A Technique for Accelerating the Convergence of Restarted GMRES ; CU-CS-945-03

Baker

Elizabeth R. Jessup

Manteuffel

Follow this and additional works at: http://scholar.colorado.edu/csci_techreports

Recommended Citation


http://scholar.colorado.edu/csci_techreports/887
A TECHNIQUE FOR ACCELERATING THE CONVERGENCE OF RESTARTED GMRES

A. H. BAKER*, E. R. JESSUP†, AND T. MANTEUFFEL‡

Abstract. We have observed that the residual vectors at the end of each restart cycle of restarted GMRES often alternate direction in a cyclic fashion, thereby slowing convergence. We present a new technique for accelerating the convergence of restarted GMRES by disrupting this alternating pattern. The new algorithm resembles a full conjugate gradient method with polynomial preconditioning, and its implementation requires minimal changes to the standard restarted GMRES algorithm.

Key words. GMRES, iterative methods, Krylov subspace, restart, nonsymmetric linear systems

AMS subject classifications. 65F10

1. Introduction. Iterative methods are a common choice for solving the large sparse system of linear equations

\[ Ax = b, \]

where \( A \in \mathbb{R}^{n \times n} \) is nonsingular and \( x, b \in \mathbb{R}^n \). A popular class of iterative methods are Krylov subspace methods. Krylov methods find an approximate solution

\[ x_i \in x_0 + K_i(A, r_0), \]

where \( K_i(A, r_0) \equiv \text{span}\{r_0, Ar_0, \ldots, A^{i-1}r_0\} \) denotes an \( i \)-dimensional Krylov subspace, \( x_0 \) is the initial guess, and \( r_0 \) is the initial residual \( (r_0 = b - Ax_0) \). Krylov methods are also known as polynomial methods since equation (2) implies that the residual \( r_i \) can be written in terms of a polynomial of \( A \): \( r_i = p(A)r_0 \). The polynomial \( p(A) \) is typically referred to as the residual polynomial.

When \( A \) is symmetric, the Conjugate Gradient (CG) method [19] is popular, efficient, and well-understood. However, for nonsymmetric \( A \), the choice of algorithm is not as clear (e.g., see [25]), though the generalized minimum residual (GMRES) algorithm [30] is arguably the most popular method for nonsymmetric linear systems. GMRES is often referred to as an “optimal” Krylov method because the residual of the approximation found in the Krylov subspace has the smallest 2-norm [17]. In other words, GMRES selects \( x_i \in x_0 + K_i(A, r_0) \) such that \( \| b - Ax_0 + z \|_2 \) is a minimum over all \( z \in K_i(A, r_0) \). This minimum residual requirement is equivalent to the condition \( r_i \perp AK_i(A, r_0) \) (e.g., see [28]).

At each iteration of GMRES, the amount of storage and computational work required increases. Therefore, when the required resources make the standard GMRES algorithm impractical, the restarted version of the algorithm is used as suggested in [30]. In restarted GMRES (GMRES\((m)\)), the method is “restarted” once the Krylov

\*Department of Applied Mathematics, University of Colorado, Boulder, CO 80309-0526, (allison.baker@colorado.edu). The work of this author was supported by the Department of Energy through a Computational Science Graduate Fellowship (DE-FG02-97ER25308).

†Department of Computer Science, University of Colorado, Boulder, CO 80309-0430 (jessup@cs.colorado.edu). The work of this author was supported by the National Science Foundation under grant no. ACI-0072119.

‡Department of Applied Mathematics, University of Colorado, Boulder, CO 80309-0526, (tmanteuf@colorado.edu).
subspace reaches dimension $m$, and the current approximate solution becomes the
new initial guess for the next $m$ iterations. The restart parameter $m$ is generally
drawn small relative to $n$ to keep storage and computation requirements reasonable.
However, choosing an appropriate restart parameter can be difficult as the choice can
significantly affect the convergence rate (e.g., see [20, 14]).

In general, restarting slows the convergence of GMRES. When an iterative
approach is restarted, information (such as the previous approximation space) is dis-
carded at each restart. Therefore, a well-known drawback of GMRES($m$) is that
orthogonality to previously generated subspaces is not preserved at each restart. In
fact, GMRES($m$) can stall as a result. Stalling means that there is no decrease in the
residual norm at the end of a restart cycle. Restarting also negates the potential for
superlinear convergence behavior [32].

This paper is organized as follows. In Section 2, we describe some existing modi-
fications to GMRES($m$) aimed at accelerating convergence or overcoming stalling. We
introduce our new acceleration technique in Section 3. We present numerical results
in Section 4 and discuss the convergence behavior of the new algorithm in Section 5.
We close with concluding remarks in Section 6.

2. Background. In this section, we briefly describe some existing modifications
to the standard GMRES algorithm. These modifications all have the common goal of
enhancing the robustness of restarted GMRES. Two primary categories of modifi-
cation include hybrid iterative methods and acceleration techniques.

Hybrid iterative methods combine standard iterative methods in a variety of ways
to reduce the number of required vector operations. Many of these methods are essen-
tially modifications to GMRES($m$) aimed at improving its performance. Nachtigal,
et al. provide a general overview of this class of iterative methods in [24]. Hybrid
algorithms typically consist of 2 phases. Phase 1 consists of a method that does not
require prior information about the matrix $A$ but rather provides information on the
spectrum of $A$ upon completion (like GMRES). Phase 2 then typically consists of some
sort of parameter-dependent polynomial iteration method that is “cheaper” than the
method chosen for phase 1. Many varieties of hybrid algorithms use GMRES in phase
1; several of these are detailed in [24]. Though hybrid methods are often successful
at reducing the number of vector-vector products (as compared with a standard al-
gorithm), they typically increase the number of matrix-vector products required for
convergence. Therefore, the performance is dependent on the machine architecture
and the problem size as the number of times a large matrix is accessed from memory
substantially impacts performance.

Our work falls into the category of acceleration techniques. These techniques
attempt to mimic the convergence of full GMRES more closely or to accelerate the
convergence of GMRES($m$) by retaining some of the information that is typically
discarded at the time of restart. In [11], Eiermann, Ernst and Schneider present a
thorough overview and analysis of the most common acceleration techniques.

Augmented methods are a class of acceleration techniques. In particular, these
methods seek to avoid stalling by improving information in GMRES at the time of the
restart. Typically a (nearly) $A$-invariant subspace is appended to the Krylov ap-
proximation space, resulting in an “augmented Krylov subspace” [5]. The invariant
subspace of $A$ associated with the smallest eigenvalues is commonly used, as these
eigenvalues are thought to hinder convergence the most. Algorithms that include spec-
tral information at the restart to overcome stalling are presented by Morgan in both
[22] and [23] and are further discussed in [5] and [29]. These augmentation techniques
are more suitable for some types of problems than others. They can be very effective
when convergence is being hampered by a few eigenvalues [22]. However, they may
have little effect on highly non-normal problems [5], or solving the eigenvalue problem
may be too costly for the technique to be beneficial [22]. Of interest to us is the simple
framework provided for appending (non-Krylov) vectors to the approximation space.

Another class of acceleration techniques is based on the fact that ideally the
approximation space should contain the correction $c$ such that $x = x_0 + c$ is the
exact solution to the problem [11]. The nested Krylov method GMRESR (GMRES
Recursive) [33] is one such technique. In GMRESR, the outer Generalized Conjugate
Residual (GCR) method [12] invokes an inner iterative method (like GMRES for
example) at each step $i$ to approximate the solution to $Ac = r_i$, where $r_i$ is the
current residual at step $i$. The approximate solution to $Ac = r_i$ then becomes the
next direction for the outer approximation space. The goal of this method is to
obtain similar convergence to that of full GMRES with less computational cost under
certain conditions. Note that the FGMRES (Flexible GMRES) method [27] can also
be viewed as a method that approximates solutions to similar residual equations at
each step. In fact, both FGMRES and GCR provide a framework for using a GMRES-
like method with any approximation space.

Another related acceleration technique is the GCRO (GCR with inner orthogonal-
ization) method by de Sturler [7]. The aim of this method is twofold: to compensate
for the information that is lost due to restarting as well as to overcome some of
the stalling problems that GMRESR can experience in the inner iteration. GCRO
is a modification to GMRESR such that the inner iterative method maintains
orthogonality to the outer approximation space. Thus the approximation from the
inner iteration at step $i$ takes into account both the inner and outer approximation
spaces. See also [10] for more details on preserving orthogonality in the inner iter-
ation of a nested Krylov method. In most cases, both GCRO and GMRESR must
be truncated to keep storage costs reasonable. Therefore, in a subsequent paper, de
Sturler [9] presents a truncated version of GCRO, the GCROT (GCRO Truncated)
method. GCROT attempts to determine which subspace of the outer approximation
space should be retained for the best convergence of future iterations as well as if any
portion of the inner Krylov subspace should also be kept.

As Fokkema et al. point out in [16], “the distinction between preconditioning and
acceleration is not a clear one.” These acceleration techniques (GMRESR, GCRO,
and FGMRES) can also be viewed as methods with variable preconditioning (allowing
the preconditioner to change with each iteration step). We will show that our new
method can also be viewed in this way.

3. A new algorithm: LGMRES, In this section, we describe a new method
for accelerating GMRES($m$). We begin with observations about the convergence
behavior of GMRES($m$) that lead us to the new technique. We then present the
algorithm and its implementation. We describe the method in the context of dynamic
preconditioning and close by comparing LGMRES to existing methods.

We call the new algorithm LGMRES (“Loose” GMRES). In theory, this new al-
gorithm resembles a full conjugate gradient method with polynomial preconditioning,
and its implementation requires minimal changes to the standard GMRES($m$) algo-
rithm. We note that the new algorithm is similar in concept to both GMRESR [33]
and GCRO [7] in that it includes previous approximations to the error in the new
approximation space. While LGMRES is not mathematically equivalent to either al-
gorithm, it is most similar to GCRO. However, the implementations are quite different
(and that of LGMRES is simpler).

The idea for LGMRES came from a desire to prevent the alternating behavior of GMRES\((m)\) described in Section 3.1. The primary motivation for this new method is that the idea and implementation easily lend themselves to a block method for solving a single right-hand side system (e.g., see [2]).

3.1. Observations of GMRES\((m)\). As previously mentioned, GMRES\((m)\) does not maintain orthogonality between approximation spaces generated at successive restarts. As a result, slow convergence or even stalling can occur.

Consider restarted GMRES\((m)\) when solving problem (1). A cycle is a group of \(m\) iterations between successive restarts. In this discussion, the restart number is denoted with a subscript: \(r_i\) is the residual after \(i\) cycles or \(m \times i\) iterations. The residual at the end of cycle \(i + 1\) is a polynomial in \(A\) times the residual from the previous cycle: \(r_{i+1} = p_{m+1}^m(A)r_i\), where \(p_{m+1}^m(A)\) is the degree \(m\) residual polynomial. During each restart cycle, GMRES\((m)\) finds an approximate solution \(x_{i+1}\) where \(x_{i+1} \in x_i + K_m(A, r_i)\) and \(r_{i+1} \perp AK_m(A, r_i)\).

We have noticed a pattern in GMRES\((m)\) where the residual vectors point in nearly the same direction at the end of every other restart cycle. In other words, \(r_{i+2} \approx \alpha r_i\) and the algorithm essentially bounces back and forth between two directions. This alternating pattern typically emerges after several restarts and becomes more pronounced as the iteration proceeds. The pattern is most pronounced (most “exact”) for symmetric matrices, but is noticeable for many nonsymmetric matrices as well. Also note that when two residuals point in nearly the same direction, then the roots of the two corresponding residual polynomials are nearly the same. Therefore, the alternating pattern can also be discovered by looking at the roots of the residual polynomials at the end of each restart. For nonsymmetric matrices, alternating roots are often noticeable only in a subset of the roots (often those approximating the smaller magnitude eigenvalues).

At this point, a general proof of why or when alternating occurs remains elusive. However, for symmetric \(A\), illustrating the potential for alternating to occur in GMRES\((m)\) is straightforward. Consider two consecutive restart cycles for GMRES\((m)\) with symmetric \(A\):

Restart Cycle 1:
\[
\begin{align*}
\text{(a)} & \text{ generate } K_m(A, r_{i-1}) \\
\text{(b)} & \text{ find } r_i \text{ s.t. } r_i \perp AK_m(A, r_{i-1})
\end{align*}
\]

Restart Cycle 1+1:
\[
\begin{align*}
\text{(c)} & \text{ generate } K_m(A, r_i) \\
\text{(d)} & \text{ find } r_{i+1} \text{ s.t. } r_{i+1} \perp AK_m(A, r_i)
\end{align*}
\]

From step (b) above:

\[r_i \perp AK_m(A, r_{i-1}) \Rightarrow r_{i-1} \perp A^T K_m(A^T, r_i)\]

Therefore since \(A = A^T\),

\[r_{i-1} \perp AK_m(A, r_i)\]

Notice that both \(r_{i+1}\) and \(r_{i-1}\) are orthogonal to the same space \(AK_m(A, r_i)\). Therefore, it is possible for alternating to occur for symmetric matrices because there is no mechanism in GMRES\((m)\) to prevent it. In fact, our experiments using GMRES\((m)\) on well-conditioned symmetric problems (not recommended in practice)
resulted in iteration counts to converge that were typically much higher than anticipated. These high iteration counts are a direct result of very pronounced alternating residual patterns.

For the special case when the restart parameter is one less than the matrix order, we can show that alternating must occur for both symmetric and skew-symmetric problems. Consider the following lemma.

**Lemma 1** (Equivalent constraints). When $A \in \mathbb{R}^{n \times n}$ is symmetric or skew-symmetric, and $w$ and $y$ are arbitrary real vectors of length $n$, the requirement that $w \perp AK_m(A,y)$ is equivalent to the requirement that $w \perp A^T K_m(A^T; y)$.

With this easily proved lemma, the following theorem is straightforward.

**Theorem 2** (Alternating residuals). When $A \in \mathbb{R}^{n \times n}$ is symmetric (or skew-symmetric) and the restart parameter is one less than the matrix order ($m = n - 1$), GMRES($m$) produces a sequence of residual vectors at the end of each restart cycle such that $r_{i+2} = \alpha r_i$, $\alpha \leq 1$.

**Proof.** During restart cycle $i$, GMRES($m$) finds an approximate solution $x_i$ such that $r_i \perp AK_m(A, r_{i-1})$. Note that for any $A \in \mathbb{R}^{n \times n}$:

$$r_i \perp AK_m(A, r_{i-1}) \Rightarrow r_{i-1} \perp A^T K_m(A^T, r_i).$$

From lemma 1,

$$r_{i-1} \perp A^T K_m(A^T, r_i) \Rightarrow r_{i-1} \perp AK_m(A, r_i).$$

Let $W_m \equiv [w_1 w_2 \ldots w_m]$ be an orthonormal basis for $AK_m(A, r_i)$. Since $m = n - 1$, there exists a $w_m$ such that $W_m = [W_m w_m]$ is an orthonormal basis for $\mathbb{R}^{n \times n}$. Therefore, $r_{i-1} \perp AK_m(A, r_i)$ implies that $r_{i-1} = \alpha w_m$, where $\alpha$ is some scalar.

Next, at restart cycle $i + 1$, GMRES($m$) finds $x_{i+1}$ such that $r_{i+1} \perp AK_m(A, r_i)$. To satisfy the constraint $r_{i+1} \perp AK_m(A, r_i), r_{i+1}$ must also lie in the direction of $w_n$. Let $r_{i+1} = \beta w_n$, where $\beta$ is some scalar. Therefore, $r_{i+1}$ and $r_{i-1}$ point in the same direction ($w_n$), and $r_{i+1} = \frac{\beta}{\alpha} r_{i-1}$. Note that because the GMRES($m$) residual norm is non-increasing, $\frac{\beta}{\alpha} \leq 1$. \hfill \Box

As previously mentioned, alternating is not restricted to symmetric matrices only, and we have detected this (inexact) alternating pattern in many problems with nonsymmetric $A$ and $m << n$. In fact, as we discuss in Section 5, the presence of a detectable alternating pattern cannot be directly correlated to any scalar measurement of the nonsymmetry or non-normality of $A$. This lack of correlation is consistent with Embree’s observation that “all the nuances of non-normality can’t be abridged into a single number that consistently describes non-normal matrix behavior” [13].

This alternating phenomenon is simply a symptom of the lack of orthogonality between the approximation space generated during a particular restart cycle of GMRES($m$) and the approximation spaces from previous cycles. The inefficiency of alternating indicates that faster convergence should be possible if some degree of orthogonality to previous approximation spaces was maintained, a goal embraced by several acceleration techniques described in the previous section.

### 3.2. Idea and implementation of LGMRES.

Recall from Section 2 that augmented methods provide a simple framework for appending vectors to the Krylov approximation space. The idea is then to append vectors to the approximation space that in some sense represent approximation spaces from previous restart cycles. Recall that after restart cycle $i + 1$ we have that $r_{i+1} \perp AK_m(A, r_i)$. Ideally,
$r_{i+1} \perp AK_m(A, r_{i-1})$ also holds, but unfortunately GMRES$(m)$ discards $K_m(A, r_{i-1})$ entirely before beginning cycle $i + 1$.

Suppose that $\hat{x}$ is the true solution to problem (1). The error after the $i$-th restart cycle is denoted by $e_i$, where

$$e_i \equiv \hat{x} - x_i.$$  

We define

$$z_i \equiv x_i - x_{i-1}$$

as the approximation to the error at the $i$-th restart cycle. Because $z_i \in K_m(A, r_{i-1})$, this error approximation is a natural choice of vector with which to augment our next approximation space $K_m(A, r_i)$; it in some sense represents the space $K_m(A, r_{i-1})$ generated in the previous cycle and subsequently discarded. Furthermore, as explicitly pointed out in [11] and noted in Section 2, if our approximation space contains the exact correction $e_i$ such that $\hat{x} = x_i + e_i$, then we have solved the problem. Therefore, including an approximation to $e_i$ (such as $z_i$) in the approximation space is a reasonable strategy. Thus, the general idea of our method is like that of both GMRESR [33] and GCRO [7].

We denote our new restarted augmented GMRES algorithm by LGMRES$(m, k)$. LGMRES$(m, k)$ augments the standard Krylov approximation space with $k$ previous approximations to the error. Therefore at the end of restart cycle $i + 1$, LGMRES$(m, k)$ finds an approximate solution to (1) in the following way:

$$x_{i+1} = x_i + q_{i+1}^{m-1}(A)r_i + \sum_{j=i-k+1}^{i} \alpha_{ij}z_j,$$

where polynomial $q_{i+1}^{m-1}$ and $\alpha_{ij}$ are chosen such that $||r_{i+1}||_2$ is minimized.

Therefore, $r_{i+1} \perp AK_m(A, r_{i})$ still holds. In particular for $k = 1$, we also have that $r_{i+1} \perp AZ_i$. Since $z_i \in K_m(A, r_{i-1})$, $r_{i+2} \approx \alpha r_i$ is no longer possible. As a result, LGMRES does not exhibit alternating behavior. The convergence behavior of LGMRES is discussed in more detail in Section 5.

Note that in (9), $k = 0$ corresponds to standard GMRES$(m)$. Polynomial $q_i^{m-1}(x)$ is generally referred to as the iteration polynomial (degree $m - 1$) in standard GMRES$(m)$ (as opposed to the residual polynomial).

LGMRES$(m, k)$ is implemented as follows. At each restart cycle $(i)$ we generate the Krylov subspace $K_m(A, r_{i-1})$ and augment it with the $k$ most recent error approximations $z_j$, $j = (i - k + 1) : i$. The augmented approximation space $M = K_m(A, r_i) + \text{span}\{z_j\}_{j=(i-k+1):i}$ is size $m + k$. We then find the approximate solution from $M$ whose corresponding residual is a minimum in the Euclidean norm. This implementation requires minimal changes to the standard GMRES$(m)$ implementation and is quite similar to that of Morgan’s GMRES with eigenvectors (GMRES-E) method described in [22].

One restart cycle $(i)$ of the LGMRES$(m, k)$ algorithm is given in Figure 1. We denote the size of the augmented approximation space by $s$, where $s \equiv m + k$. Let $V_{s+1}$ be an $n \times (s + 1)$ orthonormal matrix whose columns are an orthonormal basis for $V = K_{s+1}(A, r_i) + \text{span}\{z_j\}_{j=(i-k+1):i}$. Let $W_s$ be the $n \times s$ matrix whose first $m$ columns are equivalent to the first $m$ columns of $V_{s+1}$. The last $k$ columns of $W$ are the $k$ error approximation vectors $z_j$, $j = (i - k + 1) : i$. In other words,
1. \( r_i = b - Ax_i, \beta = \| r_i \|_2, v_1 = r_i / \beta, s = m + k \)
2. for \( j = 1 : s 
\)
3. \( u = \begin{cases} 
A v_j \\
A z_{i-(j-m-1)} 
\end{cases} \) if \( j \leq m \)
4. otherwise
5. for \( l = 1 : j 
\)
6. \( h_{l,j} = \{ u, v_l \} \)
7. \( u = u - h_{l,j} v_l \)
8. \( h_{j+1,j} = \| u \|_2, v_{j+1} = u / h_{j+1,j} \)
9. end
10. end
11. \( W_s = [v_1, \ldots, v_m, z_i, \ldots, z_{i-k+1}], H_s = \{ h_{l,j} \}_{1 \leq l \leq j+1, 1 \leq j \leq s} \)
12. find \( y_h \) s.t. \( \| \beta e_1 - H_s y_h \|_2 \) is minimized
13. \( z_{i+1} = W_s y_h \) (also \( A z_{i+1} = V_{i+1} H_s y_h \))
14. \( x_{i+1} = x_i + z_{i+1} \)

**Figure 1. LGMRES(m, k) for restart cycle i.**

\[ V_{s+1} = [v_1, v_2, \ldots, v_m, \ldots, v_{m+k+1}] \text{ and } W_s = [v_1, \ldots, v_m, z_i, \ldots, z_{i-k+1}] \]. Then the relationship

\[ AW_s = V_{s+1} H_s \]

holds for LGMRES(m, k), where \( H_s \) denotes an \((s+1) \times s\) Hessenberg matrix whose elements \( h_{l,j} \) are defined in the algorithm in Figure 1. The relationship given in our equation (10) is analogous to equation (11) in [22] and (3) in [30]. In addition, although absent from the pseudo-code in Figure 1, we typically augment the Krylov space with normalized error approximations \((z_j / \| z_j \|_2)\) so that all columns of matrix \( W \) are of unit length.

When implementing LGMRES(m, k), only \( m \) matrix-vector multiplies are required per restart cycle, irrespective of the value of \( k \), provided that we form both \( z_i \) and \( A z_i \) at the end of cycle \( i \) as is done in the algorithm given in Figure 1. Note that forming \( A z_i \) does not require an explicit multiplication by \( A \) and that at most \( k \) pairs of \( z_j \) and \( A z_j \) need to be stored. Typically the number of vectors appended, \( k \), is much smaller than the restart parameter \( m \) (discussed in Section 4). The algorithm requires storage for the following vectors of length \( n \): \( m + k + 1 \) orthogonal basis vectors \((v_1, v_2, \ldots, v_{m+k+1})\), \( k \) pairs of \( z_j \) and \( A z_j \), the approximate solution, and the right-hand side. Therefore, this implementation of LGMRES(m, k) requires storage for \( m + 3k + 3 \) vectors of length \( n \) and matrix-vector multiplies per restart cycle. Recall that standard GMRES(m+k) requires storage for \( m + k + 3 \) vectors of length \( n \) and \( m + k \) matrix-vector multiplies per restart cycle (e.g., see [28]). One could reduce the storage requirement for LGMRES(m, k) by recomputing \( A z_i \) in each cycle. The storage requirement for vectors of length \( n \) would then drop to \( m + 2k + 3 \), but the number of matrix-vector multiplies required per cycle would increase to \( m + k \). We prefer the former method (as given in Figure 1) because it reduces the number of matrix-vector multiplies and is therefore generally faster.

LGMRES(m, k) can be preconditioned in a straightforward manner. Let \( M^{-1} \) denote the preconditioner. For left preconditioning, we simply precondition the initial residual in line 1 of the algorithm in Figure 1 \((r_i = M^{-1} b - M^{-1} A x_i \)). Then we replace \( A \) with \( M^{-1} A \) everywhere in lines 3 and 13. For right preconditioning, the required modifications are subtler. To include previous approximations to the error in the approximation space, we must now append \( \tilde{z}_j \equiv M(\tilde{x}_j - x_{j-1}) = Mz_j \) instead of \( z_j \).
1. $p_0 = C r_0$ ($s_0 = p_0$)
2. For $i = 0, 1, 2, \ldots$ (until convergence)
    3. $\alpha_i = \langle A p_i, A p_i \rangle$
    4. $x_{i+1} = x_i + \alpha_i p_i$
    5. $r_{i+1} = r_i - \alpha_i A p_i$
    6. $s_{i+1} = C r_{i+1}$
    7. $\beta_{ij} = \frac{\langle A s_{i+1}, A p_j \rangle}{\langle A p_j, A p_j \rangle}$ for $j = 0, 1, \ldots, i$
    8. $p_{i+1} = s_{i+1} + \sum_{j=0}^{i} \beta_{ij} p_j$

**Fig. 2. ORTHOMIN($A^* A$, $C$, $A$)**

to the standard Krylov subspace. Therefore, we replace $A$ with $AM^{-1}$ everywhere in lines 3 and 13 and $z$ with $\tilde{z}$ everywhere in lines 3, 11, and 13. While no explicit change is required for line 14 as given in Figure 1, note that, with right preconditioning, line 14 is equivalent to $x_{i+1} = x_i + M^{-1} \tilde{z}_{i+1}$.

### 3.3. LGMRES and polynomial preconditioning

Consider the “full” (i.e., non-truncated) version of LGMRES, denoted by LGMRES($m$), in which all previous error approximations are kept (i.e., $k = i$):

\[
x_{i+1} = x_i + q_{i+1}^{m-1} (A) r_i + \sum_{j=1}^{i} \alpha_{ij} z_j.
\]

Using (8) we can rewrite (11) as

\[
z_{i+1} = q_{i+1}^{m-1} (A) r_i + \sum_{j=1}^{i} \alpha_{ij} z_j
\]

We now explain the similarity between LGMRES($m$) and a full conjugate gradient (FCG) method with polynomial preconditioning.

The ORTHOMIN($A^* A$, $C$, $A$) algorithm is a minimal residual FCG method that is mathematically equivalent to full GMRES (e.g., see [28] or [1]). ORTHOMIN($A^* A$, $C$, $A$) minimizes $\|e_i\|_{A^* A}$ at each step, and the iterates are chosen from a sequence of nested Krylov subspaces. In this sense ORTHOMIN is a CG method [1]. In what follows, $C$ denotes the preconditioner, and when the method is not truncated, it is referred to as “full”. For context, note that the “classical” CG method of Hestenes and Stiefel [19] without preconditioning is mathematically equivalent to ORTHOMIN($A$, $I$, $A$): it minimizes $\|e_i\|_A$. For reference, ORTHOMIN($A^* A$, $C$, $A$) is given in Figure 2. Note that the direction vectors $p_i$ are $A^* A$-orthogonal and span the Krylov subspace $K_i(C A, s_0)$, where $s_0 = C r_0$.

Assume that ORTHOMIN uses a polynomial preconditioner $C = Q(A)$, where $Q$ is some polynomial in $A$. Then line 8 of the ORTHOMIN algorithm in Figure 2 can be rewritten as:

\[
p_{i+1} = Q(A) r_i + \sum_{j=0}^{i} \beta_{ij} p_j.
\]

Now the similarity between LGMRES($m$) as given in equation (12) to a polynomial-preconditioned FCG method as in equation (13) is readily apparent. Notice, however,
that LGMRES changes the polynomial preconditioner \( q_{i+1}^{m-1}(A) \) at each iteration \( i \), whereas preconditioned FCG uses a constant preconditioner that is not dependent on \( i \). In addition, we can also rewrite LGMRES\((m, k)\) (truncated LGMRES) as given in (9) in the same manner as (12):

\[
  z_{i+1} = q_{i+1}^{m-1}(A)r_i + \sum_{j=i-k+1}^{i} \alpha_{ij} z_j.
\]

Then we can categorize the LGMRES\((m, k)\) method as a truncated polynomial-preconditioned FCG method. The polynomial preconditioner is a GMRES\((m)\) iteration polynomial that varies with each step \( (i) \). Furthermore, we can prove the following theorem which further illustrates the resemblance between the two methods.

**Theorem 3** (Orthogonality of the error approximations). The error approximation vectors \( z_j \equiv x_j - x_{j-1} \) with which we augment the Krylov space in full LGMRES (11) or truncated LGMRES (9) are \( A^*A \)-orthogonal.

**Proof.** First, we define subspaces \( \mathcal{M}_{i+1} \) and \( \mathcal{M}_i \) as

\[
  \mathcal{M}_{i+1} \equiv K_m(A, r_i) + \text{span}\{z_j\}_{j=(i-k+1):i}
\]

and

\[
  \mathcal{M}_i \equiv K_m(A, r_{i-1}) + \text{span}\{z_j\}_{j=(i-k):(i-1)},
\]

respectively. Recall from equations (12) and (14) that LGMRES (full and truncated) finds a \( z_{i+1} \in \mathcal{M}_{i+1} \) such that \( \|r_{i+1}\|_2 \) is minimized. Minimizing \( \|r_{i+1}\|_2 \) is equivalent to minimizing \( \|e_{i+1}\|_{A^*A} \), where \( \|e_{i+1}\|_{A^*A} \equiv (A^*Ae_{i+1}, e_{i+1})^{1/2} \) and \( \langle \cdot, \cdot \rangle \) is the Euclidean inner product. Furthermore, minimizing the \( A^*A \) norm of the error \( e_{i+1} \) is equivalent to enforcing the following constraint (e.g., see [1]):

\[
  e_{i+1} \perp_{A^*A} \mathcal{M}_{i+1}.
\]

By recursion,

\[
  e_i \perp_{A^*A} \mathcal{M}_i.
\]

From (7) and (8),

\[
  z_{i+1} = e_i - e_{i+1}.
\]

As a result,

\[
  z_{i+1} \perp_{A^*A} \mathcal{M}_i \cap \mathcal{M}_{i+1}.
\]

Recalling that \( z_i \in \mathcal{M}_i \), note that \( \{z_j\}_{j=(i-k+1):i} \subseteq \mathcal{M}_i \cap \mathcal{M}_{i+1} \). Therefore,

\[
  z_{i+1} \perp_{A^*A} \{z_j\}_{j=(i-k+1):i}.
\]

In addition, the error at each step \( e_i \) of LGMRES\((m)\) can be written in terms of a polynomial times the initial error \( e_0 \). Subtracting both sides of (11) from the true solution \( \hat{x} \), we have

\[
  e_i = e_{i-1} - q_i^{m-1}(A)r_{i-1} - \sum_{j=1}^{i-1} \alpha_{ij} z_j.
\]
Using (19) together with \( Ae_{i-1} = r_{i-1} \), the above becomes

\[
e_i = e_{i-1} - q_i^{m-1}(A)e_{i-1} + \sum_{j=1}^{i-1} \alpha_{ij}(e_j - e_{j-1}).
\]

From (23), we can indeed write \( e_i \) as a polynomial in \( A \) times the initial error \( e_0 \):

\[
e_i = Q_i^d(A)e_0,
\]

where \( d_i = i \times m \) is the degree of \( Q_i \), \( d_0 \equiv 1 \), and \( Q_0(x) \equiv 1 \). In fact, a somewhat complicated recursion is possible for \( Q_i \):

\[
Q_i^d(\xi) = Q_{i-1}^{d_{i-1}}(\xi) - q_i^{m-1}(\xi)Q_{i-1}^{d_{i-1}} + \sum_{j=1}^{i-1}(Q_j^{d_j}(\xi) - Q_{j-1}^{d_{j-1}}(\xi)).
\]

When viewing LGMRES as a polynomial-preconditioned FCG method, the variance in the preconditioning polynomial from cycle to cycle is of interest. For example, suppose the polynomial remains nearly constant. In this case, LGMRES could calculate the polynomial once and re-apply it in subsequent cycles in the manner of the hybrid methods described in Section 2, avoiding the cost of extra matrix-vector multiplies. We solved several test problems with full LGMRES and examined the roots of the iteration polynomial, \( q_i^{m-1} \) in (11), over the course of the iteration to gauge the variance in the polynomials. The polynomials were far from constant, varying enough to preclude the possibility of constant preconditioning. However, distinct root patterns were evident in several problems. As an example, Figure 3 displays the first eight sets of roots from eight restart cycles of full LGMRES with \( m = 10 \) for the size \( n = 1104 \) problem sherman4 (available from the Matrix Market Collection [26]) with a random right-hand side and zero initial guess. The eigenvalues and field of values of \( A \) are also included.

![Figure 3: The first eight sets of roots of the iteration polynomial for LGMRES(10) on problem sherman4 (n = 1104). The field of values (FOV) and eigenvalues of A are also displayed.](image)

Although full LGMRES is interesting from a theoretical point of view, it is not a practical algorithm. Storing all past values of \( z_j \) \((j = 1 : i)\) requires an increasing amount of storage at each restart. As with GMRESR and GCRO, truncating is
necessary. Therefore, in practice, we use truncated LGMRES\((m, k)\) as given in (9) with some \(k < 1\). In Section 4, we show that optimal values for \(k\) are typically very small, \(k \leq 3\). Furthermore, note that the \(A^* A\)-orthogonality of the error approximation vectors shown in Theorem 3 is not exploited in the implementation of LGMRES given in the previous section. In fact, a total of \(k\) vector products and updates per restart cycle in the algorithm given in Figure 1 are extraneous due to a zero vector product in line 5. However, for small \(k\), the benefit of modifying the LGMRES\((m, k)\) implementation to exploit this orthogonality is negligible.

3.4. Comparison to existing methods. As previously stated, LGMRES\((m, k)\) acts as an accelerator for GMRES\((m)\). The algorithm is not designed to overcome stalling as the error approximation vectors are zero when the residual norm does not decrease within a cycle. Thus, while the LGMRES implementation mimics that of Morgan’s GMRES-E [22], we do not compare the two algorithms as GMRES-E is most effective for problems that stall due to the effects of a few eigenvalues. However, as noted at the beginning of this section, the general idea of LGMRES is very similar to that of GCRO [7]; both methods look for a solution in the approximation space consisting of previous approximations to the error as well as a Krylov space built on the current residual. The algorithms are not mathematically equivalent, and we briefly explain their similarities and differences in this section. First, the theoretical differences between (non-truncated) GCRO and full LGMRES are briefly described, followed by a comparison of the two truncated algorithms: GCROT and LGMRES\((m, k)\).

The nested Krylov method GCRO consists of an outer GCR method that invokes an inner GMRES method at each iteration to find an approximation to the error. Generally a fixed number of GMRES steps are taken at each inner iteration, say \(m\). GCR is a minimum residual method that maintains two bases: \(U_i\) and \(C_i = A U_i\), where \(C_i^T C_i = I_i\). Typically \(U_i\) is an \(A^* A\)-orthogonal basis for the Krylov space \(K_i(A, r_0)\). However, the implementation of GCR is such that \(U_i\) can actually contain any vectors (i.e., \(\text{range}(U_i) \neq K_i(A, r_0)\)) [7]. In particular, in the GCRO method, \(\text{range}(U_i)\) contains all of the previous approximations to the error from the inner GMRES method. The following minimization problem is solved at each inner iteration:

\[
\text{min} \| b - A x_{i+1} \|_2 \quad \text{s.t.} \quad x_{i+1} \in \text{range}(U_i) \oplus \text{range}(W_m),
\]

where \(W_m\) is an orthogonal basis for \(K_m(A_C, r_i)\) generated by the inner GMRES method and \(A_C \equiv (I - C_i C_i^T) A\). The Krylov space \(K_m(A_C, r_i)\) is a result of GCRO maintaining orthogonality against \(C_i\) from the beginning of the Arnoldi iteration, and \(W_{m+1}\) satisfies \(W_{m+1} \perp \text{range}(C_i)\). Thus when \(r_i\) is projected onto \(AW_m\) resulting in new residual \(r_{i+1}\), that new residual is also orthogonal to \(\text{range}(C_i)\) as desired. The solution to the global minimization problem of equation (26) is then found.

Similarly, full LGMRES finds a minimum residual solution in an approximation space consisting of all previous error approximations \((z_j)\) together with a Krylov space built off the current residual:

\[
\text{min} \| b - A x_{i+1} \|_2 \quad \text{s.t.} \quad x_{i+1} \in \text{range}(Z_i) \oplus \text{range}(V_m),
\]

where \(V_m\) is an orthogonal basis for \(K_m(A, r_i)\) and \(Z_i \equiv [z_1 \ldots z_i]\). In the case of LGMRES, the Arnoldi iteration does not maintain orthogonality against the previous error approximations. Instead, the error approximations are simply appended onto the generated Krylov subspace.
The difference in generation of the Krylov subspaces is a subtle difference between the two methods. Matrices $A_C$ and $A$ do not generate equivalent residual spaces $(A_C K_m(A_C, r_i))$ and $(A K_m (A, r_i))$, respectively. See [18] for more on matrices that generate equivalent Krylov residual spaces. Therefore, the residual projected onto these spaces is not the same unless the unlikely situation occurs where $\text{range}(V_m) \perp \text{range}(C_i)$. In later papers [8] and [9], de Sturler notes that appending vectors $C_i$ after the Arnoldi iteration such that $r_i$ is projected onto $\text{range}([A W_m | C_i])$ generally results in worse convergence behavior. However, we did not find this to be the general case for LGMRES.

The truncated derivative of GCRO, GCROT [9], truncates the outer approximation space by examining angles between subspaces and determining which subspaces (not vectors) are important for convergence. It is assumed that if a subspace was important for past convergence, then it will be important for future convergence and should be retained. The implementation of GCROT is fairly complicated and requires specification of six different parameters that affect the truncation.

We found that the naive truncation strategy that is very effective for LGMRES is not effective for GCRO (i.e., keeping the most recent $k$ vectors in $U_i$ and $C_i$). The ineffectiveness of a straightforward truncation for GCRO probably motivated de Sturler’s more complex truncation strategy in GCROT. We also tried a GCROT-like truncation strategy for LGMRES. In other words, we augmented the standard Krylov approximation space with the $k$ error approximations that lie at the smallest canonical angle with the new approximation space. This strategy was not as effective in our test problems as augmenting with the $k$ most recent error approximations.

The difference in effective truncation strategies highlights the difference between the two algorithms. For the ORTHOMIN algorithm, it has been observed that truncating the recursion such that only one (i.e., ORTHOMIN(1)) or even two (ORTHOMIN(2)) previous $p_j$ vectors are stored is quite effective when $A$ is nearly symmetric [34] (see line 8 in Figure 2). For example, when $A$ is Hermitian, then ORTHOMIN(1) is equivalent to the full (non-truncated) version of ORTHOMIN. Therefore, suppose $A$ is nearly symmetric, then the most recent direction vector ($p_i$) is the most important to retain as the new direction vector ($p_{i+1}$) should be nearly $(A^* A)$ orthogonal to the previous directions ($p_1, \ldots, p_{i-1}$). See [21] or [34], for example, for more details on when truncating is effective and on which conditions on $A$ make the truncated version equivalent to the full version. The relation of LGMRES to the ORTHOMIN($A^* A, C, A$) algorithm was explained in Section 3.3. The similarity between the two algorithms may explain why, for LGMRES, retaining only the most recent error approximations is an effective form of truncation. On the other hand, GCROT produces a set of error approximation vectors such that the most recently generated vectors are not necessarily the most important. For this reason, GCROT must calculate which subspace of the space spanned by the error approximation vectors to retain when truncating. In fact, we found that GCROT tends to perform better with a larger $k$ and a smaller $m$ (often $k > m$), whereas the reverse is true for LGMRES.

The difference in Krylov subspace generation and in truncation strategies both result in a more straightforward implementation for LGMRES($m, k$) than for GCROT. Furthermore, as described in [2], the implementation of LGMRES lends itself to a block method. In Section 4.2, we compare the performance of GCRO to LGMRES($m$), where all error approximations are retained, and GCROT to LGMRES($m, k$), where only the $k$ most recent approximations are used.
4. Experimental results. We demonstrate the potential of LGMRES by presenting experimental results from a variety of problems using implementations of LGMRES in both MATLAB and a locally modified version of PETSc (Argonne National Laboratory’s Portable, Extensible Toolkit for Scientific Computation) [3, 4]. We tested problems from various sources, including the Matrix Market Collection [26] and the University of Florida Sparse Matrix Collection [6]. In Sections 4.1 and 4.2, we compare the MATLAB script’s performance to MATLAB implementations of GMRES(m) and de Stuler’s GCRO [7] and GCROT [9] algorithms for problems without preconditioning. In Section 4.3, we demonstrate the usefulness of LGMRES for larger problems with preconditioning with a PETSc implementation of LGMRES.

4.1. Comparison to GMRES(m). In this section, we demonstrate that LGMRES can significantly accelerate the convergence of restarted GMRES. To compare the performance of GMRES(m, k) and GMRES(m), we implemented each in MATLAB. Our purpose with these implementations is to gauge the acceleration potential of LGMRES as well as its range of applicability. Therefore, in this section and Section 4.2, we do not use preconditioning for the MATLAB tests. We also allow iteration counts to be unrealistically large (much greater than n), terminating after a maximum of 30000 iterations if convergence is not reached. A zero initial guess is used for all problems.

We look at a test set of 20 problems, 17 from the Matrix Market Collection and 3 convection-diffusion (CD) problems. The Matrix Market problems include the following: add20, orsreg_., orsirr_., cavity05, nos3, e05r0000, cdde1, pde900, sherman1, sherman4, sherman5, watt.2, rdb11230, jpw.h991, fs_760.1, steam2, and cavity10. We randomly generate a right-hand side for each. The three CD problems are taken from [22] and are variations of the partial differential equation (PDE) $u_{xx} + u_{yy} + D u_x = -(41)^2$ with increasing degree of nonsymmetry: $D = 1$, $D = 41$, and $D = 41^2$, which we refer to as morgan.1, morgan.41, and morgan.1681, respectively. These PDEs are discretized by central finite differences on the unit square with zero boundary conditions and step-size $h = 1/41$. We stop the iteration when the relative residual norm is less than the convergence tolerance $\zeta$, i.e., when $\|r_0\|_2/\|r_0\|_2 \leq \zeta$. We use $\zeta = 10^{-9}$ for the CD problems, pde900, jpw.h991, and fs_760.1. For the remaining more slowly converging problems, we use $\zeta = 10^{-5}$. Several restart parameters (from $m = 10, 20, 30, \text{and } 40$) are chosen for each problem, depending on its size, resulting in a total of 60 test cases.

For each of these 60 test cases, we compare the performances of GMRES(m) and LGMRES with equal-sized approximation spaces. Figure 4 shows the number of matrix-vector multiplies required for convergence for GMRES(m) and LGMRES(m $-$ k, k) with $k = 1 : 5$. In both the top and bottom plots, the y-axis is the number of matrix-vector multiplies required for convergence by GMRES(m) divided by the number required by LGMRES(m $-$ k, k). Note that the log of this ratio is plotted on the y-axis of Figure 4. The x-axis corresponds to the 60 test cases, and the results have been sorted according to ratio size.

In the top panel of Figure 4, the result of the “best” LGMRES(m $-$ k, k) for $k = 1 : 5$ is compared to GMRES(m). The bars extending above the x-axis favor LGMRES(m $-$ k, k) (55 cases)—in these cases GMRES(m) requires more matrix-vector multiplies than does LGMRES(m $-$ k, k). The bars below the x-axis favor GMRES(m) (four cases: pde900 with $m = 20$ and $m = 30$, and morgan.41 with $m = 10$ and $m = 20$). The remaining case is equal for both methods. The plot in the bottom panel shows the variance in results for LGMRES(m $-$ k, k) with $k = 1 : 5$. 
Generally $k \leq 3$ is best for LGMRES($m - k, k$), and returns are diminishing for larger $k$, especially when $m$ is small.

The effectiveness of this new algorithm depends upon the matrix and the restart parameter $m$, but the savings in matrix-vector multiplies are quite substantial in many cases. LGMRES typically does not require more iterations than does restarted GMRES.

4.2. Comparison to GCRO and GCROT. In this section, we compare the performance of GCRO to full LGMRES (LGMRES($m$)) and GCROT to LGMRES($m, k$) in MATLAB.
4.2.1. Comparing LGMRES($m$) to GCRO. We evaluate MATLAB implementations of LGMRES($m$) and GCRO in the same manner as in Section 4.1. That is, we compare the number of matrix-vector multiplies required for the relative residual norm to be less than the convergence tolerance $\zeta = 10^{-5}$. For these non-truncated methods, we use small values of $m$, $m = 10$ and $m = 5$, since storage increases with each iteration. We again test the three related CD problems (morgan_1, morgan_4, and morgan_1681), as these problems were also used by de Sturler in [9], as well as a subset of the Matrix Market problems from the previous section (add20, orsreg_1, orsrr_1, cavity05, sherman1, sherman4). We also test one new Matrix Market problem, pores_3. We include pores_3, a problem that stalls for both GMRES(10) and GMRES(5), to test de Sturler’s observation that appending error approximations (as LGMRES does) is not effective for stalling [8].

Figure 5 compares the two methods. Similar to the plots in Figure 4, bars extending above the x-axis favor LGMRES($m$) (nine cases), and the bars below the x-axis favor GCRO($m$) (eleven cases). Note that the scale of the y-axis is much smaller than in Figure 4. The two cases for which GMRES($m$) stalls are problems 19 and 20, corresponding to pores_3. LGMRES($m$) requires fewer matrix-multiplies than GCRO when $m = 5$ (problem 20), demonstrating that, for a non-truncated method, appending error approximations can be effective for stalling. In fact, as Figure 5 illustrates, our experience does not clearly indicate which algorithm is to be preferred in a given situation.

4.2.2. Comparing LGMRES($m$, $k$) to GCROT. Now we compare the more practical truncated versions of the two algorithms: LMGRES($m$, $k$) and GCROT($m$, $k$, $k$, $s$, $p1$, $p2$). For each of these truncated algorithms, the size of the approximation space is $m+k$. We use a MATLAB implementation of GCROT supplied by Oliver Ernst. Test problems are the same as in Section 4.2.1 (Figure 5) with approximation spaces of

![Matrix–Vector Multiplies Required for Convergence](image-url)
size 10 and 20. However, since neither truncated algorithm converges for the pores.3 test cases, we replaced pores.3 with the larger sherman5 matrix \((n = 3312)\) and used approximation spaces of size 20 and 30 for that problem. Again, we monitor the relative residual norm and stop when the convergence tolerance is reached: \(\epsilon = 10^{-9}\) for the three CD problems and \(\epsilon = 10^{-5}\) for the seven Matrix Market problems.

For each of the 20 test cases, we ran LGMRES\((m, k)\) with \(k = 1 : 5\) and ten permutations of GCROT\((m, k, k, s, p1, p2)\), where \(m + k\) is constant. Figure 6 compares the two methods. The y-axis indicates the number of matrix-vector multiplies required for convergence. The x-axis corresponds to the 20 test cases. The bars indicate the range (minimum to maximum) of matrix-vector multiplies required. The circles represent restarted GMRES for each problem with the corresponding approximation space size. The circles and bars reaching the top of the y-axis indicate that convergence was not reached at that point. Again, some of these iteration counts are unrealistically large, but recall that we are not considering preconditioning and are simply evaluating the relative performance of the two algorithms. In eleven cases the LGMRES minimum is lower than the GCROT minimum, and in the remaining nine cases, the reverse is true. Of interest is that the range from minimum to maximum is typically much smaller for LGMRES than for GCROT. This demonstrates the sensitivity of GCROT to the input parameters. It is not clear how to choose the optimal parameters for GCROT (for example, whether \(m > k\) or \(m < k\) will be ideal for a particular problem). Ernst also found that choosing the parameters for GCROT can be problematic [15]. However, for LGMRES, \(k \leq 3\) is nearly always the best choice and the variation in results for \(k = 1 : 3\) is generally small. We also found that neither of these truncated algorithms is particularly useful for an approximation space of size \(m + k\) when GMRES\((m + k)\) stalls.
4.3. Effectiveness for larger preconditioned problems. In this section, we demonstrate that LGMRES can be a helpful addition to preconditioning for large problems. We implemented LGMRES in C using a locally modified version of PETSc [3, 4] in order to test larger problems and to obtain reliable timing results. A PETSc implementation also provides easy access to a variety of preconditioners. First, we look at cumulative results for 15 different matrix problems. Then we more closely examine a few of those problems.

We chose a variety of test problems from the Matrix Market Collection [26], the University of Florida (UF) Sparse Matrix Collection [6], and the PETSc [3, 4] collection of test matrices. We use the ILU(p) preconditioner, where p indicates the level of fill (e.g., see [28]). If a right-hand side is not provided, we generate a random right-hand side. For reference, the test problems are listed in Table 1.

### Table 1

<table>
<thead>
<tr>
<th>Problem</th>
<th>n</th>
<th>nnz</th>
<th>ILU(p)</th>
<th>Source</th>
<th>Application Area</th>
</tr>
</thead>
<tbody>
<tr>
<td>idapml1</td>
<td>22294</td>
<td>623554</td>
<td>ILU(0)</td>
<td>MM: Sparskit</td>
<td>fluid flow</td>
</tr>
<tr>
<td>memplus</td>
<td>17756</td>
<td>126550</td>
<td>ILU(0)</td>
<td>MM: Hamaa</td>
<td>digital circuit simulation</td>
</tr>
<tr>
<td>arco5</td>
<td>38194</td>
<td>24166</td>
<td>ILU(0)</td>
<td>PC</td>
<td>multiphase flow: oil reservoir</td>
</tr>
<tr>
<td>arco6</td>
<td>33388</td>
<td>15166</td>
<td>ILU(0)</td>
<td>PC</td>
<td>multiphase flow: oil reservoir</td>
</tr>
<tr>
<td>exaf7</td>
<td>10000</td>
<td>220237</td>
<td>ILU(0)</td>
<td>PC</td>
<td>multiphase flow: oil reservoir</td>
</tr>
<tr>
<td>garon2</td>
<td>7740</td>
<td>459012</td>
<td>ILU(0)</td>
<td>UF: FIDAP</td>
<td>fluid flow</td>
</tr>
<tr>
<td>bcircuit</td>
<td>1353</td>
<td>39000</td>
<td>ILU(1)</td>
<td>UF: Garon</td>
<td>fluid flow</td>
</tr>
<tr>
<td>xens01</td>
<td>48600</td>
<td>118120</td>
<td>ILU(2)</td>
<td>UF: Ronis</td>
<td>crystalline compound analysis</td>
</tr>
<tr>
<td>pisa</td>
<td>11738</td>
<td>79656</td>
<td>ILU(0)</td>
<td>UF: Gaertner</td>
<td>acoustic radiation</td>
</tr>
<tr>
<td>afio1</td>
<td>8302</td>
<td>123567</td>
<td>ILU(0)</td>
<td>UF: Okunbor</td>
<td>heat exchanger simulation</td>
</tr>
<tr>
<td>venkat50</td>
<td>62421</td>
<td>171792</td>
<td>ILU(0)</td>
<td>UF: Simon</td>
<td>fluid dynamics</td>
</tr>
<tr>
<td>eph3</td>
<td>84617</td>
<td>463625</td>
<td>ILU(0)</td>
<td>UF: Averous</td>
<td>heat exchanger simulation</td>
</tr>
<tr>
<td>big</td>
<td>13009</td>
<td>91465</td>
<td>ILU(1)</td>
<td>UF: Zhao</td>
<td>electromagnetic systems</td>
</tr>
<tr>
<td>zhao2</td>
<td>33861</td>
<td>166953</td>
<td>ILU(0)</td>
<td>UF: Zhao</td>
<td>electromagnetic systems</td>
</tr>
</tbody>
</table>

We compare the performance of restarted GMRES to that of LGMRES(m, k) with the same approximation space size and then the same storage requirements. For LGMRES, we report results for k = 1 : 3, as we find that choosing k in this range typically results in the most improvement with the least risk of increasing execution time. All tests are run until the relative residual norm is less than the convergence tolerance ζ = 10^{-9} or a maximum of 30000 iterations is reached. In the latter case we say that the method does not converge. Recall that GMRES with left preconditioning minimizes the preconditioned residual norm (||M^{-1}r||_2)), and, therefore, the determination of convergence is based on this preconditioned residual norm as usual. The initial guess is a zero vector in all cases. Unless otherwise noted, results provided were run on a Sun UltraSPARC 10 with 256M RAM, a clock-rate of 360 MHz, a 16KB L1 cache, and a 2MB L2 cache. For each problem we report wall clock time for the linear solve. All timings are averages from five runs and have standard deviations of at most two percent, although most are less than one percent. If a method does not converge in 30000 iterations, the execution time reported reflects the time to 30000 iterations. We did not compare LGMRES(m, k) to GCROT for these larger preconditioned problems because no PETSc implementation of GCROT
is available.

In Figure 7, we compare GMRES(30) to LGMRES(29, 1), LGMRES(28, 2), and LGMRES(27, 3). All four of these methods generate an approximation space of dimension 30 during each restart cycle. Similar to the plots seen previously, the y-axis indicates the log of the ratio of the time to converge for GMRES(30) to the time to converge for both the best and worst performing case of LGMRES(30 − k, k) for k = 1 : 3, and the x-axis corresponds to the 15 test problems. Points above the x-axis favor LGMRES and points below favor GMRES. The x-axis corresponds to the test problems in the order they appear in Table 1. The best case LGMRES(30 − k, k) outperforms GMRES(30) for all problems. The worst case LGMRES(30 − k, k) outperforms GMRES(30) in all cases except for problem 12 (venkat50). In fact, since problems pesa, big, and zhao2 do not converge in 30000 GMRES(m) iterations, the gains of LGMRES(30 − k, k) over GMRES(30) are understated in this figure as LGMRES(30 − k, k) converges for all three.

For larger problems in particular, comparing restarted GMRES to an LGMRES method that requires an equal amount of storage is also of interest. Recall from Section 3.2 that GMRES(m) and LGMRES(m, k), for the implementation we have chosen, require storage for m + 3 and m + 3k + 3 vectors of length n, respectively. Therefore, both GMRES(30) and LGMRES(30 − 3k, k) have the same 33 vector storage requirement. Similar to the previous figure, Figure 8 compares GMRES(30) to LGMRES(27, 1), LGMRES(24, 2), and LGMRES(21, 3). In this comparison, one augmentation vector must be more helpful than three standard Krylov vectors for LGMRES to win. This requirement is fairly stringent for some of the larger problems given that we allow only 33 vectors of storage. Nevertheless, the majority of the problems still show improvement with LGMRES. Problems for which LGMRES is not helpful in this comparison include problem 5 (arco6), problem 7 (garon2), and again the worst case only for problem 12 (venkat50).

Now we examine problems bircircuit, fidapm11, and big from our test set in Table
1 in more detail, additionally providing timing results for full GMRES. These three
problems demonstrate different possible relations in convergence behavior between
LGMRES(\(m, k\)), GMRES(\(m\)) and full GMRES. In Section 5, we discuss the conver-
gence behavior of the LGMRES algorithm as well as the type of problems that benefit
the most (and least) when using LGMRES.

First consider the timing results from problem bcircuit. For this problem, full
GMRES requires memory resources beyond the physical memory limit of our machine.
For this reason, we had to re-run the bcircuit problem on a similar machine with four
times as much memory (a Sun UltraSPARC 10 with 1024M RAM, a clock-rate of
440 MHz, a 16KB L1 cache, and a 2MB L2 cache) to obtain timing results for full
GMRES. Therefore, for Table 2 only, all results presented for bcircuit were obtained
on this second machine. Even with the extra memory provided by the second machine,
we see that restarted GMRES(30) is more than twice as fast as full GMRES, and
LGMRES is even faster. Because every machine has a limit as to the size problems it
can reasonably solve with a full method, restarted methods and acceleration methods
provide a great advantage.

Consider the timings for problem fidapm11 given in Table 3. For this problem,
full GMRES is faster than GMRES(30) on our machine. However, LGMRES has
faster execution times than both full GMRES and GMRES(\(m\)) in all cases (comparing
equal-sized approximation spaces or equal storage requirements).

Finally, results for problem big are given in Table 4. This problem is interesting
because GMRES(30) converges very slowly. In fact, the relative residual norm is still
\(\approx .002\) after 30000 iterations. Both LGMRES(30 − 3\(k\), \(k\)) and LGMRES(30 − \(k\),
\(k\)), on the other hand, improve convergence dramatically over that of GMRES(30).
However, for this moderately sized problem, full GMRES requires only 188 iterations
and wins by a landslide.

Most of the problems presented here required a restarted method given the re-
sources of the machine chosen for the experiments. On a more powerful machine (more memory and faster processor), full GMRES might be faster for many of these problems.

Because LGMRES is an accelerator, it is not, in general, a substitute for an effective preconditioner. However, we did encounter a number of test problems for which the ILU preconditioner is not a viable option and LGMRES is able to accelerate the poor convergence of non-preconditioned GMRES(m) dramatically. However, we expect that in those cases an appropriate preconditioner would be even more effective. Nevertheless, LGMRES can be an effective addition to preconditioning for a range of larger problems. Although LGMRES improvements with preconditioning tend not to be as spectacular as the improvements seen for the non-preconditioned problems of Section 4.1, even moderate acceleration for large problems can translate into significant time savings.

5. The behavior of LGMRES. In this section, we discuss the convergence behavior of the LGMRES algorithm and further comment on the results presented in the previous section. First we discuss observed properties of the convergence of LGMRES. In particular, we compare its convergence to that observed for GMRES(m) and discussed in Section 3.1. Next we look at the effect of LGMRES on a simple convection-diffusion problem when the degree of nonsymmetry is varied. Finally we consider the possibility of an adaptive version of the algorithm. In what follows, we do not consider problems for which GMRES(m) stalls.

### Table 2

Matrix bicircuit and its corresponding right-hand side, with \( n = 68902 \) and \( \text{nnz} = 375558 \). Preconditioned with \( \text{LU}(1) \). Times are in seconds and include mean and standard deviations of times for five runs.

<table>
<thead>
<tr>
<th>Method</th>
<th>Approx. space dimension</th>
<th># vectors stored</th>
<th>Matrix-vector multiplies</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full GMRES</td>
<td>1013</td>
<td>1016</td>
<td>1013</td>
<td>2880.36 ± 9.24</td>
</tr>
<tr>
<td>GMRES(30)</td>
<td>30</td>
<td>33</td>
<td>2602</td>
<td>1135.38 ± 12.58</td>
</tr>
<tr>
<td>LGMRES(29,1)</td>
<td>30</td>
<td>35</td>
<td>2959</td>
<td>615.38 ± 5.61</td>
</tr>
<tr>
<td>LGMRES(28,2)</td>
<td>30</td>
<td>37</td>
<td>1739</td>
<td>365.16 ± 2.47</td>
</tr>
<tr>
<td>LGMRES(27,3)</td>
<td>30</td>
<td>39</td>
<td>1707</td>
<td>369.70 ± 2.48</td>
</tr>
<tr>
<td>LGMRES(27,1)</td>
<td>28</td>
<td>33</td>
<td>2631</td>
<td>533.42 ± 4.54</td>
</tr>
<tr>
<td>LGMRES(24,2)</td>
<td>26</td>
<td>33</td>
<td>2467</td>
<td>503.71 ± 3.54</td>
</tr>
<tr>
<td>LGMRES(21,3)</td>
<td>24</td>
<td>33</td>
<td>1672</td>
<td>339.42 ± 2.21</td>
</tr>
</tbody>
</table>

### Table 3

Matrix fidapm11 and its corresponding right-hand side, with \( n = 22294 \), \( \text{nnz} = 623554 \), and \( \text{ILU}(0) \) preconditioning. Times are in seconds and include mean and standard deviations of times for five runs.

<table>
<thead>
<tr>
<th>Method</th>
<th>Approx. space dimension</th>
<th># vectors stored</th>
<th>Matrix-vector multiplies</th>
<th>Execution time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full GMRES</td>
<td>952</td>
<td>955</td>
<td>952</td>
<td>854.02 ± 6.27</td>
</tr>
<tr>
<td>GMRES(30)</td>
<td>30</td>
<td>33</td>
<td>16482</td>
<td>2100.23 ± 8.40</td>
</tr>
<tr>
<td>LGMRES(29,1)</td>
<td>30</td>
<td>35</td>
<td>5511</td>
<td>704.65 ± 0.18</td>
</tr>
<tr>
<td>LGMRES(28,2)</td>
<td>30</td>
<td>37</td>
<td>2915</td>
<td>376.64 ± 0.10</td>
</tr>
<tr>
<td>LGMRES(27,3)</td>
<td>30</td>
<td>39</td>
<td>2733</td>
<td>377.19 ± 0.78</td>
</tr>
<tr>
<td>LGMRES(27,1)</td>
<td>28</td>
<td>33</td>
<td>2939</td>
<td>664.84 ± 1.89</td>
</tr>
<tr>
<td>LGMRES(24,2)</td>
<td>26</td>
<td>33</td>
<td>3399</td>
<td>431.39 ± 1.00</td>
</tr>
<tr>
<td>LGMRES(21,3)</td>
<td>24</td>
<td>33</td>
<td>2911</td>
<td>373.96 ± 0.41</td>
</tr>
</tbody>
</table>
5.1. Experimental observations. Consider the GMRES\((m)\) approximation spaces \(S_i, S_{i+1}\) and \(S_{i+2}\), generated during restart cycles \(i, i+1\), and \(i+2\) respectively. For most problems, we find that consecutive approximation spaces, such as \(S_i\) and \(S_{i+1}\), are often reasonably orthogonal to each other, whereas \(S_i\) and \(S_{i+2}\) are not. In fact, the angle between every other approximation space, i.e., between \(S_i\) and \(S_{i+2}\), can be very small. This trend holds in particular for matrices that are nearly symmetric in some sense and results in the alternating behavior described in Section 3.1.

For standard restarted GMRES, we show that the angle between two residuals from consecutive restart cycles can be expressed in terms of a ratio of their residual norms. The following simple theorem is mathematically equivalent to the result given by Simoncini as Proposition 4.1 in [31].

**Theorem 4 (Angle between consecutive residual vectors).** Let \(r_{i+1}\) and \(r_i\) be the residuals from GMRES restart cycles \(i+1\) and \(i\), respectively. Then the angle between these residuals is given by

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

**Proof.** In restart cycle \(i + 1\) of GMRES\((m)\), \(x_{i+1} = x_i + \delta_{i+1}\), where \(\delta_{i+1} \in K_m(A, r_i)\). Therefore, the corresponding residual is

\[
r_{i+1} = r_i - A\delta_{i+1}.
\]

By construction,

\[
\langle r_{i+1}, A\delta_{i+1} \rangle = 0.
\]

Substituting (29) into (30):

\[
\langle r_{i+1}, r_i \rangle = \langle r_{i+1}, r_{i+1} \rangle.
\]

By definition,

\[
\cos \angle(r_{i+1}, r_i) = \frac{\langle r_{i+1}, r_i \rangle}{||r_{i+1}||_2||r_i||_2}.
\]

Substituting (31) into (32) completes the proof. \(\square\)
The above indicates that, for GMRES(m), the convergence rate correlates to the size of the angles between consecutive residual vectors. We refer to these angles as sequential angles. If consecutive residual vectors are nearly orthogonal to each other, then convergence is fast. (If we find an \( r_{i+1} \) such that \( r_{i+1} \perp r_i \), then we have found the exact solution.) Note that this result also holds for LGMRES. Now we consider the angle between every other residual.

**Theorem 5 (Angle between every other residual vector).** Let \( r_{i+1} \) and \( r_{i-1} \) be the residuals from GMRES restart cycles \( i+1 \) and \( i-1 \), respectively. Then the angle between these residuals is given by

\[
\cos \angle(r_{i+1}, r_{i-1}) = \frac{||r_{i+1}||^2}{||r_{i-1}||^2} - \frac{\langle A\delta_{i+1}, A\delta_{i-1} \rangle}{||r_{i+1}||^2||r_{i-1}||^2},
\]

where \( r_{i+1} = r_i - A\delta_{i+1} \) and \( r_{i} = r_{i-1} - A\delta_{i} \).

**Proof.** As in the previous proof, it is easily shown that

\[
\langle r_{i+1}, r_{i-1} \rangle = \langle r_{i+1}, r_{i+1} \rangle - \langle A\delta_{i+1}, A\delta_{i} \rangle.
\]

The proof follows directly from (34).

In terms of describing convergence, the above result is not immediately helpful. However, we will discuss a few of its implications after giving a corresponding result for LGMRES. Recall from Section 3.2 that GMRES(m, k) appends \( k \) previous approximations to the error to the current Krylov approximation space. Therefore if \( k = 1 \), then \( r_{i+1} \perp AK_m(A, r_i) \) and \( r_{i+1} \perp A\delta_i \) at the end of restart cycle \( i+1 \). By (8),

\[
A\delta_i = r_{i-1} - r_i.
\]

Therefore,

\[
\langle r_{i+1}, r_{i-1} - r_i \rangle = 0
\]

after \( i+1 \) GMRES cycles, and we can prove the following theorem.

**Theorem 6 (LGMRES: Every other residual vector).** Let \( r_{i+1} \) and \( r_{i-1} \) be the residuals from LGMRES restart cycles \( i+1 \) and \( i-1 \), respectively. Then the angle between these residuals is given by

\[
\cos \angle(r_{i+1}, r_{i-1}) = \frac{||r_{i+1}||^2}{||r_{i-1}||^2}.
\]

**Proof.** This theorem directly follows from Theorems 5 and 3 (noting the correlation between \( \delta_i \) in GMRES(m) and \( \delta_i \) in LGMRES). Alternatively, from (36) and (31):

\[
\langle r_{i+1}, r_{i-1} \rangle = \langle r_{i+1}, r_i \rangle = \langle r_{i+1}, r_{i+1} \rangle.
\]

The above, combined with the definition of cosine, completes the proof.

This result indicates that, for LGMRES, the progress of the iteration also correlates with the angles between every other residual vector. We refer to these angles as skip angles. Therefore, fast convergence implies large skip angles. More generally, for any \( 0 \leq j \leq k \) and \( i \geq k \), we can show for LGMRES(m, k) that

\[
\cos \angle(r_{i+1}, r_{i-j}) = \frac{||r_{i+1}||^2}{||r_{i-j}||^2}.
\]
Experimentally, we observe that LGMRES nearly always has a larger average skip angle than does GMRES($m$). When a problem exhibits signs of alternating residuals with GMRES($m$), then the angle between $r_{i-1}$ and $r_{i+1}$ is small. In this case, since $A_\delta_{i+1} = r_i - r_{i+1}$ and $A_\delta_i = r_{i-1} - r_i$, then the term $\langle A_\delta_{i+1}, A_\delta_i \rangle$ in Theorem 5 is negative. This result is observed experimentally and can be seen pictorially in Figure 9. Since LGMRES appends a previous error approximation to the approximation space during cycle $i + 1$, the term $\langle A_\delta_{i}, A_\delta_{i-1} \rangle$ is equal to zero by construction, as verified by Theorem 6 and discussed previously in Section 3.3 (see Theorem 3). This LGMRES augmenting scheme tends to increase the skip angle over that of GMRES($m$) and prevents the alternating behavior often observed in restarted GMRES.

![Diagrams](A) and (B)

**Fig. 9.** Two cases with alternating residual vectors: $r_1$ and $r_3$ point in nearly the same direction. $\langle A_{\delta_3}, A_{\delta_2} \rangle < 0$ in both (A) and (B).

Therefore, as compared to standard GMRES($m$), LGMRES($m, k$) does not necessarily improve orthogonality between two consecutive approximation spaces $S_i$ and $S_{i+1}$, but it does typically improve orthogonality quite significantly between the current approximation space and the space generated two restart cycles ago: $S_i$ and $S_{i-2}$. This action accelerates the convergence over that of GMRES($m$) in many cases. Recall from Theorem 4 that the size of the sequential angles is directly related to the reduction in residual at each cycle. Therefore, if increasing the skip angles occurs at the expense of reducing the average sequential angle, then LGMRES augmenting slows convergence. The method that “wins” generally has large skip angles and large sequential angles.

We find that the LGMRES augmenting scheme significantly improves GMRES($m$) convergence under the following conditions: GMRES($m$) skip angles are small and continue to decrease as the iteration progresses; GMRES($m$) sequential angles are relatively small and converging to the same angle as the iteration progresses; or the average skip angle increases significantly after LGMRES augmenting. All of these conditions are typically met for problems that display alternating behavior, although some or all are evident in other problems as well. On the other hand, LGMRES is not as helpful when: GMRES($m$) skip angles are not small; GMRES($m$) sequential angles vary greatly from cycle to cycle; GMRES($m$) converges in a small number of iterations; or GMRES($m$) skip angles and sequential angles are near zero, indicating stalling. We believe that the LGMRES augmenting scheme most benefits problems that are close to symmetric in some sense as these are the problems for which alternating is most pronounced. As we pointed out in Section 3.1, the impact of LGMRES generally cannot be predicted by a scalar measurement of symmetry (or normality), and we
have seen the algorithm perform well for a variety of problems.

5.2. An example. A simple convection-diffusion problem for which LGMRES performs poorly provides insight as to the extent that degree of symmetry and normality can affect LGMRES performance, at least for one particular problem. Recall the problem considered in Sections 4.1 and 4.2: \( u_{xx} + u_{yy} + D u_x = -(41)^2 \). We again examine three cases with increasing degree of nonsymmetry: \( D = 1, D = 41, \) and \( D = 41^2 \).

In Table 5, results similar to those presented in Figure 4 are listed. However, now we compare \( \text{GMRES}(m) \) with \( \text{GMRES}(m, 1) \) to better examine the effect of appending one error approximation to the Krylov subspace. Whereas previously presented results compared methods with equal-sized approximation spaces or equal storage requirements, here the methods have equal-sized Krylov subspaces at each cycle. The column labeled “Adaptive” is discussed in the next section. In addition, a departure from normality measurement is given in Table 5 by

\[
\frac{||\Delta||_F}{||A||_F}
\]

where \( ||\Delta||_F \) is the Frobenius norm of the strictly upper triangular part of the Schur form of \( A \) (e.g., see [17]). An indication of the symmetry of \( A \) is given by

\[
\frac{||A - A^T||_2}{||A||_2}
\]

First consider morgan_1. With \( D = 1 \), the coefficient matrix \( A \) is nearly symmetric, and the residual vectors alternate noticeably with \( \text{GMRES}(m) \). Therefore, as expected, the number of matrix-vector multiplies to convergence with \( \text{GMRES}(m, 1) \) is much smaller than for \( \text{GMRES}(m) \), particularly for the \( m = 10 \) case which is particularly slowed by alternating effects.

Next, morgan_41 with \( D = 41 \) is an excellent example of the type of problem for which LGMRES performs very poorly. Because this problem converges fairly quickly with \( \text{GMRES}(m) \) and is far from symmetric, we did not expect \( \text{GMRES}(m, 1) \) to be very helpful. But the fact that \( \text{GMRES}(m, 1) \) actually slows convergence considerably was unanticipated. However, we have since found that LGMRES generally
performs poorly on problems for which the GMRES\(m\) iteration count increases with increasing \(m\), such as morgan_41. Typically the larger the restart parameter \(m\), the fewer iterations GMRES\(m\) requires [20]. However, as we see here, occasionally the reverse can be true (e.g., see [11, 14]).

Finally consider morgan_1681 with \(D = 41^2\). This problem is nearly skew-symmetric and benefits only slightly from the augmenting scheme of LGMRES\(m\), 1). Though increasing \(k\) for this problem further reduces the number of matrix-vector multiplies to convergence, the improvement over GMRES\(m\) is still quite modest compared to that of the nearly symmetric case. In general, we find that nearly skew-symmetric problems do not benefit as much from LGMRES as do nearly symmetric problems.

5.3. An Adaptive Version. The morgan_41 problem highlights the need for a potential improvement to the LGMRES algorithm; in particular, an adaptive version that determines whether or not to augment would be beneficial. Designing a simple adaptive LGMRES algorithm effective for all test cases and for all values of \(m\) has proved difficult.

We briefly describe our most promising effort to date. This adaptive algorithm is based on the assumption that skip angles should be increased such that the residual vectors at the end of every other cycle are closer to orthogonal. After \(m\) standard Arnoldi iterations in restart cycle \(i + 1\), we form the current residual \(\tilde{r}_{i+1}\). In the \(k = 1\) case, the decision is made to augment during cycle \(i + 1\) if \(\langle \tilde{r}_{i+1}, A\tilde{z}_i \rangle > 0\). Referring back to Theorem 5, note that \(\langle \tilde{r}_{i+1}, A\tilde{z}_i \rangle = -\langle A\tilde{z}_{i+1}, A\tilde{z}_i \rangle\). Results for this algorithm are given in the right column of Table 5 and are decidedly mixed. While this adaptive method usually mitigates the extent to which LGMRES fails on tricky problems, it can be less effective than standard LGMRES on others.

Deciding whether or not to augment within a given restart cycle is difficult. We find that analysis within a single restart cycle is not sufficient as augmenting has a cumulative effect. An analysis of convergence across cycles (for both GMRES and LGMRES) would provide a better understanding of the behavior of LGMRES and enable us to design a more effective adaptive strategy.

6. Concluding Remarks. In this paper, we have described a method that accelerates the convergence of GMRES\(m\). We have also discussed some interesting observed properties of the convergence of GMRES\(m\) that motivated the algorithm’s development. Experimental results demonstrate that the LGMRES augmentation scheme is an effective accelerator for GMRES\(m\) with or without preconditioning. Furthermore, the algorithm is straightforward and easy to implement. However, LGMRES is not typically a substitute for preconditioning and does not help when a problem stalls for a given restart parameter. Possible improvements to the algorithm include a robust adaptive variant. In future work, we will describe a more memory-efficient block implementation of the LGMRES algorithm.

Acknowledgments. We thank Oliver Ernst for providing us with his MATLAB implementation of GCROT.

REFERENCES

ACCELERATING THE CONVERGENCE OF RESTARTED GMRES