2) are plotted in Figure 6(b), together with that of F-GCROT(17,1). LGMRES was designed to accelerate the convergence of restarted GMRES and does not necessarily prevent stalling [1]. Therefore, it is not surprising that F-LGMRES(15,2) stalls for the full range of Mach numbers, since FGMRES(18) does the same. In the case of standard LGMRES(33,1), the slightly smaller subspace size may explain why it does not improve upon the performance of GMRES(36).

Figure 6(c) compares the performance of F-GCROT(17,1) with that of BiCGStab and FBiCGStab. As the figure shows, the Lanczos-based algorithms are the best choices for the adjoint problem at low Mach numbers ($M_i \leq 3.5$). Nevertheless, F-GCROT(17,1) remains competitive with these methods over this range, performing no worse than 10% slower. At the higher Mach numbers, F-GCROT(17,1) has the best performance and is typically more robust than BiCGStab and FBiCGStab. FBiCGStab fails to converge at $M = 0.65$ and $M = 0.75$ due to a breakdown in the algorithm ($\rho_i = 0$ in the terminology of [21]).

The results for $s = 60$ are shown in Figure 7. With an increased subspace size, restarted FGMRES performs well on all Mach numbers (Figure 7(a)). Nevertheless, F-GCROT(29,1) remains faster for all but three cases. As was the case for $s = 36$, the GMRES-Schwarz combination is not competitive with F-GCROT.

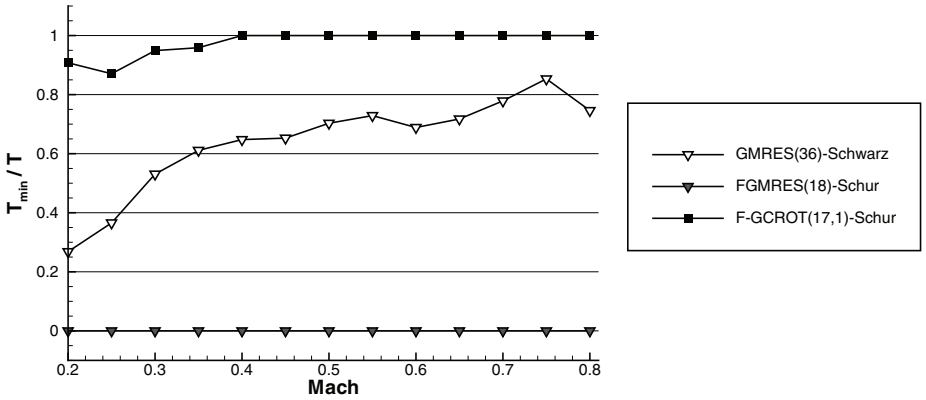
The performance of LGMRES and F-LGMRES improves when s is increased from 36 to 60; see Figure 7(b). Indeed, F-LGMRES(27,2) is competitive with F-GCROT(29,1) and has the lowest time for the Mach numbers $M = 5.5$ and $M = 6.0$. The efficiency curve of LGMRES(57,1) is similar to that of GMRES(60) for lower Mach numbers ($M < 0.45$) and is slightly better at larger Mach numbers.

For completeness, we include a comparison of F-GCROT(29,1) with BiCGStab and FBiCGStab in Figure 7(c). This figure is almost identical to Figure 6(c), because F-GCROT(29,1) and F-GCROT(17,1) behave similarly (see Figure 5).

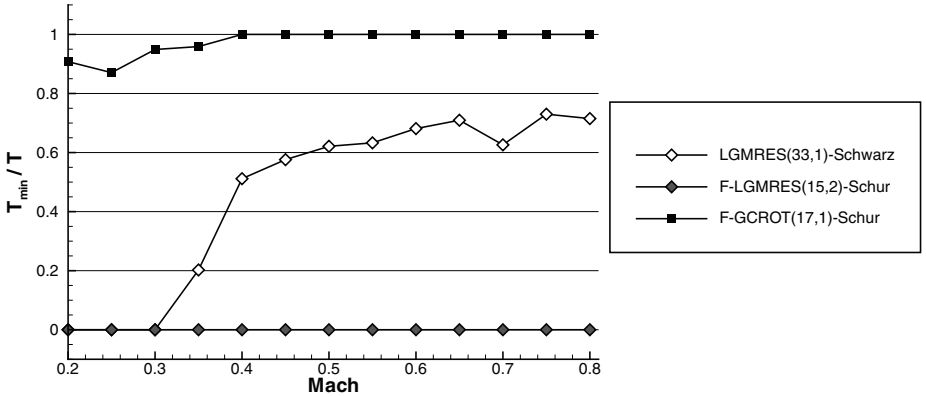
The evidence from this adjoint problem suggests the following conclusions for this class of problem:

- Schur-preconditioned F-GCROT is a good choice for a large range of Mach numbers ($M \in [0.2, 0.8]$) and is particularly well suited to larger Mach numbers ($M > 0.45$).
- For the same memory, Schur-preconditioned F-GCROT outperforms Schwarz-preconditioned GCROT.
- The performance of F-GCROT appears to be insensitive to the value of $s = 2m + 2k$ for the range $s \in [36, 60]$.
- Schwarz-preconditioned BiCGStab and Schur-preconditioned FBiCGStab are good choices for lower Mach numbers ($M < 0.45$). They are competitive for larger Mach numbers but are less robust.

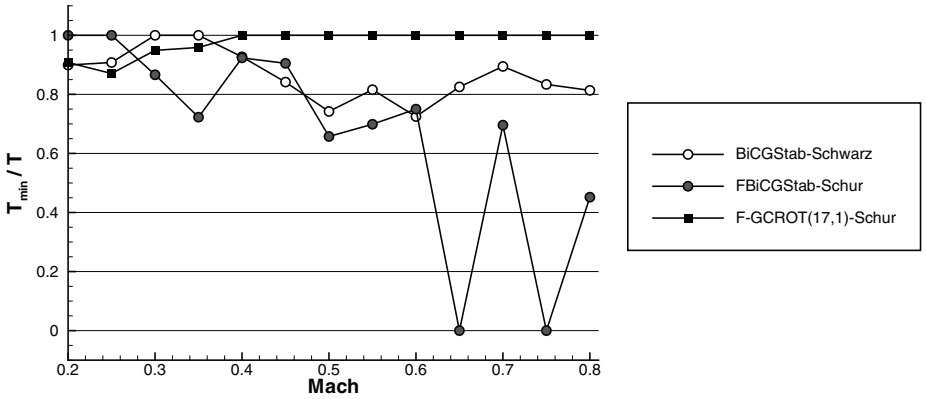
6. Conclusions. Singular-value analysis of the residual error indicates which subspaces of range (C_k) in GCRO were most important to convergence during the previous nested GMRES or FGMRES method. However, numerical experiments sug-



(a)



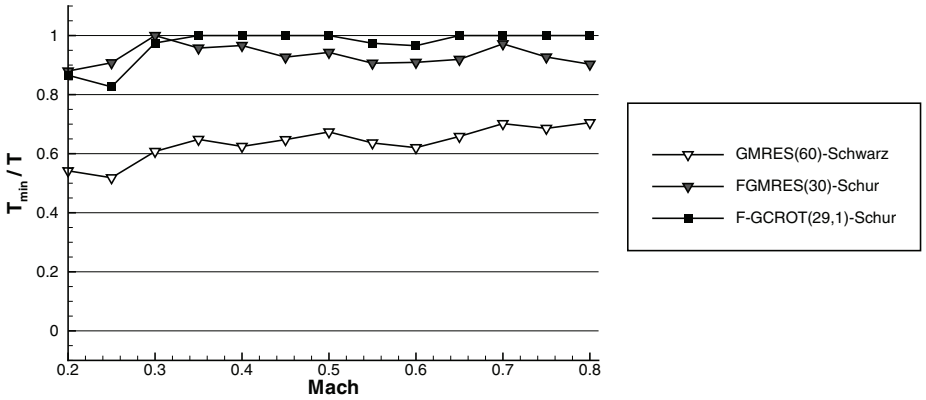
(b)



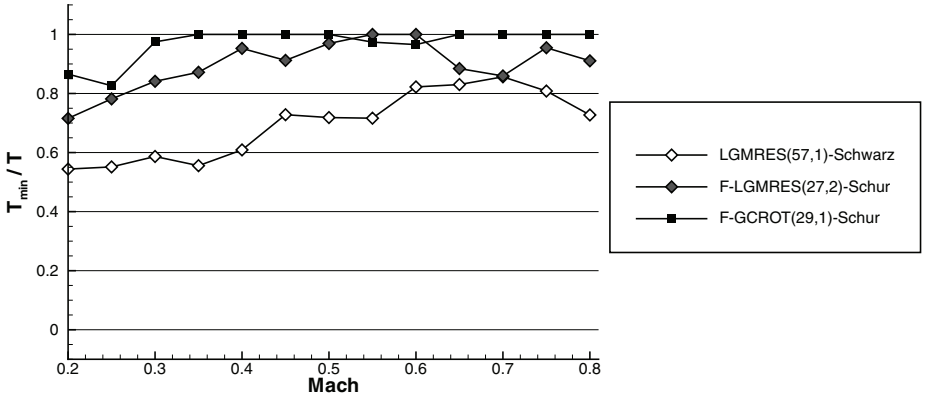
(c)

FIG. 6. *Relative efficiency of the solvers applied to the adjoint problem as a function of Mach number. Subspace-dependent memory is limited to* 36 *vectors. F-GCROT(*17,1*) is included in each figure to facilitate comparisons.*

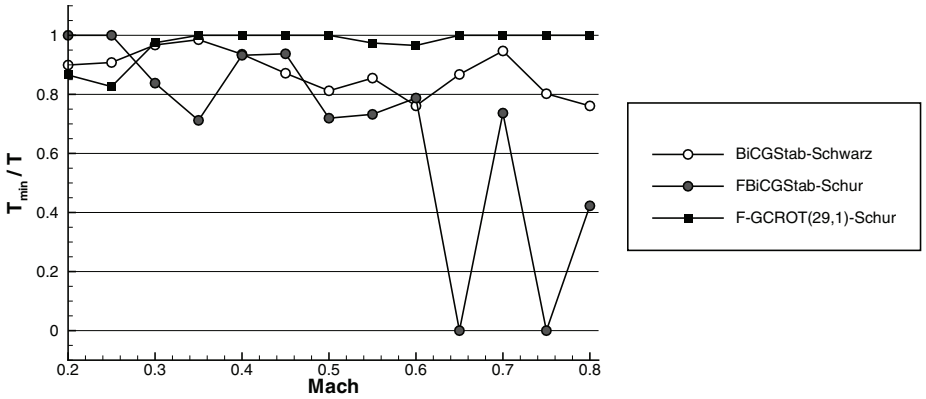
gest there is no correlation between those subspaces that contribute to convergence in a given iteration and those that will be useful in future iterations. This motivates a simplified truncation strategy for GCRO based on dropping the oldest vector from C_k, a strategy that does not require singular-value analysis. We refer to the resulting



(a)



(b)



(c)

FIG. 7. *Relative efficiency of the solvers applied to the adjoint problem as a function of Mach number. Subspace-dependent memory is limited to* 60 *vectors. F-GCROT(*29,1*) is included in each figure to facilitate comparisons.*

method as GCROT(m, k), where m is the size of the inner subspace and k is the size of the outer subspace.

Numerical experiments suggest that the parameter k in GCROT(m, k) should be chosen such that $k \leq m$. Although the optimal choice is problem dependent, the per-

formance throughout the range $k \leq m$ is typically good. The pair (m, k) determines the total number of vectors required and should be chosen based on available memory.

We have presented flexible variants of GCRO, GCROT, and LGMRES. By allowing the preconditioner to vary at each iteration, these variants have access to a large class of preconditioners. For fixed memory, F-GCROT(m, k) was shown to outperform its nonflexible implementation by taking advantage of a nonstationary preconditioner.

Based on the results presented, F-GCROT(m, k) is a promising option for the iterative solution of large-scale ($> 10^6$) nonsymmetric linear systems—particularly when such systems must be solved to a tight tolerance.

Appendix.

ALGORITHM 1. GCRO with FGMRES nested.

Data: x_0, m
Result: x

set $r_0 = b - Ax_0$
for $k = 0, 1, 2, \ldots$ **do**
  perform inner FGMRES method
  compute $\beta = \|r_k\|_2$ and $v_1 = r_k/\beta$
  for $j = 1, m$ **do**
    compute $z_j = M_j^{-1} v_j$
    compute $w = Az_j$
    orthogonalize w *against* C_k
    for $i = 1, k$ **do**
      $b_{i,j} = w^T c_i$
      $w := w - b_{i,j} c_i$
    end
    orthogonalize w *against* $V_{1:j}$
    for $i = 1, j$ **do**
      $h_{i,j} = w^T v_i$
      $w := w - h_{ij} v_i$
    end
    compute $h_{j+1,j} = \|w\|_2$ and $v_{j+1} = w/h_{j+1,j}$
  end
  define $Z_m := [z_1, z_2, \ldots, z_m]$
  compute $y_m = \operatorname{argmin}_y \|\beta e_1 - \bar{H}_m y\|_2$
  define new outer vectors
  $u_{k+1} = (Z_m - U_k B_m) y_m$
  $c_{k+1} = r_k - r_{\text{inner}} = V_{m+1} \bar{H}_m y_m$
  compute $\alpha = \|c_{k+1}\|_2$, $c_{k+1} := c_{k+1}/\alpha$, and $u_{k+1} := u_{k+1}/\alpha$
  update residual and solution
  $r_{k+1} := r_k - (c_{k+1}^T r_k) c_{k+1}$
  $x_{k+1} := x_k + (c_{k+1}^T r_k) u_{k+1}$
end

ALGORITHM 2. Flexible GCROT(m,k).

Data: x_0, m, k
Result: x

set $r_0 = b - Ax_0$, $C_k = 0$, and $U_k = 0$
for $l = 0, 1, 2, \ldots$ **do**
  perform inner FGMRES method
  compute $\beta = \|r_l\|_2$, $v_1 = r_l/\beta$, and $m_l = m + \max(k - l, 0)$
  for $j = 1, m_l$ **do**
    compute $z_j = M_j^{-1} v_j$
    compute $w = Az_j$
    orthogonalize w against C_k
    for $i = 1, \min(l, k)$ **do**
      $b_{i,j} = w^T c_i$
      $w := w - b_{i,j} c_i$
    end
    orthogonalize w against $V_{1:j}$
    for $i = 1, j$ **do**
      $h_{i,j} = w^T v_i$
      $w := w - h_{ij} v_i$
    end
    compute $h_{j+1,j} = \|w\|_2$ and $v_{j+1} = w/h_{j+1,j}$
  end
  define $Z_{m_l} := [z_1, z_2, \ldots, z_{m_l}]$
  compute $y_{m_l} = \text{argmin}_y \|\beta e_1 - \bar{H}_{m_l} y\|_2$
  define new outer vectors
  $\hat{u} = (Z_{m_l} - U_k B_{m_l}) y_{m_l}$
  $\hat{c} = r_l - r_{\text{inner}} = V_{m_l+1} \bar{H}_{m_l} y_{m_l}$
  compute $\alpha = \|\hat{c}\|_2$, $\hat{c} := \hat{c}/\alpha$, and $\hat{u} := \hat{u}/\alpha$
  update residual and solution
  $r_{l+1} := r_l - (\hat{c}^T r_l)\hat{c}$
  $x_{l+1} := x_l + (\hat{c}^T r_l)\hat{u}$
  if $(l > k)$ **then**
    discard the oldest vectors from C_k and U_k
    $C_k := C_k \begin{bmatrix} e_2 & e_3 & \cdots & e_k \end{bmatrix}$
    $U_k := U_k \begin{bmatrix} e_2 & e_3 & \cdots & e_k \end{bmatrix}$
  end
  $C_k := \begin{bmatrix} C_k & \hat{c} \end{bmatrix}$
  $U_k := \begin{bmatrix} U_k & \hat{u} \end{bmatrix}$
end

ALGORITHM 3. Flexible LGMRES(m,k).

Data: x_0, m, k
Result: x

set $r_0 = b - Ax_0$, $C_k = 0$, and $U_k = 0$
for $l = 0, 1, 2, \ldots$ **do**
  perform inner FGMRES method
  compute $\beta = \|r_l\|_2$, $v_1 = r_l/\beta$ and $m_l = m + \max(k - l, 0)$
  for $j = 1, m + k$ **do**
    if $j \leq m_l$ **then**
      compute $z_j = M_j^{-1} v_j$
      compute $w = Az_j$
    else
      $w = Au_{j-m_l} = c_{j-m_l}$
    end
    orthogonalize w against $V_{1:j}$
    for $i = 1, j$ **do**
      $h_{i,j} = w^T v_i$
      $w := w - h_{ij} v_i$
    end
    compute $h_{j+1,j} = \|w\|_2$ and $v_{j+1} = w/h_{j+1,j}$
  end
  define $Z_{m_l} := [z_1, z_2, \ldots, z_{m_l}]$
  compute $y_{m+k} = \text{argmin}_y \|\beta e_1 - \bar{H}_{m+k} y\|_2$
  define new outer vectors
  $\hat{u} = \begin{bmatrix} Z_{m_l} U_k \end{bmatrix} y_{m+k}$
  $\hat{c} = A\hat{u} = V_{m+k+1} \bar{H}_{m+k} y_{m+k}$
  update residual and solution
  $r_{l+1} := r_l - \hat{c}$
  $x_{l+1} := x_l + \hat{u}$
  if $(l > k)$ **then**
    discard the oldest vectors from C_k and U_k
    $C_k := C_k \begin{bmatrix} e_2 \; e_3 \; \cdots \; e_k \end{bmatrix}$
    $U_k := U_k \begin{bmatrix} e_2 \; e_3 \; \cdots \; e_k \end{bmatrix}$
  end
  $C_k := \begin{bmatrix} C_k & \hat{c} \end{bmatrix}$
  $U_k := \begin{bmatrix} U_k & \hat{u} \end{bmatrix}$
end

Acknowledgments. We sincerely thank the anonymous referees whose input and suggestions helped improve the manuscript. The editor's helpful comments were also greatly appreciated. We acknowledge the SciNet Consortium for providing the HPC resources used to obtain the results reported in section 5.2.

REFERENCES

[1] A. H. BAKER, E. R. JESSUP, AND T. MANTEUFFEL, *A technique for accelerating the convergence of restarted GMRES*, SIAM J. Matrix Anal. Appl., 26 (2005), pp. 962–984.
[2] R. F. BOISVERT, R. POZO, K. REMINGTON, R. BARRETT, AND J. DONGARRA, *The Matrix Market: A web resource for test matrix data*, in The Quality of Numerical Software, Assessment and Enhancement, R. F. Boisvert, ed., Chapman & Hall, London, 1997, pp. 125–137.

[3] E. de Sturler, *Nested Krylov methods based on GCR*, J. Comput. Appl. Math., 67 (1996), pp. 15–41.
[4] E. de Sturler, *Truncation strategies for optimal Krylov subspace methods*, SIAM J. Numer. Anal., 36 (1999), pp. 864–889.
[5] S. C. Eisenstat, H. C. Elman, and M. H. Schultz, *Variational iterative methods for non-symmetric systems of linear equations*, SIAM J. Numer. Anal., 20 (1983), pp. 345–357.
[6] W. D. Gropp, D. K. Kaushik, D. E. Keyes, and B. F. Smith, *High-performance parallel implicit CFD*, Parallel Comput., 27 (2001), pp. 337–362.
[7] M. H. Gutknecht, *Variants of BICGSTAB for matrices with complex spectrum*, SIAM J. Sci. Comput., 14 (1993), pp. 1020–1033.
[8] J. E. Hicken, *Efficient Algorithms for Future Aircraft Design: Contributions to Aerodynamic Shape Optimization*, Ph.D. thesis, University of Toronto, Toronto, 2009.
[9] J. E. Hicken and D. W. Zingg, *A parallel Newton-Krylov solver for the Euler equations discretized using simultaneous approximation terms*, AIAA J., 46 (2008), pp. 2773–2786.
[10] J. E. Hicken and D. W. Zingg, *Aerodynamic optimization algorithm with integrated geometry parameterization and mesh movement*, AIAA J., 48 (2010), pp. 400–413.
[11] D. Knoll and D. Keyes, *Jacobian-free Newton-Krylov methods: A survey of approaches and applications*, J. Comput. Phys., 193 (2004), pp. 357–397.
[12] *Matrix Market*, Technical report, National Institute of Standards and Technology, Gaithersburg, MD, 2009 (accessed 15 February 2009).
[13] J. A. Meijerink and H. A. van der Vorst, *An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix*, Math. Comp., 31 (1977), pp. 148–162.
[14] R. B. Morgan, *A restarted GMRES method augmented with eigenvectors*, SIAM J. Matrix Anal. Appl., 16 (1995), pp. 1154–1171.
[15] R. P. Pawlowski, J. N. Shadid, J. P. Simonis, and H. F. Walker, *Globalization techniques for Newton–Krylov methods and applications to the fully coupled solution of the Navier–Stokes equations*, SIAM Rev., 48 (2006), pp. 700–721.
[16] Y. Saad, *A flexible inner-outer preconditioned GMRES algorithm*, SIAM J. Sci. Comput., 14 (1993), pp. 461–469.
[17] Y. Saad and A. V. Malevsky, *P-SPARSLIB: A Portable Library of Distributed Memory Sparse Iterative Solvers*, Technical report UMSI-95-180, Minnesota Supercomputing Institute, University of Minnesota, Minneapolis, 1995.
[18] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.
[19] Y. Saad and M. Sosonkina, *Distributed Schur complement techniques for general sparse linear systems*, SIAM J. Sci. Comput., 21 (1999), pp. 1337–1356.
[20] V. Simoncini and D. B. Szyld, *Recent computational developments in Krylov subspace methods for linear systems*, Numer. Linear Algebra Appl., 14 (2007), pp. 1–59.
[21] 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. Statist. Comput., 13 (1992), pp. 631–644.
[22] H. A. van der Vorst and C. Vuik, *GMRESR: A family of nested GMRES methods*, Numer. Linear Algebra Appl., 1 (1994), pp. 369–386.
[23] J. A. Vogel, *Flexible BiCG and flexible Bi-CGSTAB for nonsymmetric linear systems*, Appl. Math. Comput., 188 (2007), pp. 226–233.