Deflation Strategies to Improve the Convergence of Communication-Avoiding GMRES

Ichitaro Yamazaki, Stanimire Tomov, and Jack Dongarra
Department of Electrical Engineering and Computer Science
University of Tennessee, Knoxville, U.S.A.
Email: {iyamazak, tomov, dongarra}@eecs.utk.edu

***Abstract*—The generalized minimum residual (GMRES) method is a popular method for solving a large-scale sparse nonsymmetric linear system of equations. On modern computers, especially on a large-scale system, the communication is becoming increasingly expensive. To address this hardware trend, a communication-avoiding variant of GMRES (CA-GMRES) has become attractive, frequently showing its superior performance over GMRES on various hardware architectures. In practice, to mitigate the increasing costs of explicitly orthogonalizing the projection basis vectors, the iterations of both GMRES and CA-GMRES are restarted, which often slows down the solution convergence. To avoid this slowdown and improve the performance of restarted CA-GMRES, in this paper, we study the effectiveness of deflation strategies. Our studies are based on a thick restarted variant of CA-GMRES, which can implicitly deflate a few Ritz vectors, that approximately span an eigenspace of the coefficient matrix, through the standard orthogonalization process. This strategy is mathematically equivalent to the standard thick-restarted GMRES, and it requires only a small computational overhead and does not increase the communication or storage costs of CA-GMRES. Hence, by avoiding the communication, this deflated version of CA-GMRES obtains the same performance benefits over the deflated version of GMRES as the standard CA-GMRES does over GMRES. Our experimental results on a hybrid CPU/GPU cluster demonstrate that thick-restart can significantly improve the convergence and reduce the solution time of CA-GMRES. We also show that this deflation strategy can be combined with a local domain decomposition based preconditioner to further enhance the robustness of CA-GMRES, making it more attractive in practice.**

I. Introduction

The cost of executing software can be modeled as a function of its computational and communication costs (we ignore the potential overlap of the computation and communication, which could reduce the cost by a factor of two). For instance, we can model the computational cost based on the number of required arithmetic operations, while the communication includes data movement or synchronization between parallel execution units, as well as data movement between the levels of the local memory hierarchy. On modern computers, the communication is becoming increasingly expensive compared to the computation, in terms of both time and energy consumption. It is critical to take such hardware trends into consideration when designing high-performance software for new and emerging computers.

The Generalized Minimum Residual (GMRES) method [13] is a popular Krylov subspace projection method for solving a large-scale nonsymmetric linear system of equations, $A\mathbf{x} = \mathbf{b}$. To address the current hardware trend, the techniques, which were originally proposed as a s-step method [21], have been adapted to avoid communication of GMRES in recent years [5]. We studied the performance of this communication-avoiding variant of GMRES (CA-GMRES) on multicore CPUs with multiple GPUs on a single compute node [23] and on a hybrid CPU/GPU cluster [25]. Such a hybrid CPU/GPU architecture is becoming popular in high-performance computing due to their potential of enabling exascale computing [3], but the same hardware trend exists and its communication cost is becoming increasingly expensive. Our performance results demonstrated that CA-GMRES can obtain speedups of about $2\times$ over GMRES by avoiding the communication on such hybrid architectures (using up to 120 GPUs).

Using GMRES or CA-GMRES, the solution converges with nonincreasing residual norm. However, each basis vector is explicitly orthogonalized against the previous basis vectors, and as the subspace dimension grows, it becomes increasingly expensive to generate a new basis vector in terms of its computation, communication, and storage requirements. To mitigate this increasing costs of generating a large projection subspace, the iteration is generally restarted after computing a fixed number of basis vectors. Since the dimension of the subspace is now limited, restarting the iteration often slows down, or even prevents, the solution convergence. To mitigate this convergence slowdown of GMRES, several techniques have been proposed that deflate an approximate eigenspace associated with the smallest eigenvalues of the coefficient matrix A (such eigenspace is known to slow down the GMRES's convergence [18]). To examine the effectiveness of deflation techniques on CA-GMRES, in this paper, we adapt a thick restarting strategy [10] and integrate two deflation techniques into CA-GMRES, implicit deflation [1] and implicit restart [9]. Our experimental results demonstrate that thick-restart requires only a small increase in the computation, while greatly improving the convergence of CA-GMRES. These deflation strategies are mathematically equivalent to those proposed for GMRES and do not increase the communication or storage cost of CA-GMERS. As a result, the deflated variant of CA-GMRES obtains the same performance benefits over the deflated GMRES as the standard CA-GMRES does over GMRES, improving the robustness of CA-GMRES, and making it more attractive in practice.

Recently, we proposed a domain decomposition (DD) based

preconditioning framework that have shown to work seamlessly with a CA-Krylov method [25]. We observed that in many cases, the preconditioner is effective in reducing both the iteration counts and the total solution time. This itself is a significant contribution since one major reason why the CA-Krylov method has not been widely adapted in practice is the lack of such preconditioners. However, this DD preconditioner is local in nature, and its effectiveness may degrade on a larger number of subdomains. Since the deflation strategy studied in this paper can be combined with any preconditioner, it may provide a simple framework to introduce a global preconditioner on top of the local DD preconditioner. We present several experimental results to examine this potential.

The rest of the paper is organized as follows; in Section II, we first review CA-GMRES and briefly describe our implementation on a hybrid CPU/GPU cluster. We then outline, in Section III, our thick-restarted CA-GMRES, and in Sections IV and V, respectively, we describe the implicit deflation and implicit restart strategies, which our thick-restart strategy uses to generate its deflation subspace. Finally, in Section VI, we study the effects of deflation on the performance of CA-GMRES with and without preconditioning on a hybrid CPU/GPU cluster. We provide our final remarks in Section VII.

We use the following notations in the rest of the paper: the j-th column of a matrix A is denoted by $\mathbf{a}_j$, while $A_{j_1:j_2}$ is the submatrix consisting of the j_1-th through the j_2-th column of A, $A_{i_1:i_2,j_1:j_2}$ is the submatrix consisting of the i_1-th through the i_2-th rows of $A_{j_1:j_2}$, and $a_{i,j}$ is the (i,j)-th element of A. We use $[\mathbf{x};\mathbf{y}]$ to represent the vector with a vector $\mathbf{x}$ stacked on top of another vector $\mathbf{y}$, i.e., $[\mathbf{x};\mathbf{y}] = \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix}$, while $\mathbf{0}_k$ is a k-length vector whose elements are all zeros, and $\mathbf{e}_k$ is the k-th column of an identity matrix, whose dimension should be clear from the context. The dimension of the coefficient matrix A is denoted by n.

II. Communication-avoiding GMRES

GMRES's j-th iteration generates a new basis vector $\mathbf{q}_{j+1}$ for a Krylov projection subspace [13]. This is done by first multiplying the previously-generated basis vector $\mathbf{q}_j$ with the sparse coefficient matrix A (*SpMV*), and then orthonormalizing (*Orth*) the resulting vector against all the previously-orthonormalized basis vectors $\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_j$. This explicit orthogonalization of the basis vectors becomes increasingly expensive as the iteration proceeds. To avoid the increasing cost of computing a large projection subspace, the iteration is restarted after computing a fixed number $m+1$ of basis vectors. Before restart, GMRES updates the approximate solution $\widehat{\mathbf{x}}$ by solving a least-squares problem $\mathbf{g} := \arg\min_{\mathbf{t}} \|\mathbf{c} - H\mathbf{t}\|_2$, where $\mathbf{c} := Q_{1:m+1}^T\mathbf{r} = [\|\mathbf{r}\|_2; \mathbf{0}_m]$, $\mathbf{r}$ is the residual vector (i.e., $\mathbf{r} = \mathbf{b} - A\widehat{\mathbf{x}}$), $H := Q_{1:m+1}^T A Q_{1:m}$, and the approximate solution is updated by $\widehat{\mathbf{x}} := \widehat{\mathbf{x}} + Q_{1:m}\mathbf{g}$. Then, the iteration is restarted with the new residual vector $\mathbf{r}$ as the starting vector (i.e., $\mathbf{q}_1 = \mathbf{r}/\|\mathbf{r}\|_2$).

The matrix H, a by-product of the orthogonalization procedure, has an upper Hessenberg form. Hence, the least-squares problem to update the approximate solution can be efficiently solved, requiring only about $3(m+1)^2$ floating-point operations (flops). On the other hand, for an n-by-n matrix A with $nnz(A)$ nonzeros, *SpMV* and *Orth* require a total of about $2m \cdot nnz(A)$ and $2m^3n$ flops over the m iterations, respectively (i.e., $n, nnz(A) \gg m$). Hence, the GMRES's computational cost is typically dominated by *SpMV* and *Orth*. In addition, *SpMV* and *Orth* dominate the communication cost of GMRES. This includes point-to-point messages or neighborhood collectives for *SpMV*, and global all-reduces in *Orth*, as well as data movement between the levels of the local memory hierarchy (for reading the sparse matrix and for reading and writing vectors, assuming they do not fit in cache). On modern computers, such communication is becoming increasingly expensive compared to computation, and as a result, *SpMV* and *Orth* often dominate the solution time of GMRES. CA-GMRES [5] aims to reduce this communication by redesigning the algorithm and replacing *SpMV* and *Orth* with three new kernels – matrix powers kernel (*MPK*), block orthogonalization (*BOrth*), and tall-skinny QR (*TSQR*) – that generate and orthogonalize a set of s basis vectors all at once. In theory, CA-GMRES generates these s basis vectors with the communication cost that is no more than that of a single GMRES iteration (plus a lower-order term) [7].

The combined cost of *MPK*, *BOrth*, and *TSQR* to generate the basis vectors typically dominates the total cost of CA-GMRES. To accelerate the solution process, our CA-GMRES implementation on a hybrid CPU/GPU cluster generates the basis vectors on the GPUs, while each MPI process redundantly solves the least squares problem on the CPUs. We use either a matrix reordering or a graph partitioning algorithm to distribute both the matrix A and the basis vectors $\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_{m+1}$ over the GPUs in a 1D block row format (see Section VI). A more detailed description of our implementation can be found in [23], [25].

III. Thick-restarted CA-GMRES

Restarting slows down the solution convergence because some useful information (e.g., eigenspace associated with the smallest eigenvalues) is discarded and must be recomputed during each restart loop. In this section, we outline the thick restart strategy to improve the solution convergence of CA-GMRES by retaining some of the useful information at each restart. In particular, this strategy restarts the iteration with a few approximate eigenvectors, called Ritz vectors, of A in addition to the current residual vector $\mathbf{r}$. It is based on the application of the Krylov Schur method for solving an eigenvalue problem [17] to GMRES to solve a linear system. Our approach is mathematically equivalent to the implicit deflation and implicit restart [1], [9]. However, our implementation is based on a thick restart algorithm, and is different from these approaches [1], [9] which are based on the implicitly restarted Arnoldi algorithm [15]. A thick restarted variant of

CA-GMRES(A, M, $\mathbf{b}$, s, m):
repeat (restart-loop)
1.Generate Krylov Basis: $O(m \cdot nnz(|A|) + m^2 n)$ flops on GPUs.
($\widehat{\mathbf{x}} = \mathbf{0}_n$, $\mathbf{q}_1 = \mathbf{r}/\|\mathbf{r}\|_2$, $\mathbf{r} = \mathbf{b} - A\widehat{\mathbf{x}}$, and $k = 0$, initially)
1.1. Initialization
Sparse Matrix Vector Multiply (*SpMV*):
$\mathbf{q}_{k+1} := A\mathbf{q}_k$
Orthogonalization (*Orth*):
orthonormalize $\mathbf{q}_{k+1}$ against $Q_{1:k}$
1.2. Restart-loop
for $j = k+1, 1+s, \ldots, m$ **do**
Matrix Powers Kernel (*MPK*):
for $k = j, j+1, \ldots, j+s-1$ **do**
$\mathbf{q}_{k+1} := A\mathbf{q}_k$ (*SpMV*)
end for
Block Orthonormalization (*BOrth*):
orthogonalize $Q_{j+1:j+s}$ against $Q_{1:j}$, i.e.,
$Q_{j+1:j+s} := Q_{j+1:j+s} - Q_{1:j}R_{1:j,j+1:j+s}$
Tall-Skinny QR (*TSQR*) factorization:
orthonormalizing $Q_{j+1:j+s}$ against each other, i.e.,
$Q_{j+1:j+s}R_{j+1:j+s,j+1:j+s} = Q_{j+1:j+s}$
Projected Matrix Computation
compute $H_{j:j+s-1,j+1:j+s}$
from $H_{1:j,1:j+1}$ and $R_{1:j+s,j+1:j+s}$
end for

2.Restart: $O(m^2)$ and $O(nmk)$ flops on CPUs and GPUs, respectively.
2.1 solve $\mathbf{g} = \min_{\mathbf{t}} \|H\mathbf{t} - Q_{1:m+1}^T\mathbf{r}\|_2$, and
compute $\widehat{\mathbf{x}} = \widehat{\mathbf{x}} + Q_{1:m}\mathbf{g}$ and $\mathbf{r} = \mathbf{b} - A\widehat{\mathbf{x}}$
2.2 compute k Ritz vector $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k$ to keep
(see Sections IV and V)
2.3 orthogonalize $[V_{1:k}\mathbf{r}]$ to generate $Q_{1:k+1}$
2.4 compute $H_{1:k+1,1:k}$
until solution convergence

Fig. 1. Thick-restarted CA-GMRES(s, k, m).

an implicitly-restarted GMRES has been described in [10]. In this section, we extend that to CA-GMRES.[1]

We assume that to restart the iteration, the approximate eigenpairs, called Ritz pairs, $(\theta_i, \mathbf{v}_i)$, have been generated such that their residual vectors, $A\mathbf{v}_i - \theta_i\mathbf{v}_i$, align with the current residual vector, $\mathbf{r}$, of the linear system (i.e., $\mathbf{r} = \mathbf{b} - A\widehat{\mathbf{x}}$). We show two approaches to generate such vectors in Sections IV and V. When k such vectors are kept at restart, they satisfy the following relation:

$$AV_{1:k} = [V_{1:k}, \bar{\mathbf{r}}]\, T_{1:k+1,1:k}, \tag{1}$$

where $V_{1:k}$ are the k kept Ritz vectors, $T_{1:k+1,1:k} = \left[\Sigma_{1:k,1:k}; \mathbf{t}^T\right]$, $\Sigma_{1:k,1:k}$ is a diagonal matrix with the corresponding kept Ritz values on diagonal (i.e., $\Sigma_{1:k,1:k} = \mathrm{diag}(\theta_1, \theta_2, \ldots, \theta_k)$), the i-th element of $\mathbf{t}$ is equal to the residual norm of the i-th Ritz pairs (i.e., $t_i = \|A\mathbf{v}_i - \theta_i\mathbf{v}_i\|_2$), and $\bar{\mathbf{r}} = \mathbf{r}/\|\mathbf{r}\|_2$. To restart the iteration, we compute a QR factorization of the $k+1$ kept vectors, $\widehat{Q}_{1:k+1}\widehat{R}_{1:k+1,1:k+1} := [V_{1:k}\bar{\mathbf{r}}]$, where $\widehat{Q}_{1:k+1}$ has orthonormal columns, $\widehat{R}_{1:k+1,1:k+1}$ is upper-triangular, and we put the "hat" over the next basis vectors $\widehat{Q}_{1:k+1}$ to distinguish them from the current basis vectors $Q_{1:m+1}$. Then, the relation (1) becomes

$$A\widehat{Q}_{1:k} = \widehat{Q}_{1:k+1}\widehat{H}_{1:k+1,1:k}, \tag{2}$$

[1]It was pointed out that similar studies have been independently conducted in [20].

where $\widehat{H}_{1:k+1,1:k} = \widehat{R}_{1:k+1,1:k+1}T_{1:k+1,1:k}\widehat{R}^{-1}_{1:k,1:k}$. When we encounter a complex Ritz value, we keep its conjugate pairs to form a real Schur decomposition and avoid the complex arithmetic.

By their construction (see Sections IV and V), these $k+1$ basis vectors $\widehat{Q}_{1:k+1}$ span a Krylov subspace [1], [9], which can be expanded using *MPK*, *BOrth*, and *TSQR*, as in CA-GMRES without thick-restart. The projected matrix $\widehat{H}_{1:k+1,1:k}$ is expanded correspondingly, using the formula similar to those in CA-GMRES without thick-restart [5, Section 3.3.4]. The only difference is that when computing $\widehat{H}_{k+1,k+1:k+s}$, the $(k+1)$-th row of $\widehat{H}_{1:k+1,1:k}$ is no longer in a Hessenburg form. Hence, extra computation is needed to compute $\widehat{H}_{k+1,k+1:k+s}$ (i.e., $O(sk)$ flops). In addition since the leading $(k+1)$-by-k block, $\widehat{H}_{1:k,1:k}$, of the projected matrix is now fully dense, the computational cost of solving the least square problem is slightly increased:

$$\mathbf{g} = \arg\min_{\mathbf{t}} \|\widehat{\mathbf{c}} - \widehat{H}_{1:m+1,1:m}\mathbf{t}\|_2, \tag{3}$$

where $\widehat{\mathbf{c}} = Q_{1:m+1}^T\bar{\mathbf{r}} = [Q_{1:k+1}^T\bar{\mathbf{r}}; \mathbf{0}_{m-k}]$. Moreover, a general eigensolver is needed to compute the eigenpairs of $\widehat{H}_{1:m,1:m}$ at restart (e.g., `xGEEV` instead of `xHEEV` of LAPACK). However, these computational overheads are insignificant in comparison to the cost of generating the basis vectors (i.e., $O(m(nnz(A)+mn)$ flops). Figure 1 shows the pseudocode of the thick-restarted CA-GMRES, which is mathematically equivalent to the thick-restarted GMRES. Aside from the restarting procedure, the rest of the thick-restarted algorithm is identical to that of the CA-GMRES algorithm without thick-restart. In particular, the approximate eigenspace, $V_{1:k}$, of A is implicitly deflated during *BOrth*. In Section VI, we present performance results to demonstrate that thick-restart requires only an insignificant overhead, while the convergence can be dramatically improved. As a result, the solution time may be greatly reduced by thick-restarting the CA-GMRES iteration.

Recently, a CA formulation [2] of a deflation technique [14] was integrated into a CA variant of the Conjugate Gradient (CG) method. This formulation is mathematically equivalent to a classical deflated CG [14], and can be viewed as a type of preconditioner based on a low-rank matrix, where a general low-rank subspace is explicitly deflated from each basis vector (their experiments were based on the low-rank matrices, which consist of the eigenvectors corresponding to the smallest eigenvalues of the coefficient matrix, while in our experiments, we used the thick-restart strategy to implicitly reflate the approximation to the same eigenspace). In contrast, by restricting our deflation subspace to a subspace spanned by the Ritz and residual vectors, thick-restarting strategy only requires a small computational overhead of recomputing the Ritz vectors at each restart, and it can implicitly deflate the subspace during the standard orthogonalization process. Finally, it has been proposed to exploit data sparsity in the matrix powers computation by representing a general sparse matrix as a combination of a low-rank matrix and a remaining matrix which has a sparsity structure more favorable for *MPK* [6]. We

are currently investigating a general low-rank preconditioner for CA-GMRES based on this idea. The deflation techniques studied in this paper can be combined with such low-rank preconditioners.

IV. Implicit Deflation

Implicit deflation [1] aims to retain some information at restart by keeping a few Ritz vectors in addition to the new residual vector, $\mathbf{r} = \mathbf{v} - A\widehat{\mathbf{x}}$. Before computing the Ritz vectors, the projected matrix $H_{1:m,1:m}$ is rotated such that the residual vectors, $A\mathbf{v}_i - \theta_i\mathbf{v}_i$, of the Ritz pairs $(\theta_i, \mathbf{v}_i)$ are aligned with the residual vector $\mathbf{r}$;

$$AQ_{1:m} = Q_{1:m}\bar{H}_{1:m,1:m} + \bar{h}_{m+1,m}\bar{\mathbf{r}}\mathbf{e}_m^T, \quad (4)$$

where $\bar{H} = H_{1:m,1:m} - \frac{1}{\zeta}\mathbf{z}\mathbf{e}_m^T$, $\bar{h}_{m+1,m} = \frac{h_{m+1,m}}{\zeta}$, $\mathbf{z} = Q_{1:m}^T\mathbf{r}$, and $\zeta = \mathbf{q}_{m+1}^T\mathbf{r}$. Then, the Ritz pairs are computed as $(\theta_i, \mathbf{v}_i = Q_{1:m}\mathbf{x}_i)$, where $(\theta_i, \mathbf{x}_i)$ is the eigenpairs of $\bar{H}_{1:m,1:m}$;

$$\bar{H}_{1:m,1:m}X_{1:m} = X_{1:m}\Sigma_{1:m,1:m}, \quad (5)$$

where $\Sigma_{1:m,1:m} = \mathrm{diag}(\theta_1, \theta_2, \ldots, \theta_m)$. Moreover, because of (4) and (5), the residual vectors of the Ritz pairs satisfy

$$AV_{1:m} - V_{1:m}\Sigma_{1:m,1:m} = h_{m+1,m}\mathbf{q}_{m+1}\mathbf{e}_m^T X_{1:m}.$$

Hence, the i-th residual norm t_i of (1) can be computed cheaply by $t_i = h_{m+1,m}|x_{m,i}|$. We refer to the thick-restarted CA-GMRES using these Ritz pairs associated with $\bar{H}_{1:m,1:m}$ as ID-CAGMRES.

V. Implicit Restart

Instead of using Ritz vectors, *implicit restart* [9] restarts the iteration, using harmonic Ritz pairs $(\theta_i, \mathbf{v}_i = Q_{1:m}\mathbf{x}_i)$, where $(\theta_i, \mathbf{x}_i)$ is now the eigenpairs of a generalized eigenvalue problem,

$$H_{1:m,1:m}^T\mathbf{x}_i = \frac{1}{\theta_i}(H_{1:m,1:m}^T H_{1:m,1:m} + h_{m+1,m}^2\mathbf{e}_m\mathbf{e}_m^T)\mathbf{x}_i.$$

In our implementation, we solve an equivalent standard eigenvalue problem,

$$(H_{1:m,1:m} + h_{m+1,m}^2\mathbf{f}\mathbf{e}_m^T)\mathbf{x}_i = \theta_i\mathbf{x}_i,$$

where $\mathbf{f} = H_{1:m,1:m}^{-T}\mathbf{e}_m$. In many cases, harmonic Ritz values provide more accurate approximation to the smallest eigenvalues than the standard Ritz values do [11]. With the harmonic Ritz pairs, the residual vectors are now given by

$$\begin{aligned} &AV_{1:m} - V_{1:m}\Sigma_{1:m,1:m} \\ &\quad = h_{m+1,m}(h_{m+1,m}V_{1:m}\mathbf{f} + \mathbf{q}_{m+1})\mathbf{e}_m^T X_{1:m}, \end{aligned} \quad (6)$$

and the residual norm can be still computed cheaply for the convergence check. In our implementation, instead of using (6) to compute the residual norms, we recover the Arnoldi relation (2) by computing the i-th element t_i of the vector $\mathbf{t}$ based on the following equalities:

$$\begin{aligned} t_i &= (A\mathbf{v}_i - \theta_i\mathbf{v}_i)^T\bar{\mathbf{r}} \\ &= -\theta_i\mathbf{v}_i^T\bar{\mathbf{r}} \\ &= -\theta_i\widehat{R}_{1:i,i}^T(\widehat{Q}_{1:k}^T\bar{\mathbf{r}}), \end{aligned}$$

where the second equality follows since $(A\mathbf{v}_i)^T\bar{\mathbf{r}} = \mathbf{0}$ [9, Lemma 5.3], and $\widehat{Q}_{1:k}^T\bar{\mathbf{r}}$ in the third equality is needed to solve the least-squares problem (3) at the next restart. Hence, the residual norms can be computed without significant computational overhead. We refer to the thick-restarted CA-GMRES using these harmonic Ritz pairs as IR-CAGMRES.

Name	Source	n	nnz/n
`sherman3`	UF collection	$5,005$	4.0
`PDE(1.0275)`	Trillinos	$1,030,301$	26.5

Fig. 2. Test matrices.

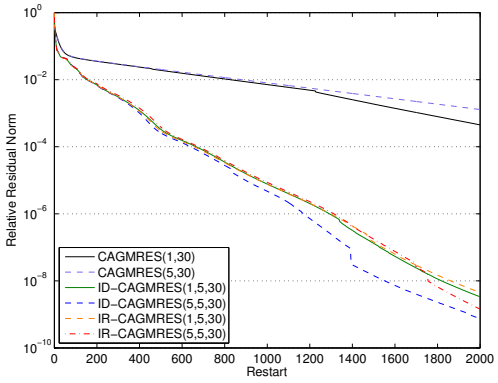
VI. Experimental Results

We now study the effectiveness of the thick-restart strategy to improve the convergence and performance of CA-GMRES with restart. Table 2 lists the two test matrices used for our experiments. All the experiments were conducted on the Keeneland system[2] at the Georgia Institute of Technology. Each of its compute nodes consists of two six-core Intel Xeon CPUs and three NVIDIA M2090 GPUs, with 24GB of main CPU memory per node and 6GB of memory per GPU. We used the GNU `gcc` 4.4.6 compiler and CUDA `nvcc` 5.0 compiler with the optimization flag `-O3`, and linked with Intel's Math Kernel Library (MKL) version 2011_sp1.8.273 and OpenMPI 1.6.1.

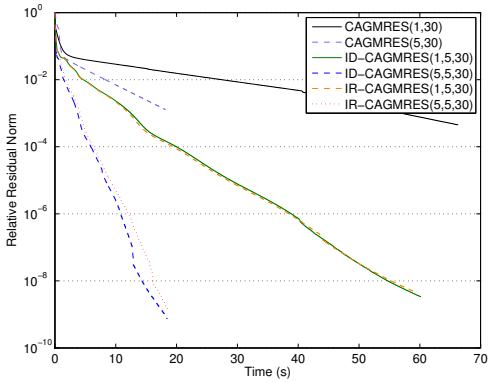
To distribute the matrix among the GPUs, we used a k-way graph partitioning algorithm of METIS[3]. The performance of CA-GMRES depends critically on the orthogonalization algorithms. For all of our experiments in this paper, we used the classical Gram-Schmidt process [4] for *BOrth* and the Cholesky QR factorization [16] for *TSQR*. In order to ensure the numerical stability, we always performed the full reorthogonalization for both *BOrth* and *TSQR*. To compute *SpMV*, the local submatrix of the coefficient matrix A is stored in the ELLPACKT sparse matrix storage format on each GPU [19]. Though our implementation allows each MPI process to manage multiple GPUs on a node, in this paper, we let each process manage one GPU. In our previous experiments [23], [25], these configurations gave good performance of CA-GMRES without deflation. Finally, for solving symmetric eigenvalue problems, there are several heuristics to select the Ritz pairs to keep at each restart [22], [24]. For our experiments with ID-CAGMRES and IR-CAGMRES for solving nonsymmetric linear systems in this paper, we simply kept the fixed number k of the Ritz vectors corresponding to the smallest Ritz values, unless otherwise specified.

Figure 3 compares the convergence history of CA-GMERS(s,m) with those of ID-CAGMRES(k,s,m) and IR-CAGMRES(k,s,m), in terms of the residual norm. The test matrix used for this experiment is called `sherman3` and arises from an oil reservoir simulation. This matrix belongs to the group of the test matrices used in the original papers [1], [9], and is available from the University of Florida Sparse

[2] http://keeneland.gatech.edu/KDS
[3] www.cs.umn.edu/~metis

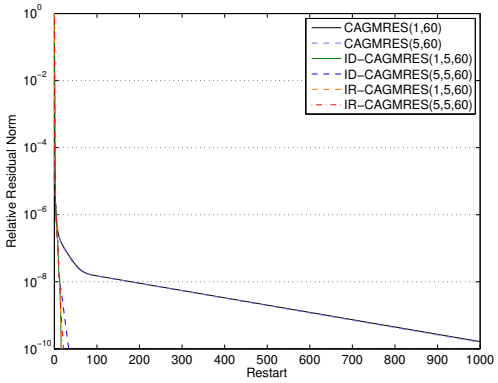


(a) at restart.

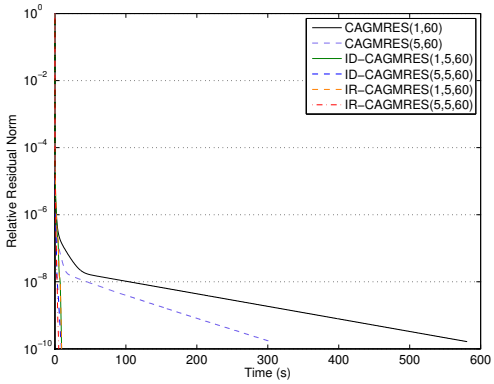


(b) vs. time.

Fig. 3. Residual norm convergence for `sherman3` matrix, 6 GPUs.



(a) at restart.



(b) vs. time.

Fig. 4. Residual norm convergence for `PDE(1.0275)` matrix, 6 GPUs.

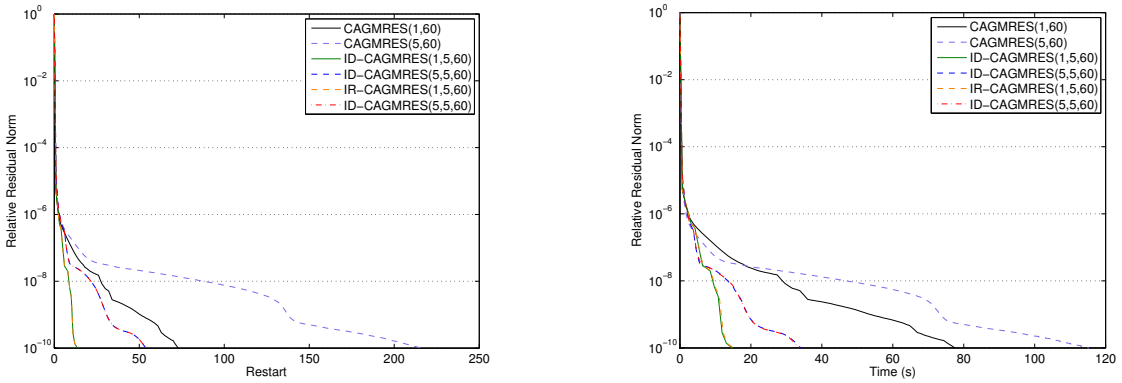
Matrix Collection[4]. Figure 3(a) clearly demonstrates that the convergence rate can be greatly improved by deflating the approximate eigenspace. CA-GMRES achieved about the same convergence using $s = 1$ and 5, where CA-GMRES is equivalent to GMRES when $s = 1$. In all of our experiments, both implicit deflation and implicit restart strategies obtained similar improvements. Figure 3(b) shows the convergence history against time. Using a larger value of s (i.e., $s = 5$), CA-GMRES avoids communication, and reduced the iteration time significantly. The solution time was further reduced by deflating the approximate eigenspace. Figure 4 compares the convergence histories for one of the PDE matrices used in our previous paper [25]. The matrix is symmetric indefinite, and we see more significant benefits of deflation.

For the rest of this section, we study the effects of deflation on the preconditioned CA-GMRES convergence. Our preconditioner is based on domain decomposition (DD) techniques proposed in [25]. This DD preconditioner is similar to block Jacobi preconditioner, but each diagonal block is split into smaller diagonal blocks so that the preconditioner can be applied without requiring any additional communication from what is already needed by *MPK* (i.e., the number of GPUs is equal to the number of subdomains or diagonal blocks). For our experiments in this paper, we focused on the right preconditioning, that generates the right-preconditioned Krylov subspace $\mathcal{K}_{m+1}(AM^{-1}, \mathbf{q}_1) \equiv \text{span}(\mathbf{q}_1, AM^{-1}\mathbf{q}_1, \ldots, (AM^{-1})^m\mathbf{q}_1)$, where M is our DD preconditioner and $\mathbf{q}_1$ is a starting vector. For the inexact solution of each subdomain problem, we used an ILU(0) preconditioner while diagonal Jacobi preconditioners were used on both underlap and overlap. Each MPI process computes its local ILU(0) preconditioner on the CPU, using the ITSOL package[5], copies it to the corresponding local GPU, and applies the preconditioner using the CuSPARSE triangular solver at each iteration.

Figure 5 shows the convergence history of the preconditioned CA-GMRES for the PDE matrix. The figure indicates that, though the convergence rate was significantly improved

[4]http://www.cise.ufl.edu/research/sparse/matrices/

[5]http://www-users.cs.umn.edu/~saad/software/ITSOL/



(a) at restart. (b) vs. time.

Fig. 5. Residual norm convergence for `PDE(1.0275)` matrix with *Preco*, 6 GPUs.

using this DD preconditioner, it can be further improved by deflating the approximate eigenspace. The effectiveness of our DD preconditioner degraded with a greater step size s. This is because in order not to increase the communication for the larger step size, the diagonal blocks of the preconditioner must be split into smaller blocks. Even when the deflation technique is used in combination with the DD preconditioner, the convergence rates still degraded with a larger step size. In fact, when $s = 5$ with the deflation, the iteration count was greater using the preconditioner. This is because the effectiveness of the deflation depends on the spectrum distribution of the coefficient matrix A or the preconditioned matrix AM^{-1}. The PDE matrix A has a few negative eigenvalues. Deflating the eigenspace associated with these eigenvalues significantly improves the convergence. In contrast, the spectral distribution of the preconditioned matrix AM^{-1} may not be as favorable for the deflation. In the end, with the overhead of applying the preconditioner, the solution time was shorter, using only the deflation technique without the preconditioner, for this particular test matrix.

Figure 6 shows the convergence histories with different numbers of subdomains. Our DD preconditioner is local in nature, and the iteration count of the preconditioned CA-GMRES often increases with the number of subdomains. With deflation, the number of iterations was about the same on 6 and 12 GPUs. However, compared to using 12 GPUs, both the iteration count and the solution time still increased on 24 GPUs. Figure 7 shows the results, still using the preconditioner, but now with different configurations of the kept Ritz values (i.e., five smallest, four smallest plus one largest, and three smallest plus two largest Ritz values). In comparison to keeping just the smallest Ritz values, keeping the largest Ritz values seems to slow down the convergence. These results indicate that these largest Ritz values converge quickly. On the other hand, the smallest Ritz values seem to converge slowly. Since these Ritz values must be recomputed during each restart loop, the solution convergence slowed down. As a result, for this test matrix, it was more effective to keep the smallest Ritz values rather than keeping the largest Ritz values at restart.

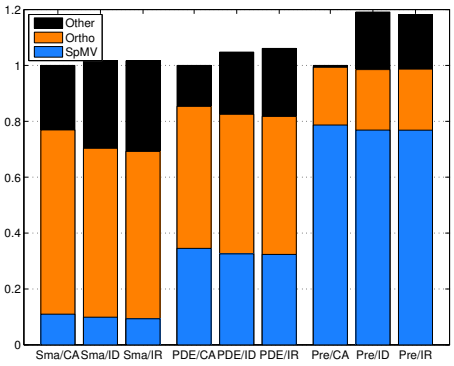
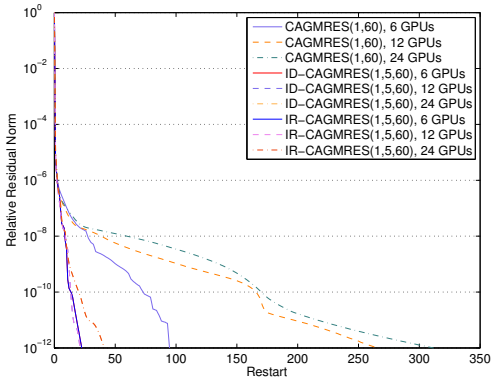


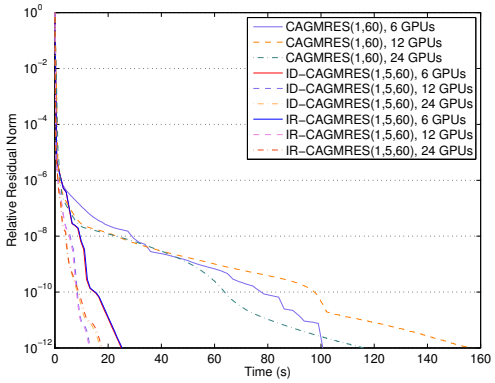
Fig. 8. Breakdown of average restart-loop time (normalized with standard CA-GMRES), 6 GPUs. In the figure, "Sma" and "PDE" denote the `sherman3` and `PDE(1.0275)` matrices, respectively, while "CA," "ID," and "IR" respectively denote CA-GMRES, ID-CAGMRES, and IR-GMRES.

Figure 8 shows the breakdown of the average time spent generating $m + 1$ basis vectors and restarting the iteration. In the figure, "Other" includes the restarting time, and shows that the thick-restarting based on both the implicit deflation and the implicit restart slightly increased the restarting time.[6] With preconditioning, the relative cost of the thick-restart increased further since we need to apply the preconditioner to update the approximate solution (i.e., $\widehat{x} = \widehat{x} + W_{1:m}\mathbf{g}$, where $W_{1:m} = M^{-1}Q_{1:m}$ and $Q_{1:m}$ is the orthonormal basis vectors). More specifically, our implementation of CA-GMRES without deflation is based on a flexible version of GMRES [12], and saves the preconditioned basis vectors. Hence, without deflation, our CA-GMRES computes

[6]In our current implementation, the Ritz vectors $V_{1:k}$ are explicitly orthogonalized against each other to form (2). The cost of the orthogonalization and of the thick restarting may be further reduced by, instead of orthogonalizing the Ritz vectors, orthogonalizing the kept eigenvectors $X_{1:k}$ of the projected matrix $H_{1:m,1:m}$ [10].

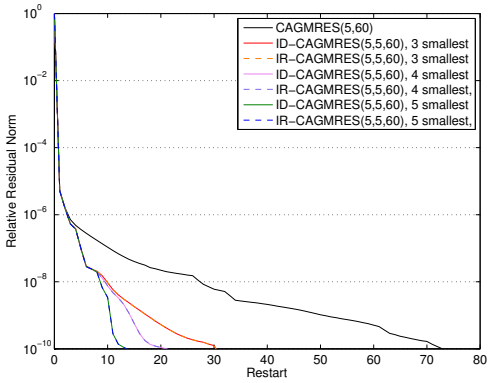


(a) at restart.

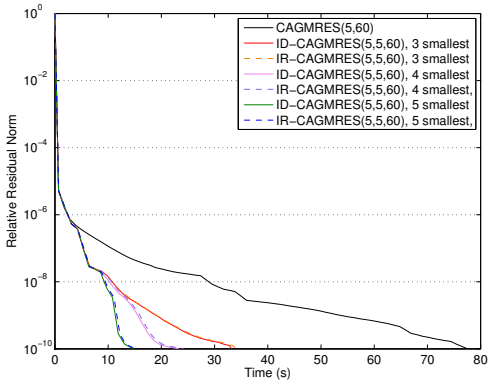


(b) vs. time.

Fig. 6. Residual norm convergence for `PDE(1.0275)` matrix with *Preco* and different numbers of subdomains.



(a) at restart.



(b) vs. time.

Fig. 7. Residual norm convergence for `PDE(1.0275)` matrix with *Preco* and different configurations of kept Ritz values, 6 GPUs.

$\widetilde{H}_{1:m+1,1:m} := Q_{1:m+1}^T A\widetilde{W}_{1:m}$, where $\widetilde{W}_{1:m}$ spans the same subspace as $W_{1:m}$, but is generated during *MPK* and before *BOrth*. As a result, the solution can be updated without an additional application of the preconditioner. Though the projected matrices $\widetilde{H}_{1:m,1:m}$ is different from $H_{1:m,1:m}$ for $s > 1$, the computed solution is mathematically the same. With thick-restart, we use $H_{1:m,1:m}$ to update the solution because the Ritz pairs are computed from $H_{1:m,1:m}$ (see Sections III through V). Hence, we must update the solution vector based on $\widehat{x} = \widehat{x} + M^{-1}Q_{1:m}\mathbf{g}$, requiring one additional application of M^{-1} to the vector $Q_{1:m}\mathbf{g}$. Finally, when $k+1$ Ritz vectors are kept at restart, CA-GMRES iterates k less time to generate $m-k$ additional basis vectors (i.e., the dimension of the projection subspace is always the same, m). Hence, "SpMV" and "Orth" times were both slightly reduced, using deflation. In the end, the thick-restart introduced only a small overhead, while significantly reducing the iteration count, and the total solution time of CA-GMRES was reduced using the deflation.

Our focus of the paper is to study the effectiveness of deflation to improve the robustness of CA-GMRES, while our previous papers [23], [25] demonstrated the effectiveness and scalability of CA-GMRES without deflation. Before concluding this experimental section, we include Figure 9 to show the scaling of CA-GMRES on a hybrid CPU/GPU cluster

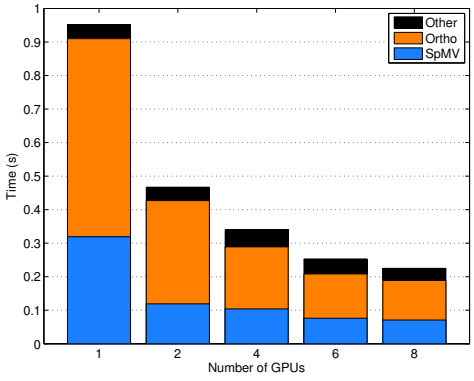


Fig. 9. Parallel strong scaling of IR-CAGMRES(5,5,60)'s restart-loop for `PDE(1.0275)` matrix.

just for a reference. More parallel performance studies of our implementation can be found in [23], [25], where we demonstrated that CA-GMRES can obtain the speedups of about two over GMRES on multicore CPUs with multiple GPUs on single compute node and on a hybrid CPU/GPU cluster. The current studies are the improvements over these previous results.

VII. Conclusion

We studied a thick-restarting strategy to improve the convergence and performance of CA-GMRES with restart. This strategy restarts the iteration using a subspace spanned by a few Ritz vectors in addition to the current residual vector, and it is mathematically equivalent to thick-restarted GMRES. Since the subspace is implicitly deflated during the standard orthogonalization process, this restarting strategy requires only a small computational overhead without increasing the communication or storage cost. Our experimental results demonstrated that the thick-restarting strategy can greatly improve the convergence of CA-GMRES, reducing the time to solution. Hence, this strategy can improve the robustness of CA-GMRES, making it more attractive in practice. We are investigating the effectiveness of *augmentation* [8] compared to deflation to improve the performance of CA-GMRES. Though augmentation requires an additional sparse-matrix multiply with the basis vectors spanning the augmented space, it may provide more flexible deflation framework since any subspace can be augmented to the Krylov subspace.

We also studied the effects of the deflation on the performance of a preconditioned CA-GMRES. Our preconditioner is based on a domain decomposition (DD) [25], and is local in nature. As a result, iteration count can increase on a larger number of subdomains. The thick-restarting strategy may provide a potential to introduce global preconditioning on top of the local DD preconditioning. Unfortunately, the effectiveness of the deflation depends on the spectral distribution of the coefficient and preconditioned matrices, and even using the thick restart strategy, the iteration count may increase on a larger number of subdomains. We are currently studying a more general framework to integrate a global low-rank preconditioner on top of our local DD preconditioner (i.e., not restricted to a subspace spanned by the Ritz vectors of the preconditioned matrix AM^{-1}). The deflation techniques studied in this paper can be combined with these local DD and global low-rank preconditioners.

Acknowledgments

This material is based on work partly supported by the U.S. Department of Energy (DOE), Office of Science, Office of Advanced Scientific Computing Research (ASCR). This research was supported in part by U.S. Department of Energy (DOE) Grant #DE-SC0010042: "Extreme-scale Algorithms & Solver Resilience (EASIR)," and NSF SI2-SSI-1339822 "Sustained Innovation for Linear Algebra Software (SILAS)." We thank the members of the EASIR project for helpful discussions.

References

[1] C. Calvez and B. Molina. Implicitly restarted and deflated GMRES. *Numerical Algorithms*, 21:261–285, 1999.

[2] E. Carson, N. Knight, and J. Demmel. An efficient deflation technique for the communication-avoiding conjugate gradient method. In *proceedings of numerical analyis and scientific computation with applications (NASCA)*, 2013.

[3] J. Dongarra and et. al. The international exascale software project roadmap. *Int. J. High Perform. Comput. Appl.*, 25:3–60, 2011.

[4] G. Golub and C. van Loan. *Matrix Computations*. The Johns Hopkins University Press, Baltimore, MD, 4rd edition, 2012.

[5] M. Hoemmen. *Communication-avoiding Krylov subspace methods*. PhD thesis, EECS Department, University of California, Berkeley, 2010.

[6] N. Knight, E. Carson, and J. Demmel. Exploiting data sparsity in matrix powers computations. In *proceedings of the international conference on parallel processing in applied mathematics*, 2013.

[7] M. Mohiyuddin. *Tuning Hardware and Software for Multiprocessors*. PhD thesis, EECS Department, University of California, Berkeley, 2012.

[8] R. Morgan. A restarted GMRES method augmented with eigenvectors. *SIAM J. Matrix Anal. Appl.*, 16:1154–1171, 1995.

[9] R. Morgan. Implicitly restarted GMRES and Arnoldi methods for nonsymmetric systems of equations. *SIAM J. Matrix Anal. Appl.*, 21:1112–1135, 2000.

[10] R. Morgan. GMRES with deflated restarting. *SIAM J. Sci. Comput.*, 24:20–37, 2002.

[11] C. Paige, B. Parlett, and H. van der Vorst. Approximate solutions and eigenvalue bounds from Krylov subspaces. *Numer. Linear Algebra Appl.*, 2:115–133, 1995.

[12] Y. Saad. A flexible inner-outer preconditioned gmres algorithm. *SIAM J. Sci. Comput.*, 14:461–469, 1993.

[13] Y. Saad and M. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 7:856–869, 1986.

[14] Y. Saad, M. Yeung, J. Erhel, and F. Guyomarc'h. A deflated version of the conjugate gradient algorithm. *SIAM J. Sci. Comput.*, 21:1909–1926, 2000.

[15] D. Sorensen. Implicit application of polynomial filters in a k-step Arnoldi method. *SIAM J. Matrix Anal. Appl.*, 13:357–385, 1992.

[16] A. Stathopoulos and K. Wu. A block orthogonalization procedure with constant synchronization requirements. *SIAM J. Sci. Comput.*, 23:2165–2182, 2002.

[17] G. Stewart. A Krylov-Schur algorithm for large eigenproblems. *SIAM J. Mat. Anal. Appl.*, 23:601–614, 2001.

[18] H. van der Vorst and C. Vuik. The superlinear convergence behavior of GMRES. *J. Comput. Appl. Math.*, 48:327–341, 1993.

[19] F. Vazquez, G. Ortega, J. Fernandez, and E. Garzon. Improving the performance of the sparse matrix vector product with GPUs. In *the proceedings of the international conference on computer and information technology (CIT)*, pages 1146–1151, 2010.

[20] D. Wakam and J. Erhel. Parallelism and robustness in GMRES with a Newton basis and deflated restarting. *Electronic Transactions on Numerical Analysis*, 40:381–406, 2013.

[21] H. Walker. Implementation of the GMRES method using Householder transformations. *SIAM J. Sci. Comput.*, 9:152–163, 1988.

[22] K. Wu and H. Simon. Thick-restart Lanczos method for large symmetric eigenvalue problems. *SIAM J. Mat. Anal. Appl.*, 22:602–616, 2000.

[23] I. Yamazaki, H. Anzt, S. Tomov, M. Hoemmen, and J. Dongarra. Improving the performance of CA-GMRES on multicores with multiple GPUs. Technical Report UT-EECS-14-722, University of Tennessee, Knoxville, 2014. To appear in the proceedings of the IEEE International Parallel and Distributed Symposium (IPDPS).

[24] I. Yamazaki, Z. Bai, H. Simon, L.-W. Wand, and K. Wu. Adaptive projection subspace dimension for the thick-restart Lanczos method. *ACM Trans. Math. Softw.*, 37, 2010.

[25] I. Yamazaki, S. Rajamanickam, E. Boman, M. Hoemmen, M. Heroux, and S. Tomov. Domain decomposition preconditioners for communication-avoiding Krylov methods on a hybrid CPU/GPU cluster. 2014. To appear in the proceedings of the Conference on High Performance Computing Networking, Storage and Analysis (SC).