Electronic Transactions on Numerical Analysis.
Volume 40, pp. 381-406, 2013.

ISSN 1068-9613.

ETNA
Kent State University
http://etna.math.kent.edu

PARALLELISM AND ROBUSTNESS IN GMRES WITH A NEWTON BASIS AND DEFLATED RESTARTING*

DESIRE NUENTSA WAKAM† AND JOCELYNE ERHEL‡

Abstract. The GMRES iterative method is widely used as a Krylov subspace technique for solving sparse linear systems when the coefficient matrix is nonsymmetric and indefinite. The Newton basis implementation has been proposed on distributed memory computers as an alternative to the classical approach with the Arnoldi process. The aim of our work here is to introduce a modification based on deflation techniques. This approach builds an augmented subspace in an adaptive way to accelerate the convergence of the restarted formulation. In our numerical experiments, we show the benefits of using this implementation with hybrid direct/iterative methods to solve large linear systems.

Key words. augmented Krylov subspaces, adaptive deflated GMRES, Newton basis, hybrid linear solvers

AMS subject classifications. 65F10, 65F15, 65F22

1. Introduction.

In this paper, we are interested in solving large systems of linear algebraic equations

$$Ax = b, \tag{1.1}$$

where A is an $n \times n$ real nonsingular matrix and b and x are n-dimensional real vectors. In practical algorithms, the original problem (1.1) is transformed into the following one

$$M_L^{-1} A M_R^{-1} \tilde{x} = M_L^{-1} b, \quad \tilde{x} = M_R x, \tag{1.2}$$

where M_L^{-1} and M_R^{-1} are the action of a preconditioning of the system from the left ($M_R = I$), from the right ($M_L = I$), or both. On parallel computers, we assume that these preconditioners are derived from some algebraic decomposition of the input matrix. However, they can be any approximation of the inverse of the matrix A, and we refer the reader to the survey on preconditioning techniques [9]. These preconditioners are generally combined with Krylov subspace methods as accelerators. The GMRES method [41] is widely used in this context. From this method, many improvements have been proposed to enhance its robustness and parallel efficiency; see, for instance, [6, 10, 12, 19, 20, 22, 24, 31, 37, 42]. In this work, we propose a new formulation of this method which combines two main approaches, namely the Newton basis GMRES [6] and the augmented basis for the restarted GMRES [31]. Our approach benefits from the enhanced parallelism of the former and the robustness of the latter. For the sake of clarity, we give here the formulation of the GMRES algorithm as first proposed by Saad and Schultz [41].

We consider the right preconditioned matrix $B \equiv AM^{-1}$ in this paper. The proposed algorithms can be derived with little effort for the left preconditioned matrix. Flexible preconditioning, which allows for an iterative preconditioner [39], can be combined with deflation [23]. In this work, we focus on a fixed preconditioning for the sake of simplicity.

Given an initial guess x_0, the GMRES method finds the j-th approximate solution x_j of the form

$$x_j \in x_0 + M^{-1}\mathcal{K}_j(B, r_0),$$

*Received December 22, 2011. Accepted July 15, 2013. Published online on October 28, 2013. Recommended by Y. Saad.
†INRIA Bordeaux, 200 Avenue de la vieille tour, 33405 Talence Cedex, France (desire.nuentsa_wakam@inria.fr).
‡INRIA, Campus universitaire de Beaulieu, 35042 Rennes Cedex, France (jocelyne.erhel@inria.fr).

ETNA
Kent State University
http://etna.math.kent.edu

where $r_0 = b - Ax_0$ is the initial residual vector and $\mathcal{K}_j(B, r_0)$ is the j-th Krylov subspace defined as

$$K_j(B, r_0) = \text{span}\{r_0, Br_0, \ldots, B^{j-1}r_0\}.$$

The goal behind GMRES is to minimize at each step the Euclidean norm of the residual, i.e.,

$$\|b - Ax_j\| = \min_{u \in x_0 + M^{-1}\mathcal{K}_j(B,r_0)} \|b - Au\|. \tag{1.3}$$

An orthonormal basis $V_{j+1} = [v_0, \ldots, v_j]$[1] of $\mathcal{K}_{j+1}(B, r_0)$ is generated such that

$$v_0 = \frac{r_0}{\beta}, \quad \beta = \|r_0\|, \quad BV_j = V_{j+1}H_{j+1,j} = V_jH_j + h_{\{j+1,j\}}v_je_j^T. \tag{1.4}$$

It can be shown [40] that (1.3) reduces to

$$\|\beta e_1 - H_{j+1,j}y_j\| = \min_{y \in \mathbb{R}^j} \|\beta e_1 - H_{j+1,j}y\|, \tag{1.5}$$

and the approximate solution x_j can be written as

$$x_j = x_0 + M^{-1}V_jy_j.$$

Our work combines two improvements of this method. In GMRES(m), the method restarts at some step m to reduce storage and computational requirements as the iteration proceeds. The deflated and augmented approaches [5, 10, 22, 27, 31] keep some useful information at the time of the restart to enhance robustness. We briefly review these methods in Section 2. The second improvement builds the orthonormal basis with a parallel algorithm that reduces the number of exchanged MPI messages on distributed-memory computers. Indeed, the time to exchange a MPI message of size n words is given by $\gamma + \delta \cdot n$, where γ is the network latency (in seconds) and δ is the inverse bandwidth (in seconds per word). Hence, reducing the number of MPI messages is equivalent to minimizing the overhead due to this latency. From the original formulation of GMRES, the basis V is built and orthogonalized by the modified Gram-Schmidt implementation (MGS) of the Arnoldi process (V is referred to as *Arnoldi basis*). This process induces a high communication overhead due to the numerous inner products: for instance, a GMRES cycle of m iterations requires approximately $\frac{1}{2}(m^2 + 3m)$ global communications for the inner products. On high latency networks, the start-up time due to these collective communication can easily dominate. Moreover, the kernel operations in MGS have very low granularity, such that they do not fully utilize the computer architecture. In the classical Gram-Schmidt implementation (CGS) of the Arnoldi process, the communication time can be reduced by accumulating and broadcasting multiple inner products together. However, the low granularity of the kernel operations in the orthogonalization procedure remains because of the sequential form of the Arnoldi process. Moreover, CGS is more sensitive to rounding errors than MGS [25]. Alternative implementations [6, 15, 19, 20, 24, 26, 37, 42] have been proposed. They divide the process into two main phases: first, a nonorthogonal basis of the Krylov subspace is generated and then orthogonalized as a group of vectors in the second phase. As first proposed by Bai, Hu, and Reichel [6], the *a priori* basis is built with the aid of Newton polynomials. We will refer to this as the *Newton-basis GMRES*. Later on, the orthogonalization is done by replacing the vector-vector operations of the MGS method by the task of computing a dense

[1] Throughout this paper, we use a zero-based numbering for all the vectors in the basis.

ETNA
Kent State University
http://etna.math.kent.edu

QR factorization. De Sturler [15] analyzed the parallel implementation of the second phase and suggested a distributed MGS factorization to overlap communication with computation. Sidje and Philippe [43], Erhel [19], and Sidje [42] used a different orthogonalization strategy called RODDEC, which combines Householder factorization with Givens rotations and requires only point-to-point communication. Demmel et al. [16] proposed a different QR factorization called *Tall Skinny* QR (TSQR), which reorganizes the computation to reduce the memory access and exploit the data locality.

Our proposal in this work is to combine the Newton-basis GMRES with augmented and deflated GMRES. The new approach is simple and can be used together with any of the previous orthogonalization strategies once the augmented a priori basis is built. The motivation for our work is two-fold: previous studies [37] have shown that when the size of the Newton basis grows, the vectors become increasingly dependent. As a result, the method may experience a slow convergence rate. With the new approach, the basis is kept small and augmented with some useful approximate eigenvectors. The second motivation is related to GMRES(m) preconditioned by domain decomposition methods. Indeed, with Schwarz-based preconditioners, when the number of subdomains increases, the preconditioner becomes less and less robust and the method requires more iterations to converge. In this situation, the basis size is usually increased to prevent stagnation. In the proposed approach, we show that by adaptively adding more approximate eigenvectors, the convergence rate is improved. Recently, Mohiyuddin et al. [30] and Hoemmen [24] suggested a new formulation in their *communication avoiding* GMRES, which does not require the Krylov basis size to be equal to the number of vectors generated a priori. Their formulation builds the Krylov basis with several steps of the Arnoldi process where each step builds a set of vectors with the Newton polynomials. Our proposed approach can be used as well with their formulation for problems that are very sensitive to the restarting procedure in GMRES. Indeed, we show in Section 3 how the augmented basis can be formulated in their approach. The remaining part of this paper is organized as follows: in Section 2, we briefly review how the deflation of eigenvalues is used in the restarted GMRES. In Section 3, we derive the new approach combining deflation of the Newton basis with GMRES, and we discuss parallel implementation. Section 4 is focused on numerical experiments to show the benefits of our approach.

2. Restarted GMRES accelerated by deflation. A practical implementation of GMRES is based on restarting a minimum residual iteration when the correction space reaches a given dimension m. At the time of restart, information from the previous Krylov subspace is discarded and the orthogonality between successive Krylov subspaces is not preserved. The worst case is when successively generated Krylov subspaces are very close. As a result, there is no significant reduction in the residual norm and the iterative process stagnates. Deflation techniques are a class of acceleration strategies that collect useful information at the time of restart mainly to avoid stagnation and for improving the rate of convergence. The main idea behind these methods is to remove the smallest eigencomponents from the residual vector as they are known to slow down the convergence of GMRES [48]. For a general analysis of acceleration strategies for minimal residual methods, we refer the reader to Eiermann, Ernst, and Schneider [17]. For general Krylov subspace methods, the recent reviews in [21, 45] are also of great interest.

In deflation techniques, the Krylov subspaces are enriched by some approximation of invariant subspaces associated to a selected group of eigenvalues (generally the smallest ones). Two strategies are often used, namely by preconditioning the linear system [10, 19, 27] or by augmenting the Krylov subspace [31, 32].

Let $\lambda_1, \lambda_2, \ldots, \lambda_n$ be the eigenvalues of B. Consider an exact invariant subspace of dimension r corresponding to the r smallest eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_r$ and an orthonormal

ETNA
Kent State University
http://etna.math.kent.edu

basis of this subspace, written as U in matrix form. Using the preconditioner $\bar{M}^{-1}$ defined as

$$\bar{M}^{-1} = I_n + U(|\lambda_n|T^{-1} - I_r)U^T, \quad T = U^T BU, \tag{2.1}$$

the eigenvalues of $B\bar{M}^{-1}$ are $\lambda_{r+1}, \ldots, \lambda_n, |\lambda_n|$ with multiplicity at least r [22]. Then, GMRES(m) applied to $B\bar{M}^{-1}$ converges faster since the smallest eigencomponents that slow down the convergence are deflated. In practice, U is replaced by a basis of an approximate invariant subspace.

The actual implementation in [22] and the improvements in [10] rely on the approximation of U which is updated at each restart by computing the Ritz values from the Arnoldi relation in equation (1.4) to yield a more accurate basis. In [5], U is computed by the Implicit-Restarted Arnoldi (IRA) process and the result is used to form a left preconditioner. The adaptive preconditioner by Kharchenko and Yeremin [27] is built such that the Ritz values which approximate the largest eigenvalues of B are translated to a cluster around one.

Complexity overhead is similar in the preconditioning and the augmented subspace approaches. In both cases, computing a new basis at each restart involves a small eigenvalue problem. Also, it is easy to update the preconditioning matrix with a new basis U [10]. However, applying the preconditioner $\bar{M}^{-1}$ in (2.1) may induce an overhead for a large number of processors because of global MPI reduction operations. On the other hand, in the augmented approach presented below [31, 32], the communication is reduced since there is no global reduction operation.

The augmented approaches form the new approximation with a projection onto a subspace $\mathcal{C} = \mathcal{K}_m(B, r_0) + \mathcal{W}$, where $\mathcal{W} = \text{span}\{u_0, \ldots, u_{r-1}\}$. Minimal changes are required to the existing kernel operations as the vectors are directly added to the existing Krylov basis. Moreover, when the vectors $u_0, \ldots, u_{r-1}$ are the harmonic Ritz vectors, Morgan [32] has shown that the augmented subspace $\mathcal{C}$ is itself a Krylov subspace and writes

$$\mathcal{C} = \mathcal{K}_m(B, r_0) + \mathcal{W} = \mathcal{K}_{m+r}(B, q_m(B)r_0),$$

where $q_m(B)$ is a polynomial of degree at most m.

3. Deflated GMRES in the Newton basis. In this section, we derive the new implementation of the GMRES algorithm where the Krylov subspace is spanned by the Newton polynomials and augmented with eigenvectors. Compared to the previous Newton basis implementations [6, 19, 24, 42], the main difference is that the new approach uses deflation strategies to recover the information that is lost at the time of restart. Hence for the problems that are sensitive to the restarting procedure, our implementation should converge faster than the previous approaches for the same basis size. Compared to GMRES-E by Morgan [31], our approach communicates less and should produce kernels that are better suited for parallel computations. Compared to GMRESDR of Morgan [32], however, we have not investigated whether the proposed augmented basis is itself a Krylov basis, and we leave it to future work.

3.1. Augmenting the Newton basis. We now derive the proposed approach. Our motivation is to get an Arnoldi-like relation for the augmented basis when the eigenvectors are added at the end of the Newton basis. Let B be the preconditioned matrix, x_0 an initial guess, and r_0 the initial residual vector. An m-dimensional Krylov subspace is spanned by the Newton polynomials applied to r_0 of the form

$$\mathcal{P}_{j+1}(B) := \sigma_{j+1}(B - \lambda_{j+1}I)\mathcal{P}_j(B), \quad j = 0, 1, 2, \ldots,$$

where $\mathcal{P}_0(B) := r_0$, σ_j and $\lambda_j \in \mathbb{R}$; see [6, 37]. We will discuss the choice of these scalars later.

ETNA
Kent State University
http://etna.math.kent.edu

Let $\mathcal{W} = \operatorname{span}\{u_0, u_1, \ldots, u_{r-1}\}$ be a coarse subspace of dimension r. We discuss the computation of these vectors in Section 3.2.5. Let $s = m + r$ and $\mathcal{C}_s$ be the augmented subspace defined by

$$\mathcal{C}_s = \mathcal{K}_m(B, r_0) + \mathcal{W}$$

as before. We write the basis of $\mathcal{W}$ in matrix form as

$$U_r = \begin{bmatrix} u_0 & \ldots & u_{r-1} \end{bmatrix}.$$

We also define the augmented subspace $\hat{\mathcal{C}}_{s+1}$ by

$$\hat{\mathcal{C}}_{s+1} = \mathcal{K}_{m+1}(B, r_0) + B\mathcal{W}.$$

PROPOSITION 3.1. *Suppose that* $\dim(\hat{\mathcal{C}}_{s+1}) = s + 1$. *Then there exists an orthonormal basis* $V_{s+1} = \begin{bmatrix} V_{m+1} & \hat{V}_r \end{bmatrix} \in \mathbb{R}^{n\times(s+1)}$ *of the subspace* $\hat{\mathcal{C}}_{s+1}$ *such that* V_{m+1} *is an orthonormal basis of* $\mathcal{K}_{m+1}(B, r_0)$ *and*

$$W_s = \begin{bmatrix} V_m & U_r \end{bmatrix}$$

is a basis of $\mathcal{C}_s$. *These bases are related by an Arnoldi-like relation*

$$BW_s = V_{s+1}\bar{H}_s = V_s H_s + h_{s+1,s} v_{s+1} e_s^T, \tag{3.1}$$

where $\bar{H}_s \in \mathbb{R}^{(s+1)\times s}$ *is an upper Hessenberg matrix. Moreover, the vector* $x_s \in \mathbb{R}^n$ *given by*

$$x_s = x_0 + M^{-1} W_s y_s,$$

where $y_s \in \mathbb{R}^n$ *solves the least-squares problem* $J_s(y)$ *defined by*

$$J_s(y) = \|\beta e_1 - \bar{H}_s y\|_2, \quad \beta = \|r_0\|_2,$$

minimizes the residual norm $\|b - Ax_s\|$ *over* $x_0 + M^{-1}\mathcal{C}_s$.

Proof. From $k_0 = r_0/\|r_0\|_2$, a set of vectors k_j can be generated such that

$$\sigma_{j+1} k_{j+1} = \begin{cases} (B - \lambda_{j+1} I) k_j & \text{if} \quad 0 \le j \le m-1 \\ B u_{j-m} & \text{if} \quad m \le j \le s-1, \end{cases} \tag{3.2}$$

where λ_j and σ_j, $j = 1, 2, \ldots$, are user-specified real scalars. We discuss in Sections 3.2.1 and 3.2.2 their optimal choice. In matrix form, relation (3.2) writes

$$\begin{aligned} BK_m &= K_{m+1}\bar{T}_m \\ BU_r &= \hat{K}_r D_r, \end{aligned} \tag{3.3}$$

with $K_{m+1} = \begin{bmatrix} k_0 & k_1 & \ldots & k_m \end{bmatrix}$, $\hat{K}_r = \begin{bmatrix} k_{m+1} & \ldots & k_s \end{bmatrix}$,

$$\bar{T}_m = \begin{bmatrix} \lambda_1 & & & & & \\ \sigma_1 & \lambda_2 & & & & \\ & \sigma_2 & \lambda_3 & & & \\ & & \ddots & \ddots & & \\ & & & & \lambda_{m-1} & \\ & & & & \sigma_{m-1} & \lambda_m \\ & & & & & \sigma_m \end{bmatrix} \in \mathbb{R}^{(m+1)\times m},$$

ETNA
Kent State University
http://etna.math.kent.edu

and $D_r = \text{diag}\{\sigma_{m+1}, \ldots, \sigma_s\} \in \mathbb{R}^{r\times r}$. A QR factorization of $\begin{bmatrix} K_{m+1} & \hat{K}_r \end{bmatrix}$ yields

$$\begin{bmatrix} K_{m+1} & \hat{K}_r \end{bmatrix} = V_{s+1}R_{s+1} = \begin{bmatrix} V_{m+1} & \hat{V}_r \end{bmatrix} \begin{bmatrix} R_{m+1} & R_{m+1,r} \\ 0 & R_r \end{bmatrix}. \tag{3.4}$$

Using equations (3.3) and (3.4), we get

$$BV_m = BK_mR_m^{-1} = V_{m+1}R_{m+1}\bar{T}_mR_m^{-1},$$
$$BU_r = \hat{K}_rD_r = (V_{m+1}R_{m+1,r} + \hat{V}_rR_r)D_r.$$

It is rewritten as

$$BW_s = V_{s+1}\bar{H}_s,$$

where

$$\bar{H}_s = R_{s+1}\begin{bmatrix} \bar{T}_m & 0 \\ 0 & D_r \end{bmatrix}\begin{bmatrix} R_m^{-1} & 0 \\ 0 & I_r \end{bmatrix} = \begin{bmatrix} H_s \\ h_{s+1,s}e_s^T \end{bmatrix}. \tag{3.5}$$

The first part (3.1) is thus proved.

The second part of the proposition is similar to the optimality property of the augmented GMRES [12, Algorithm 2.1]. With $x_s = x_0 + M^{-1}W_sy$ in the affine space $x_0 + M^{-1}\mathcal{C}_s$, the corresponding residual vector is expressed as

$$\begin{aligned} b - Ax_s &= b - A(x_0 + M^{-1}W_sy) = r_0 - V_{s+1}\bar{H}_sy = \beta k_0 - V_{s+1}\bar{H}_sy \\ &= V_{s+1}(\beta e_1 - \bar{H}_sy), \end{aligned} \tag{3.6}$$

where $\beta = \|r_0\|$ and $e_1 = [1, 0, \ldots, 0]^T$. Introduce the function

$$J_s(y) = \|b - A(x_0 + M^{-1}W_sy)\|_2.$$

It follows from equation (3.6) and the fact that V_{s+1} is orthogonal that

$$J_s(y) = \|\beta e_1 - \bar{H}_sy\|_2.$$

Thus by taking the vector $y_s \in \mathbb{R}^m$ which minimizes the function $J_s(y)$, the approximate solution $x_s = x_0 + M^{-1}W_sy_s$ will have the smallest residual in $x_0 + M^{-1}\mathcal{C}_s$. □

We refer to the matrix W_s as the *augmented Newton basis* of the subspace $\mathcal{C}_s$, and to the matrix V_{s+1} as the *augmented orthonormal basis* of the subspace $\hat{\mathcal{C}}_{s+1}$. The induced GMRES is the *augmented Newton basis GMRES*, which we denote by AGMRES.

It is also possible to choose $\begin{bmatrix} K_m & U_r \end{bmatrix}$ as a basis of $\mathcal{C}_s$, avoiding the computation of R_m^{-1}. However, the vectors U_r are computed at each restart by using a Rayleigh-Ritz procedure (see Section 3.2.5), which requires the storage of V_{s+1} and the computation of $V_{s+1}^TW_s$. Thus, it saves memory and computation to select $W_s = \begin{bmatrix} V_m & U_r \end{bmatrix}$.

This proof assumes that the basis vectors $k_j, j = 0, \ldots, s$, are generated through one pass in the kernel computation of (3.2). There are some situations where m is too large to guarantee robustness (well-conditioned basis) or good performance (best value for data locality in multicore nodes). In recent work, Hoemmen [24] uses the μ-step Arnoldi method of Kim and Chronopoulos [28][2] in his *Arnoldi(μ, t)* to build the basis vectors through multiple

[2]The original method is referred to as s-step Arnoldi instead of μ-step Arnoldi but we choose μ here to differentiate with the size s of our augmented basis.

ETNA
Kent State University
http://etna.math.kent.edu

passes of the kernel computation in (3.2). Hence, the process of computing s basis vectors is divided into t steps where each step generates μ basis vectors with the Newton polynomials. The restart length is thus $s = \mu \cdot t$. We show in the following that Proposition 3.1 holds in the case of a μ-step basis. We explain the basic idea with $t = 2$.

Let k_0 be the starting vector and $s = \mu \cdot t + r = m + r$. As in the first part of (3.2), we generate the sequence of vectors

$$\sigma_{j+1} k_{j+1} = (B - \lambda_{j+1} I) k_j, \quad 0 \leq j \leq \mu - 1.$$

It follows that

$$B K_\mu^{(0)} = K_{\mu+1}^{(0)} \bar{T}_\mu^{(0)},$$

where $K_{\mu+1}^{(0)} = \begin{bmatrix} k_0 & k_1 & \dots & k_\mu \end{bmatrix} \in \mathbb{R}^{n \times (\mu+1)}$ and $\bar{T}_\mu^{(0)} \in \mathbb{R}^{(\mu+1) \times \mu}$ is a bidiagonal matrix. A QR factorization of $K_{\mu+1}^{(0)}$ gives

$$K_{\mu+1}^{(0)} = V_{\mu+1}^{(0)} R_{\mu+1}^{(0)}$$

and thus

$$B K_\mu^{(0)} = V_{\mu+1}^{(0)} \bar{H}_\mu^{(0)} = V_\mu^{(0)} H_\mu^{(0)} + h_{\mu+1,\mu} V_{\mu+1}^{(0)} e_\mu e_\mu^T, \tag{3.7}$$

where $\bar{H}_\mu^{(0)} = R_{\mu+1}^{(0)} \bar{T}_\mu^{(0)}$ and e_μ is the μ-th unit vector. This first step is just the derivation of the Arnoldi-like relation for the (non-augmented) Newton basis. Note that we do not have a mathematically equivalent Arnoldi relation as in equation (1.4). $\bar{H}_{\mu+1}^{(0)}$ is not equal in exact arithmetic to the Hessenberg matrix $\bar{H}$ of that equation as we avoid dealing with the term $(R_{\mu+1}^{(0)})^{-1}$. However, the columns of $V_{\mu+1}^{(0)}$ form an orthogonal basis of a μ-dimensional Krylov basis. Using the last column as a starting vector, we can thus build the second μ-step basis. At this step, we add the eigenvectors in the subspace by augmenting the μ-step basis.

Let $k_\mu = V_{\mu+1}^{(0)} e_\mu$ be a μ-step augmented basis generated as follows

$$\sigma_{j+1} k_{j+1} = \begin{cases} (B - \lambda_{\mu-j+1} I) k_j & \text{if } \mu \leq j \leq m-1 \\ B u_{j-m} & \text{if } m \leq j \leq s-1. \end{cases}$$

In matrix form we get

$$B \begin{bmatrix} K_\mu^{(1)} & U_r \end{bmatrix} = K_{\mu+r+1}^{(1)} \begin{bmatrix} \bar{T}_\mu^{(1)} & 0 \\ 0 & D_r \end{bmatrix}. \tag{3.8}$$

The matrices $\bar{T}_\mu^{(1)}$ and D_r are analogous to the matrices in equation (3.1). At this point, to avoid loss of orthogonality, the vectors $K_{\mu+r+1}^{(1)} = \begin{bmatrix} k_\mu & k_{\mu+1} & \dots & k_{\mu+m+r} \end{bmatrix}$ should be orthogonalized against the previous vectors $V_\mu^{(0)}$. This can be done by a block Gram-Schmidt method which is equivalent to writing[3]

$$\hat{K}_{\mu+r+1}^{(1)} = \left(I - V_\mu^{(0)} (V_\mu^{(0)})^T \right) K_{\mu+r+1}^{(1)}. \tag{3.9}$$

Note that the same stability problems may arise just as in the classical Gram-Schmidt process. We discuss this issue at the end of this subsection. So far, the vectors $\hat{K}_{\mu+r}^{(1)}$ are orthogonal to

[3] Note that the first vector k_μ is already orthogonal to $V_\mu^{(0)}$, but we choose to orthogonalize it again.

ETNA
Kent State University
http://etna.math.kent.edu

the basis vectors, but it remains to orthogonalize them between each other. This can be done by a dense QR factorization to produce

$$\hat{K}^{(1)}_{\mu+r+1} = V^{(1)}_{\mu+r+1} R^{(1)}_{\mu+r+1}. \tag{3.10}$$

From equations (3.7) and (3.8) we get

$$B\begin{bmatrix} K^{(0)}_{\mu} & K^{(1)}_{\mu} & U_r \end{bmatrix} = \begin{bmatrix} V^{(0)}_{\mu} & K^{(1)}_{\mu+r+1} \end{bmatrix} \begin{bmatrix} H^{(0)}_{\mu} & 0 \\ h_{\mu+1,\mu} e_1 e_{\mu}^T & \bar{C}_{\mu+r} \end{bmatrix}, \tag{3.11}$$

where

$$\bar{C}_{\mu+r} = \begin{bmatrix} \bar{T}^{(1)}_{\mu} & 0 \\ 0 & D_r \end{bmatrix}.$$

Knowing that a QR factorization update has been performed on $K^{(1)}_{\mu+r+1}$, we get from equations (3.9) and (3.10) that,

$$\begin{bmatrix} V^{(0)}_{\mu} & K^{(1)}_{\mu+r+1} \end{bmatrix} = \begin{bmatrix} V^{(0)}_{\mu} & V^{(1)}_{\mu+r+1} \end{bmatrix} \begin{bmatrix} I & (V^{(0)}_{\mu})^T K^{(1)}_{\mu+r+1} \\ 0 & R^{(1)}_{\mu+r+1} \end{bmatrix}. \tag{3.12}$$

Substituting (3.12) in (3.11), we find

$$BW_s = V_{s+1} \bar{H}_s,$$

where $W_s = \begin{bmatrix} K^{(0)}_{\mu} & K^{(1)}_{\mu} & U_r \end{bmatrix}$, $V_{s+1} = \begin{bmatrix} V^{(0)}_{\mu} & V^{(1)}_{\mu+r} \end{bmatrix}$, and

$$\bar{H}_s = \begin{bmatrix} I_{\mu,\mu} & (V^{(0)}_{\mu})^T K^{(1)}_{\mu+r+1} \\ 0 & R^{(1)}_{\mu+r+1} \end{bmatrix} \begin{bmatrix} H^{(0)}_{\mu} & 0 \\ h_{\mu+1,\mu} e_1 e_{\mu}^T & \bar{C}_{\mu+r} \end{bmatrix}.$$

From the fact that $K^{(1)}_{\mu+r+1} e_1$ is orthogonal to $V^{(0)}_{\mu}$ and that $R^{(1)}_{\mu+r+1} e_1 = e_1$, we obtain

$$\bar{H}_s = \begin{bmatrix} H^{(0)}_{\mu} & (V^{(0)}_{\mu})^T K^{(1)}_{\mu+r+1} \bar{C}_{\mu+r} \\ h_{\mu+1,\mu} e_1 e_{\mu}^T & R^{(1)}_{\mu+r+1} \bar{C}_{\mu+r} \end{bmatrix},$$

which is a Hessenberg matrix. The first part of Proposition 3.1 is thus proved and the second part is similar to the previous proof.

In GMRES with the μ-step Newton basis, it is useful to control the conditioning of the basis generated with the Newton polynomials by choosing a suitable value of μ. On multicore nodes, a well-chosen value of μ will also improve the data locality during the computation of the kernel computations (generation of the basis and orthogonalization) [16, 24]. The drawback of this formulation is that when the new set of basis vectors is orthogonalized against all the previous vectors already computed, it is important to perform a QR factorization update. Sometimes when a block Gram-Schmidt process is used, a reorthogonalization strategy should be performed to avoid a loss of orthogonality; see, for instance, [25]. This process induces more computational cost as the number of steps t increases. As for the scalar formulation, the augmented basis will thus help to reduce this cost by reducing the number of steps t. We do not further investigate reorthogonalization, and we focus in this paper on the basic implementation of a $(\mu + r)$-step augmented Newton basis.

ETNA
Kent State University
http://etna.math.kent.edu

Algorithm 1 AGMRES(m, r): Augmented Newton-basis GMRES.

Require: $x_0, m, itmax, \epsilon, r, l, rmax, smv, bgv$;

1: **Perform** one cycle of GMRES(m) [41, Algorithm 4] to find a new approximation x_m, the residual r_m, and the matrices H_m and V_m satisfying equation (1.4).
if($\|r_m\| < \epsilon$) **return**
2: Set $x_0 \leftarrow x_m$ and $r_0 \leftarrow r_m$; $\beta = \|r_0\|$;
3: **Compute** m Ritz values $\{\lambda_j\}_{j=1}^{j=m}$ of AM^{-1} from H_m and order them with the Leja ordering [6].
4: **If** $(r > 0)$ extract r Ritz vectors U_r for the augmented basis.
5: **while** $(\|r_0\| > \epsilon)$ **do**
6: **Compute** the basis vectors of equation (3.2) [19, 42] to get K_{m+1} and $\bar{T}_m$.
7: **Compute** $\hat{K}_r$ and D_r using the second part of equation (3.2).
8: **Compute** the QR factorization $\begin{bmatrix} K_{m+1} & \hat{K}_r \end{bmatrix} = V_{s+1} R_{s+1}$.
9: **Derive** $W_s = \begin{bmatrix} V_m & U_r \end{bmatrix}$.
10: **Compute** the $(s+1) \times s$ Hessenberg matrix $\bar{H}_s$ from equation (3.5).
11: **Solve** $y_s = \min \|\beta e_1 - \bar{H}_s y\|_2$.
12: **Compute** $x_s = x_0 + M^{-1} W_s y_s$, $r_s = b - A x_s$, $it \leftarrow it + s$.
13: **if**($\|r_s\| < \epsilon$ **or** $it > itmax$) **return**
14: Set $x_0 \leftarrow x_s$ and $r_0 \leftarrow r_s$;
15: **if** $r > 0$ **then**
16: $Iter = s * \log\left(\frac{\epsilon}{\|r_s\|}\right) / \log\left(\frac{\|r_s\|}{\|r_0\|}\right)$
17: **if** $(Iter > smv * (itmax - it))$ **then**
18: **if** $((Iter > bgv * (itmax - it))$ **and** $(r < rmax)$ **and** $(l > 0))$ **then**
19: $r \leftarrow r + l$ /*Increase the number of eigenvalues to deflate*/
20: **end if**
21: **Replace** the r approximate eigenvectors $u_0, u_1, \ldots, u_{r-1}$ by r harmonic Ritz vectors of $B \equiv AM^{-1}$.
22: **end if**
23: **end if**
24: **end while**

3.2. AGMRES: Augmented Newton-basis GMRES. In this section we discuss the parallel implementation of the GMRES method where the Newton basis is augmented with a few approximate eigenvectors. The main steps are outlined in Algorithm 1.

If we compare AGMRES with the related implementations of the GMRES method, we can make the following observations.

- Compared to the standard GMRES method, AGMRES produces kernels that are better suited for parallel computations during the generation of the orthogonal Krylov basis in steps 6 and 8. However, in addition to the basis V_{m+1}, it keeps $2r$ vectors U_r and $\hat{V}_r$. It is thus worth mentioning that AGMRES(m, r) requires as much memory as GMRES$(m + 2r)$. Nevertheless, our numerical experiments indicate that, on most test cases, AGMRES(m, r) produces a better convergence acceleration than GMRES$(m + 2r)$ and even GMRES$(2m)$.
- The GMRES-E of Morgan [31] keeps a second basis as well. However its implementation is based on the Arnoldi process. It will thus communicate more for the same convergence behavior. Our implementation includes an adaptive strategy that allows us to increase the number of extracted eigenvectors if necessary.
- Compared to CA-GMRES of Hoemmen [24], our implementation is limited to one μ-step Newton basis. However, in the previous section we have shown how an aug-

ETNA
Kent State University
http://etna.math.kent.edu

mented basis can be defined for more than one μ-step basis. For the same restart length, CA-GMRES(μ, t) and GMRES($\mu \cdot t$) produce the same convergence behavior. AGMRES($\mu \cdot t, r$) is more likely to produce a faster convergence than these two approaches when the convergence rate is affected by the restarting procedure.

So far, the algorithm starts with an initial approximation of the solution vector x_0 (in practice, we use a zero vector), the size m of the Krylov basis, the maximum number of iterations $itmax$ allowed, and the desired accuracy ϵ. The remaining input values are used for the augmented basis: the number of eigenvectors r that are added at each step and the parameters $l, rmax, smv,$ and bgv for the adaptive strategy; see Section 3.2.6. The main steps of the algorithm are the computation of the shifts (steps 1 and 3), the generation of the augmented Newton basis at step 6, and its orthogonalization at step 8. The approximate solution is updated at step 12. At step 21, the Ritz vectors are recomputed in order to add them to the Newton basis. The adaptive strategy is implemented in steps 15–23. All these steps are explained in the next sections.

3.2.1. Computation of the shifts. The generation of the Krylov subspace with the Newton polynomials uses the scalars λ_j, $j = 1, \ldots, m$ to produce a stable basis. Bai et al. [6] have shown that a good choice would be to use the eigenvalues of B numbered according to the following modified *Leja order* (see [38])

$$|\lambda_1| = \max_{j=1,\ldots,m} |\lambda_j|$$
$$\prod_{k=1}^{j} |\lambda_{j+1} - \lambda_k| = \max_{l=1,\ldots,m} \prod_{k=1}^{j} |\lambda_l - \lambda_k|, \quad j = 1, \ldots, m-1.$$

In practice, the spectrum of B is not available and very expensive to compute. In this situation, the Ritz values of B, which are the eigenvalues of the Hessenberg matrix H_m in equation (1.4), are used. This implies that m steps of the Arnoldi process should be performed to find these values. At step 1, we perform one cycle of the Arnoldi-GMRES method. From this we get an approximation of the solution x_m and the associated residual r_m. This vector is used as the initial search direction for the Newton-basis GMRES in step 5. At step 3, each process computes the eigenvalues of its own copy of the Hessenberg matrix H_m and orders them with the Leja ordering. This step so far uses the parallelism inside the matrix-vector product and the preconditioning operation. But it requires global communication as pointed out in Section 1. Note that when m gets large, it may be expensive to perform this Arnoldi-GMRES step . The cost here is comparable to one step of the Newton basis GMRES mainly in terms of granularity and the volume of MPI messages. In practice, we use a small value of m to show the benefits of augmenting the basis. Nevertheless, if a large basis is used, a solution could be to employ a μ-step basis as explained in the previous section. Another solution, as advised by Philippe and Reichel [37], is to perform one cycle of an Arnoldi-GMRES with a smaller basis to get a subset of these values. From this subset, a convex hull is defined and continuously updated with new values collected during the Newton-basis GMRES iterations.

3.2.2. Computation of the Newton basis with scaling factors. The first $m+1$ vectors of K_{s+1} can be generated using Algorithm 1.1 in [42]; then it is easy to generate the last r vectors from U_r. Note that when a particular eigenvalue λ_{j+1} is complex and assuming that $\mathrm{Im}(\lambda_{j+1}) > 0$ (such a case always exists by considering the complex conjugate pairs of eigenvalues and the modified Leja ordering), the complex arithmetic is avoided by writing

ETNA
Kent State University
http://etna.math.kent.edu

the first part of equation (3.2) as

$$\sigma_{j+1}k_{j+1} = (B - \mathrm{Re}(\lambda_{j+1}))Ik_j$$
$$\sigma_{j+2}k_{j+2} = (B - \lambda_{j+1}I)(B - \bar{\lambda}_{j+1}I)k_j = \sigma_{j+1}(B - \mathrm{Re}(\lambda_{j+1})I)k_{j+1} + \mathrm{Im}(\lambda_{j+1})^2 k_j.$$

In this case, the matrix $\bar{T} \in \mathbb{R}^{s+1\times s}$ in the equations (3.1) and (3.8) is tridiagonal. So far, the scalars $\sigma_j, j = 1, \ldots, m$ in (3.1) are used to control the growth of the vectors $\{k_j\}_{j=1}^{j=m}$. The common choice is to take $\sigma_j = \|k_j\|$. The parallelism inside this step is kept by the preconditioning and the parallel matrix-vector operations $(AM^{-1} - \lambda I)k \equiv A(M^{-1}k) - \lambda k$. When $\sigma_j = \|k_j\|$, then there are $(m + r)$ global communications, which are far fewer than the $\frac{1}{2}(m^2 + 3m)$ global communications in the Arnoldi process. For some particular cases, this norm can be computed distributively. When using for instance the explicit formulation of the multiplicative Schwarz procedure, the basis vectors are computed in a pipeline across all the subdomains. Each process is thus able to compute its own contribution to the norm and the basis vectors are normalized *a posteriori* [4, 33]. When the size of the basis is small enough, the rows and columns of the matrix can be equilibrated and no scaling, thus no global communication, is needed during the computation of the basis vectors [24].

3.2.3. Orthogonalization of the basis. After the basis vectors have been computed, they should be orthogonalized amongst each other at step 8 of Algorithm 1 to produce the orthogonal system V_{s+1}. At the end of step 6, the vectors K_{s+1} are distributed on all processors as a contiguous block of rows which is equivalent to the classical 1D row-wise partitioning for the matrix-vector products. Any algorithm for parallel dense QR factorization can now be used to orthogonalize the system K_{s+1}. In our implementation, we use the RODDEC algorithm described in [42, Section 4.2]. This method performs first a Householder orthogonalization on each block of rows. This is done in a perfect parallel phase by all the processes having the rows. After that, Givens rotations are used to annihilate the blocks below the first one. During this second step, the processors are placed on a ring topology and each process sends the required data on this ring. This step requires $\mathcal{O}(m^2)$ point-to-point messages on a ring topology, and the average message length is $\frac{1}{2}(m + 1)$ double precision numbers. The approach advocated in [13] maps the processors on a binary tree to eliminate the extra triangular factors of RODDEC. The TSQR algorithm of Demmel et al. [16], which gives a more general divide-and-conquer algorithm, can be used as well at this step. It requires $\mathcal{O}(\log(P))$ MPI messages, where P is the total number of MPI processes sharing the system K_{s+1}.

3.2.4. Updating the current approximation. At the end of the QR factorization, the triangular matrix R_s of equation (3.4) is usually available on one process. In the RODDEC algorithm, it is available in the last process. It can be broadcasted to all other processes such that the steps 10 and 11 are done by all the processes. When the number of MPI processes gets large, it is more efficient to perform these steps on the last process and to broadcast only the result of the least-squares problem at step 11. In our implementation, we choose to send a copy of the matrix since it is required by all processes to update the eigenvectors; see Section 3.2.5. So far, the Hessenberg matrix $\bar{H}_s$ is assembled from R_s and $\bar{T}_s$ using a modification of Algorithm 1.2 in [42]. The modification allows us to take into account the scaling factors of the augmented vectors in the basis. A QR factorization is performed on the output Hessenberg matrix to solve the least-squares problem in the minimization step. The LAPACK routine *dgeqrf* is applied for this purpose. The output solution is used to compute the new approximate solution at step 12. Note that since we are using right preconditioning, we can obtain an estimate of the true residual norm without explicitly computing the residual vector r_s. Nevertheless, at the time of restart, we need r_s for the new search direction.

ETNA
Kent State University
http://etna.math.kent.edu

3.2.5. Updating the eigenvectors. When the iterative process starts at line 16 of Algorithm 1, the eigenpairs (u_j, λ_j) of $B \equiv AM^{-1}$ are approximated using the first GMRES(m) cycle with a standard projection technique as follows,

$$V_m^T(B - \lambda_j I)V_m g_j = 0,$$

leading to the eigenvalue problem

$$H_m g_j = \lambda_j g_j.$$

The Ritz values $\lambda_j, j = 0, \ldots, m-1$, are used as shifts for the Newton basis and the vectors corresponding to the r smallest eigenvalues $u_j = V_m g_j, j = 0, \ldots, r-1$, are used to augment the Newton basis.

Then in order to change the vectors U_r at step 21, we use a Rayleigh-Ritz procedure. Indeed, as advised by previous studies [12, 31], this procedure does better at finding eigenvalues near zero. Using the augmented subspace $\mathcal{C}_s$, each extracted approximate eigenvector u is expressed as $u = W_s g_i$. Using BW_s and W_s, the Galerkin condition becomes

$$(BW_s)^T(B - \lambda_j I)W_s g_j = 0.$$

It follows from relation (3.1) that

$$\underbrace{\bar{H}_s^T \bar{H}_s}_{G_s} g_j = \lambda_j \underbrace{\bar{H}_s^T V_{s+1}^T W_s}_{F_s} g_j.$$

We thus obtain a dense generalized eigenvalue problem of size $s \times s$ where $(\lambda_j, W_s g_j)$ is a harmonic Ritz pair of B. Multiplying F_s and G_s by H_s^{-T}, we get

$$\begin{aligned} H_s^{-T} G_s &= H_s^{-T} \begin{bmatrix} H_s^T & \alpha e_s \end{bmatrix} \begin{bmatrix} H_s \\ \alpha e_m^T \end{bmatrix} \\ &= H_s + h_{s+1,s}^2 H_s^{-T} e_s e_s^T, \\ H_s^{-T} F_s &= \begin{bmatrix} I_s & h_{s+1,s} H_s^{-T} e_s \end{bmatrix} V_{s+1}^T W_s, \end{aligned}$$

with $V_{s+1}^T W_s = \begin{bmatrix} V_{s+1}^T V_m & V_{s+1}^T U_r \end{bmatrix}$ and $V_{s+1}^T V_m = \begin{bmatrix} I_m \\ 0 \end{bmatrix}$.

The overhead here is small since it induces only r dot products of size n and a small generalized eigenvalue problem.

If the basis of $\mathcal{C}_s$ is equal to $\begin{bmatrix} K_m & U_r \end{bmatrix}$, then the overhead is larger since it involves s dot products. Nevertheless, in most test cases the numerical experiments show that when the convergence is accelerated by deflation, the time to update the approximate eigenvectors is negligible compared to the total time saved without the deflation. Moreover, the adaptive strategy proposed next sets off deflation only if the convergence is too slow.

3.2.6. Adaptive strategy. When the desired accuracy is not achieved, the method restarts and r new approximate eigenvectors (corresponding to the eigenvalues to deflate) are extracted from the s-dimensional subspace $\mathcal{C}_s$. This process may become expensive and is not beneficial if the convergence rate is not improved enough. We thus propose an adaptive strategy which detects if the deflation process will be beneficial to speed up the convergence or to avoid stagnation. This approach is based upon the work by Sosonkina et al. [47], which has been used successfully in another formulation of the deflated GMRES [35]. At line 16, based

ETNA
Kent State University
http://etna.math.kent.edu

on the convergence rate already achieved, we estimate the remaining number of steps ($Iter$) needed to reach the desired accuracy ϵ. We use a small multiple (smv) of the remaining number of steps to detect some insufficient reduction in the residual norm. If it is greater than a small multiple (smv) of the number of steps allowed ($itmax$), then we switch to the deflation. We use a large multiple (bgv) of $itmax$ to detect near-stagnation in the iterative process. In this case, the number of eigenvectors to augment is increased by a fixed (small) value. Clearly with the parameters $r, l, Iter, smv, bgv$, the adaptive strategy can be sketched as follows:

- If $Iter \leq smv * itmax$, then the convergence rate is good enough and no more update should be done on the eigenvectors already computed.
- If $smv * itmax < Iter \leq bgv * itmax$, then there is an insufficient reduction in the residual norm and the r eigenvectors are updated for the next cycles of AGMRES.
- If $Iter > bgv * itmax$, then stagnation may have occurred and we increase the number of eigenvalues to extract/update by a fixed number l. This can be changed at runtime by the user. Unless stated otherwise, we use $l = 1$ in all our test cases.

Note that there are more sophisticated methods to ensure that for some given values of m, GMRES(m) (and thus AGMRES(m)) will not stagnate; see, for instance, [44, 46]. Our current stagnation test is computed *a posteriori* and should be mostly used to detect a very slow reduction in the residual norm. Although the proposed parameters are problem dependent, they can be useful to avoid the stagnation if there is some previous knowledge about the convergence behavior for the problems under study. Some numerical results are given in this sense in the next section.

4. Numerical experiments. This section presents some numerical results to show the parallel efficiency and the numerical robustness of the proposed approach. We first present the template for all the numerical tests in Section 4.1 and the test cases in Section 4.2.

4.1. Test routines and implementation notes. Implementations are done using the PETSc routines and data structures [7, 8]. Algorithm 1 has been implemented by a *KSP* module called AGMRES using a locally modified version of PETSc revision 3.1.p8. It uses routines for matrix-vector product evaluation, the application of the preconditioner, and the other parallel linear algebra functions. Algorithm 1 can be used transparently with any preconditioner implemented in the package including the domain decomposition preconditioners. We use the Restricted Additive Schwarz (RAS) method [11] applied as a right preconditioner in all our tests. The main steps are outlined in Algorithm 2.

Algorithm 2 Test routine for the parallel computation of the system (1.2) using the Restricted Additive Schwarz method and the GMRES-based accelerator.

1: Read the matrix from a binary file and store it in a distributed CSR format. Read the right-hand side vector and store it accordingly.
2: Perform parallel iterative row and column scaling on the matrix and the right-hand side vector [3].
3: Partition the weighted graph of the matrix in parallel with PARMETIS.
4: Redistribute the matrix and right-hand-side according to the PARMETIS partitioning.
5: Define the overlap between the submatrices for the additive Schwarz preconditioner.
6: Set up the submatrices (ILU or LU factorization using MUMPS [2]).
7: Iteratively solve the system using either the KSP AGMRES (Algorithm 1) or the PETSc built-in KSP GMRES [41, Algorithm 4].
8: Write the solution vector to a binary file.

ETNA
Kent State University
http://etna.math.kent.edu

Note that at step 7 of our test routine, we compare AGMRES with the classical implementation of GMRES. As stated earlier, either classical Gram-Schmidt or modified Gram-Schmidt can be used for the Arnoldi process. The main advantage of CGS over MGS is the number of MPI messages, the amount of MPI reductions, and the granularity in the computational kernel. However, a practical implementation of CGS includes a possible refinement strategy to be as stable as MGS. During our numerical experiments, however, this refinement was not used in CGS, and we did not notice any difference between GMRES-MGS and GMRES-CGS. We therefore present the results of GMRES with CGS. Unless stated otherwise, the stopping criterion of GMRES and AGMRES is $\frac{\|b-Ax\|}{\|b\|} < 10^{-10}$ and the maximum number of iterations is 1000. In AGMRES, the residual norm is computed only at each outer iteration. In GMRES, it is available during each inner iteration. Note that since we are using a right preconditioner, this residual norm is obtained cheaply from the Givens rotations that are used to transform the Hessenberg matrix in (1.5) into a triangular matrix.

In the following, since right preconditioning is used, the number of iterations is understood as the total number of matrix-vectors products and preconditioning steps. Hence in GMRES(m), it is equivalent to the counts of $A(M^{-1}k)$. In AGMRES, it is equal to the size of the augmented basis times the restart cycles. So far, AGMRES(m) refers to the Algorithm 1 without the deflation (i.e., $r = 0, l = 0$). In AGMRES(m, r), r vectors corresponding to the smallest harmonic Ritz values are added to the basis and replaced at each restart. With the adaptive strategy, whenever it is necessary, r is increased by $l = 1$ (unless stated otherwise). In the numerical experiments, the basis of the augmented subspace $\mathcal{C}_s$ is equal to $\begin{bmatrix} K_m & U_r \end{bmatrix}$.

4.2. Test problems. The matrices of the test problems arise from industrial applications in fluid dynamics and from convection-diffusion problems. The main characteristics are listed in Table 4.1.

The problems IM07R, VV11R, and RM07R arise from design optimization in computational fluid dynamics simulations. They are provided by the FLUOREM company, a CFD software editor[4]. Table 4.1 lists the coefficient matrices with their main characteristics. The physical equations are the Reynolds-Averaged Navier-Stokes equations for compressible flows discretized using the finite volume methods as presented in [36]. The resulting matrix is comprised by $b \times b$ blocks, where b is the number of fluid conservative variables (density, velocity, energy, and turbulent variables). The matrix RM07R is available online in the University of Florida sparse matrix collection (see [14]) in the FLUOREM directory. The matrix is structurally symmetric in the blocks. Regarding the values, the matrix is nonsymmetric and indefinite. In [34, 36], preliminary studies have shown that hybrid solvers based on GMRES and Schwarz-based preconditioners offer robust approaches to solve these systems efficiently. As pointed out in [36], we avoid the ILU factorization in the subdomain matrices because of its unpredictable behavior. We therefore rely on a direct solver (MUMPS) within each subdomain.

The test cases 3DCONSKY_121 and 3DCONSKY_161 correspond to the *convective SkyScraper* problem in [1, 29]. The physical equation is given by the boundary value problem

$$\begin{aligned} \operatorname{div}(a(x)u) - \operatorname{div}(\kappa(x)\nabla u) &= f \qquad \text{in } \mu \\ u &= 0 \qquad \text{on } \partial\mu_D \\ \frac{\partial u}{\partial n} &= 0 \qquad \text{on } \partial\mu_N, \end{aligned}$$

where $\mu = [0,1]^3$, $\partial\mu_N = \partial\mu \backslash \partial\mu_D$. The tensor κ is isotropic and discontinuous. The domain contains many zones of high permeability which are isolated from each other. Let $[x]$

[4] www.fluorem.com/en/softwares/optimization/turb-opty-cfd

ETNA
Kent State University
http://etna.math.kent.edu

TABLE 4.1
Characteristics of test matrices, N: number of rows/columns, NNZ: nonzero entries.

Matrix	N	NNZ	geometry
IM07R	261,465	26,872,530	3D
VV11R	277,095	30,000,952	3D
RM07R	272,635	37,355,908	3D
3DCONSKY_121	1,771,561	50,178,241	3D
3DCONSKY_161	4,173,281	118,645,121	3D

denote the integer value of x. Then κ is given in 3D by

$$\kappa(x) = \begin{cases} 10^3 * ([10 * x_2] + 1) & \text{if } [10 * x_i] = 0 \text{ mod}(2), i = 1, 2, 3, \\ 1 & \text{otherwise.} \end{cases}$$

For the numerical tests, the right-hand side is set to $f = x_1^2 + x_2^2 + x_3^2$ and the velocity field to $a = (1000, 1000, 1000)^T$. The discretization is done using P2-type finite element methods in the Freefm++[5] package. We consider a uniform grid with $n \times n \times n$ nodes and choose $n = 121$ and 161. During our experiments, we rely on the $ILU(1)$ factorization to approximate the solutions on the subdomains induced by the additive Schwarz method.

4.3. Platform of tests. Experiments are done on a distributed memory supercomputer VARGAS[6] which has 3584 Power6 CPUs. Each Power6 CPU is a dual-core 2-way SMT with a peak frequency at 4.7 GHz. The computer is made of 112 nodes connected through an Infiniband network. Each node has 32 Power6 CPUs that access 128 GB of local memory in a non-uniform way (hardware NUMA nodes). The memory accessed by a single MPI process is limited to 3.2 GB for the data and 0.5 GB for the stack.

4.4. Analysis of convergence for the matrix RM07R. We first consider the large test case RM07R from the FLUOREM collection and compare AGMRES(m, r) with GMRES(m). We first use AGMRES without deflation ($r = 0$). The goal is to confirm that the two methods have the same convergence behavior for a reasonable restart length. After that, we show the benefits of using deflation when the restart length in AGMRES is small and when the number of subdomains increases. We finish this section by stating the benefits of using an adaptive strategy.

In Figure 4.1, we plot the convergence of GMRES(m) and AGMRES(m) with three restart lengths, $m = 32, 48,$ and 64. The number of subdomains is 32 and the LU factorization is used within the subdomains. The first remark on Figure 4.1 is that there is no real difference between the residual norm obtained from the two strategies. Secondly, the convergence curve for GMRES(m) indicates periodic stagnation in the iterative process. These ticks occur at the time of restart and are more visible when m is small, hence the larger number of iterations. These ticks suggest that some information is lost at the time of restart and that the augmented basis could be beneficial to improve the convergence rate in these cases. The other test cases show a similar behavior.

Now we show the impact of deflation by augmenting the basis. In Figure 4.2, we display the convergence history of GMRES(m) and AGMRES(m, r) with $m = 32, 48,$ and $r = 2$, that is we compute two approximate eigenvectors at each restart and use a basis of size $s = m + 2$. The number of subdomains is still 32. The adaptive strategy is not used at

[5] http://www.freefem.org/ff++/index.htm
[6] http://www.idris.fr/su/Scalaire/vargas/hw-vargas.html

ETNA
Kent State University
http://etna.math.kent.edu

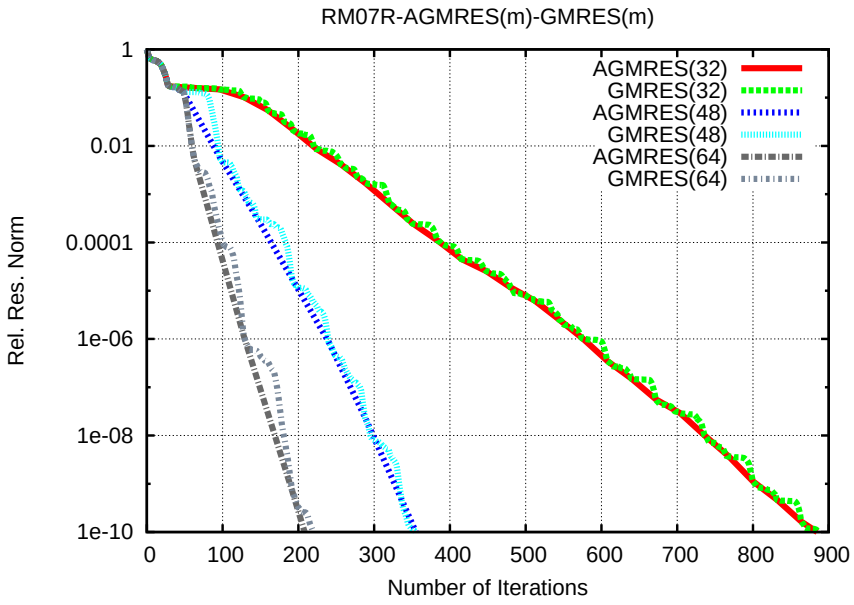

FIG. 4.1. *RM07R: Influence of the restart length in AGMRES and GMRES, 32 subdomains.*

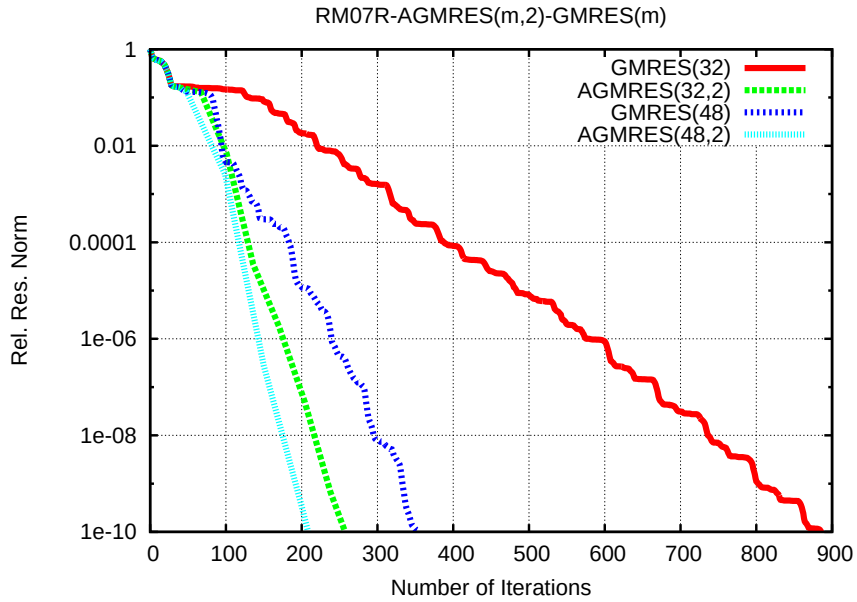

FIG. 4.2. *RM07R: Influence of the augmented basis in AGMRES over GMRES, 32 subdomains.*

this point. It can be clearly noticed that adding only two eigenvectors to the basis is sufficient to speed up the convergence in AGMRES. For instance, GMRES(32) requires 886 iterations while AGMRES(32,2) needs only 272 iterations. When we increase the restart length to 48, GMRES benefits greatly and requires 355 iterations to reach the desired accuracy while AGMRES(48,2) needs 250 iterations. The general remark here is that AGMRES(32,2) and

ETNA
Kent State University
http://etna.math.kent.edu

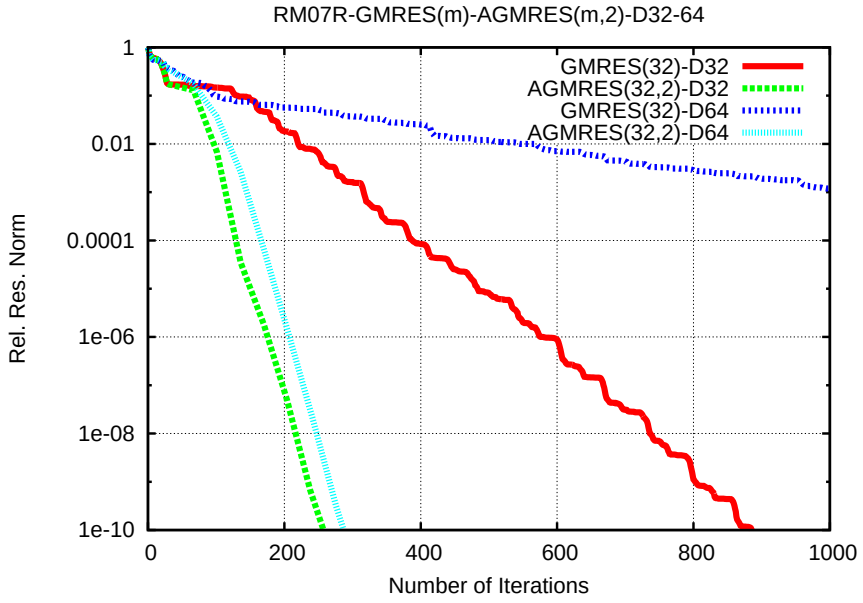

FIG. 4.3. *RM07R: Influence of the number of subdomains on the convergence of GMRES. The restart length is fixed and the benefits of the augmented basis in AGMRES is clearly visible.*

TABLE 4.2
RM07R: Number of iterations with GMRES(m), AGMRES($m, 2$), and AGMRES($m, 4$) as a function of the number of subdomains in the Restricted Additive Schwarz.

KSP	GMRES(m)			AGMRES($m, 2$)			AGMRES($m, 4$)		
D \ m	32	48	64	32	48	64	32	48	64
8	93	70	57	100	98	57	105	100	57
16	254	169	123	169	148	130	177	153	132
32	886	355	220	272	250	196	212	205	200
64	-	702	445	311	303	265	287	258	270

AGMRES(48,2) have almost the same convergence rate while GMRES is more sensitive to the restart length. This is more visible when the number of subdomains vary.

The robustness of Schwarz preconditioners decreases as the number of subdomains increases. GMRES will thus require more and more iterations, particularly if the restart length is fixed. We show this behavior in Figure 4.3, where the restart length is fixed and the number of subdomains is increased. Clearly, as expected, the number of iterations in GMRES increases as we add more subdomains. For instance, GMRES(32) requires 886 iterations with 32 subdomains. With 64 subdomains, we carry out 1000 iterations without reaching the prescribed tolerance of 10^{-10}. In AGMRES(m, r), there is no such difference. As we increase the number of subdomains, we observe that the convergence rates remain quite close. Indeed, AGMRES(32,2) requires respectively 272 and 311 iterations for 32 and 64 subdomains. The fact that the number of iterations increases only slightly when increasing D has great impact on the scalability of AGMRES. We give timings in the next section.

In GMRES, a better convergence rate can be obtained if the restart length is increased as a function of the number of subdomains. We show in Table 4.2 that in such a case, it is

ETNA
Kent State University
http://etna.math.kent.edu

still beneficial to have an augmented basis in AGMRES. These results can be divided into three parts:

1. With 8 subdomains, GMRES needs fewer iterations than AGMRES for all values of the restart length. Note that this difference is mainly due to the fact that the stopping test is computed only at each outer cycle in AGMRES. The accuracy achieved in AGMRES for these cases is always better. Typically, AGMRES gives an accuracy of 10^{-14} in the computed residual while it is 10^{-11} for GMRES.
2. For the same reasons, AGMRES needs more iterations than GMRES for 16 subdomains and a large value of m. However, for small values of m, AGMRES is clearly better than GMRES.
3. For 32 subdomains, AGMRES needs fewer iterations than GMRES for all restart lengths. The dash in GMRES(32) for the 64 subdomains indicates that the desired accuracy has not been reached within the 1000 iterations allowed. On the contrary, it requires only 300 iterations for AGMRES(m, r) to converge.
   Thus, the main empirical conclusion from these experiments and others not reported here is that AGMRES is less sensitive to the restart length and the number of subdomains than GMRES. On the other hand, AGMRES is rather sensitive to the number of extracted eigenvectors. As for the basis size, it is indeed difficult to know how many vectors should be added to the basis to improve convergence. If r (the number of eigenvectors to be added) is very large, the process of updating the eigenvectors could add more overhead. If r is small, deflation might not be beneficial. The proposed adaptive strategy provides a trade-off between these two bounds.

If some information about the convergence behavior has been collected before, then it can be used to define the smv and bgv parameters in an adaptive strategy. Our goal is to show that this technique can be used to speed up the convergence by adaptively adjusting the frequency and the number of extracted eigenvalues. We take the smallest restart length $m = 32$, a large number of subdomains, and the smallest number of harmonic Ritz vectors $r = 1$. Yet, we know from $D = 16$ that GMRES(32) and thus AGMRES(32) needs roughly 254 iterations. From the adaptive strategy, we still set the maximum number of iterations $itmax = 1000$ but now we set $smv = 0.1$ and $bgv = 0.2$. As explained in Section 3.2.6, $smv \times itmax$ defines the lower bound below which it is not beneficial to use an augmented basis, and the product $bgv \times itmax$ defines the upper bound beyond which slow convergence rate is expected and some action should be done. In this last case, we increase r by a fixed value l. We take $l = 2$ in this case. Figure 4.4 displays the convergence history of AGMRES($m, 1$) with $m = 24$ and $m = 32$. It can be observed that when $r = 1$ and without an adaptive strategy, the augmented basis does not contain enough spectral information to speed up the convergence. When r is adaptively increased, the basis recovers more and more spectral information and the convergence rate gets better. At convergence, for instance, we obtain $r = 3$ and $r = 5$, respectively for AGMRES(24,1) and AGMRES(32,1) with the adaptive strategy, while $r = 1$ for the non-adaptive strategy. It should be noted, however, that the actual limitation of the proposed adaptive strategy is the choice of the appropriate values of smv and bgv. It is heuristic and problem-dependent. Nevertheless, if there is some experimental knowledge available about the convergence of GMRES on similar problems, a good interval around $itmax$ can be set by smv and bgv to detect near-stagnation and to switch to the augmented basis.

4.5. Convergence analysis for the other matrices. Table 4.3 reports the number of iterations for the two test cases IM07R and VV11R. For AGMRES(m, r) we use $r = 2$ and adaptively increase it until $rmax = 5$. As noted before, we observe that for a fixed value of m, the number of iterations increases as D increases. It increases faster for GMRES than

ETNA
Kent State University
http://etna.math.kent.edu

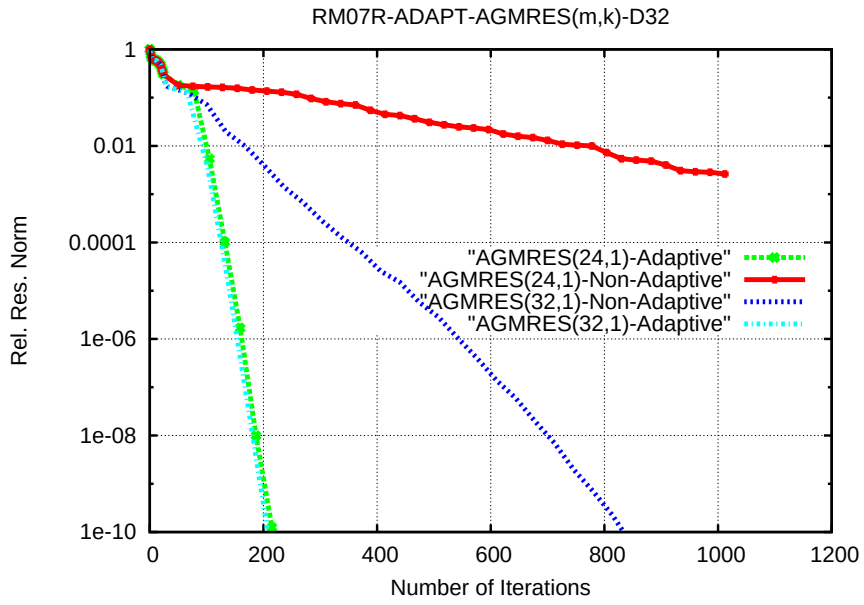

FIG. 4.4. *RM07R: Benefits of the adaptive deflation strategy. Restart=24 and 32, 32 subdomains.*

TABLE 4.3
VV11R & IM07R: Number of iterations with GMRES(m), AGMRES(m, r) as a function of the number of subdomains in the Restricted Additive Schwarz; r is adaptively increased until $rmax = 5$.

KSP	GMRES(m)			AGMRES($m, 2$)		
D \ m	24	32	48	24	32	48
VV11R						
8	251	191	147	248	172	146
16	499	458	288	492	304	207
32	-	957	670	641	541	516
IM07R						
8	240	235	189	249	203	195
16	695	623	521	378	370	316
24	927	913	759	492	444	408
32	-	-	833	724	629	579

for AGMRES. We note here again that deflation is needed to reach a good accuracy for large values of D. For the IM07R test case, for instance, and for 32 subdomains in the additive Schwarz method, neither GMRES(24) nor GMRES(32) can produce the desired accuracy while AGMRES(32) requires 724 iterations to converge.

We end this section by considering the two matrices arising from the convection-diffusion problem. In Table 4.4, we report on the number of iterations. Unlike the previous test cases, here GMRES is less sensitive to the restart parameter and the variation of subdomains. Hence the augmented basis is not as beneficial for the convergence as it was before. Nevertheless, AGMRES is still faster than GMRES if we consider the parallel efficiency. This is the topic of the next sections.

ETNA
Kent State University
http://etna.math.kent.edu

TABLE 4.4
Number of iterations with GMRES and AGMRES for the test problems 3DCONSKY_121 and 3DCONSKY_161.

Matrix	3DCONSKY_121		3DCONSKY_161	
D \ m	GMRES(16)	AGMRES(16,1)	GMRES(16)	AGMRES(16,1)
16	158	169	229	177
32	164	141	251	177
64	170	141	261	177
128	180	141	262	177
256	202	159	266	195

4.6. Analysis of the CPU time. To better show the benefits of using an augmented subspace approach with the Newton basis, we analyze the timing results in this section. The paramount goal when showing these results is that, as we increase the number of subdomains, we should be able to get a decrease in the iterative time. In GMRES(m) and AGMRES(m), the best way is undoubtedly to increase the restart length as well. Even then, the time will not decrease efficiently because of the negative effects of the restarting procedure and the weakness of the one-level Schwarz preconditioner. In AGMRES(m, r), only a few extracted Ritz vectors are sufficient to decrease the time and obtain a significant efficiency.

Table 4.5 compares GMRES(m), AGMRES(m), and AGMRES(m, r) for the test case RM07R by varying the number of subdomains D, the restart length m, and the number of harmonic Ritz values r, which increases adaptively until $rmax = 5$. The number of MPI processes is equal to the number of subdomains. The total time is the CPU time required to perform all the steps in Algorithm 2. The iterative time is the time spent in step 7. The set up time is the difference between the two times. It is independent of the method and of m. It decreases when D increases because the subdomains become smaller and the LU factorizations are faster. Thus from now on, we focus on the iterative time. The time per iteration is the time of one cycle divided by the number of matrix-vectors products in the cycle, which is m or $m+r$. It includes the time to compute the orthonormal basis (with Arnoldi GMRES or QR factorization for AGMRES) and the time to update the eigenvectors U for AGMRES(m, r). The iterative time is thus the product of the time per iteration and the number of iterations. The behavior of both GMRES(m) and AGMRES(m) is similar. Increasing m has two opposite effects: it decreases the number of iterations (in some cases, the number of cycles remain the same for AGMRES) and increases the time per iteration because of the orthogonalization steps. Thus in most cases, there is an optimal value of m, which depends on D, with a minimal solution time. Increasing D has also two opposite effects but in the reverse way: it increases the number of iterations and decreases the time per iteration, thus there is in general an optimal value of D, which depends on m. Even though their behavior is similar, AGMRES(m) clearly performs faster than GMRES(m) for all but one configuration. This is mainly due to a faster time per iteration thanks to a more efficient parallel algorithm. This is explored in the next section by analyzing the communication volume.

The objective of deflation in AGMRES(m, r) is two-fold: to get an algorithm that is less sensitive to m and to increase the number of subdomains (thus the number of MPI processes). For D fixed, there is still an optimal value of m but it is smaller. The iterative time decreases from $D = 8$ until $D = 64$. Thus, our method allows us to choose a small value of m and to reduce the CPU time with a large number of subdomains. We get indeed a more efficient parallelism because the number of iterations does not inflate. Clearly, AGMRES(m, r) gives

ETNA
Kent State University
http://etna.math.kent.edu

TABLE 4.5
Timing statistics for RM07R. D: number of subdomains and number of MPI processes, Total Time: CPU elapsed time in seconds, Iter. Time: CPU time in the iterative phase, Time/Iter: average time spent in each iteration (matrix-vector product and preconditioning step), MSG: MPI messages and reductions. With AGMRES(m, r), r is adaptively increased until $rmax = 5$.

D	Algo.	Total Time	Iter. Time	Time/Iter	MSG ($\times 10^4$)
8	GMRES(32)	427.3	327.33	3.52	1.74
	GMRES(48)	386.1	291.64	4.166	1.41
	GMRES(64)	358.1	264.58	4.64	1.03
	AGMRES(32)	358.5	263.08	2.74	1.3
	AGMRES(48)	369.1	271.9	2.832	1.45
	AGMRES(64)	329.4	236.76	4.228	1.23
	AGMRES(32,2)	347.4	257.11	2.624	1.32
	AGMRES(48,2)	373.1	277.98	2.837	1.48
	AGMRES(64,2)	329.4	236.76	4.228	1.23
16	GMRES(32)	379.3	349.97	1.378	13.1
	GMRES(48)	333.1	302.66	1.791	9.05
	GMRES(64)	286.8	257.03	2.09	6.88
	AGMRES(32)	305.8	276.1	1.079	8.1
	AGMRES(48)	263.0	230.5	1.201	6.78
	AGMRES(64)	256.8	227.82	1.78	5.56
	AGMRES(32,2)	224.1	193.39	1.316	9.84
	AGMRES(48,2)	240.9	210.56	1.376	10.01
	AGMRES(64,2)	231.4	201.05	1.547	5.66
32	GMRES(32)	573.4	557.13	0.629	96.25
	GMRES(48)	239.5	223.54	0.63	39.74
	GMRES(64)	158.4	139.2	0.633	25.38
	AGMRES(32)	273.0	256.91	0.287	54.97
	AGMRES(48)	167.1	150.84	0.393	25.93
	AGMRES(64)	131.4	114.83	0.449	19.42
	AGMRES(32,4)	91.41	75.23	0.357	31.83
	AGMRES(48,4)	94.79	79.028	0.38	33.9
	AGMRES(64,4)	99.45	83.148	0.406	32.24
64	GMRES(32)	-	-	-	-
	GMRES(48)	214.8	204.16	0.291	227.02
	GMRES(64)	165.6	156.44	0.352	145.69
	AGMRES(32)	-	-	-	-
	AGMRES(48)	167.0	157.72	0.219	132.42
	AGMRES(64)	97.87	86.066	0.192	88.67
	AGMRES(32,4)	62.39	52.839	0.202	101.53
	AGMRES(48,4)	67.0	57.733	0.22	110.99
	AGMRES(64,4)	63.15	53.788	0.203	116.08

the smallest CPU time. These results are confirmed with other test cases shown in Table 4.6 and Figures 4.5 and 4.6.

It is better for GMRES(m) to choose a small number of subdomains D and a large restart length m. On the contrary, it is more efficient to choose a large number of subdomains D and a small restart length m with our method AGMRES(m, r). Clearly, AGMRES(m, r) is

ETNA
Kent State University
http://etna.math.kent.edu

TABLE 4.6
Timing statistics for GMRES and AGMRES for the test cases VV11R and IM07R. D: number of subdomains and number of MPI processes, Iter. Time: time spent in the iterative phase, MSG: MPI messages and reductions. With AGMRES(m, r), r is adaptively increased until $rmax = 5$.

D \ m	24		32		48		
	Iter. Time	MSG	Iter. Time	MSG	Iter. Time	MSG	
	GMRES(m)						
8	92.84	2.05	68.95	1.69	77.7	1.47	VV11R
16	101.1	12.27	89.37	11.47	63.2	7.66	
32	-	-	31.2	22.5	29.7	18.54	
	AGMRES($m, 2$)						
8	52.8	1.28	38.5	1.02	40.5	1.05	
16	51.8	7.4	34.5	4.91	28.08	3.87	
32	38.3	25.6	31.2	22.5	29.7	18.5	
	GMRES(m)						
8	76.219	2.6	73.3	2.63	63.669	2.31	IM07R
16	111.74	20.06	96.246	18.25	83.583	15.76	
32	-	-	-	-	77.066	59.87	
	AGMRES($m, 2$)						
8	45.781	1.65	40.905	5.48	40.85	1.52	
16	36.492	21.65	34.803	24.12	33.65	23.64	
32	33.262	94.54	27.837	93.27	27.109	105.35	

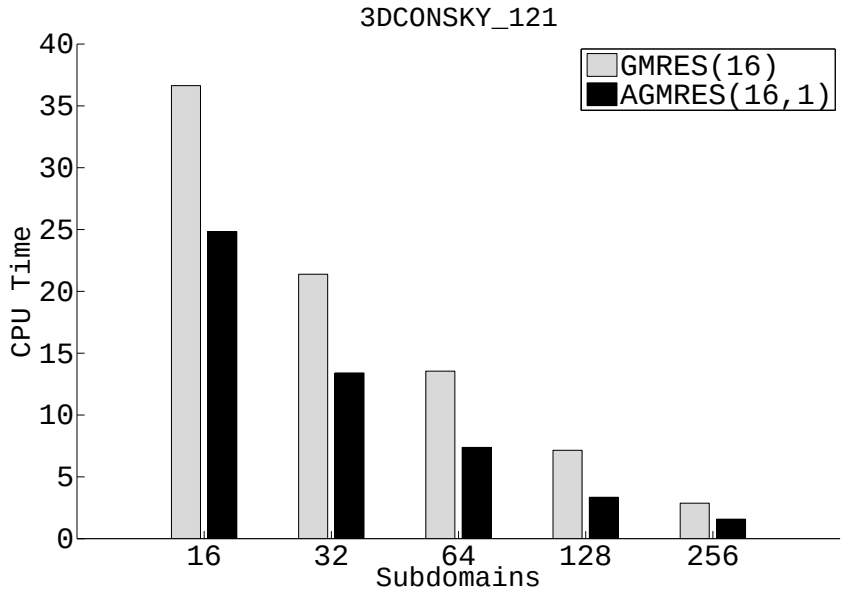

FIG. 4.5. *CPU time for the iterative phase for the 3D* $121 \times 121 \times 121$ *convective SkyScraper problem (matrix size 1,771,561, nonzeros 50,178,241), 16 to 256 subdomains, ILU(1) in subdomains,* $m = 16$, $r = 1$.

ETNA
Kent State University
http://etna.math.kent.edu

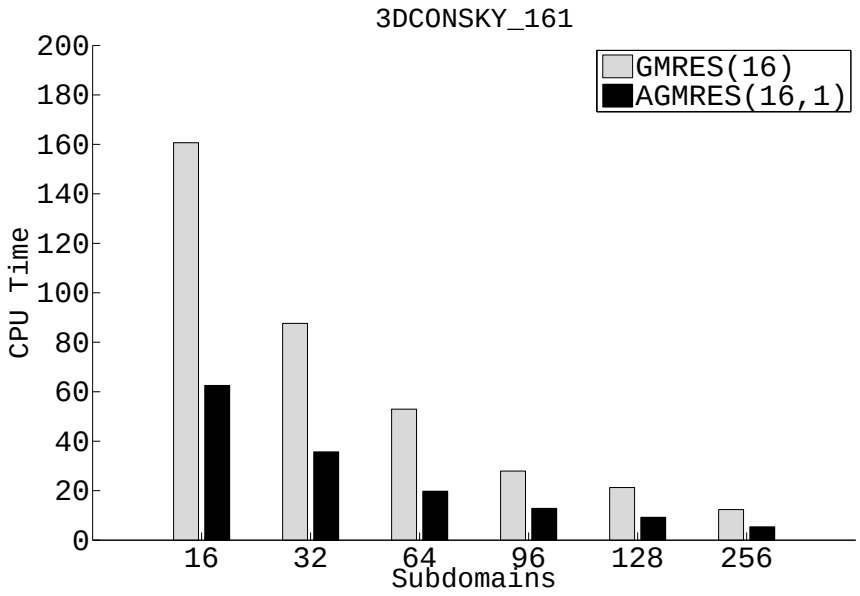

FIG. 4.6. *CPU time in the iterative phase of GMRES and AGMRES for the 3D* $161 \times 161 \times 161$ *convective SkyScraper problem (matrix size 4,173,281, nonzeros 118,645,121), 16 to 256 subdomains, ILU(1) in subdomains,* $m = 16$, $r = 1$.

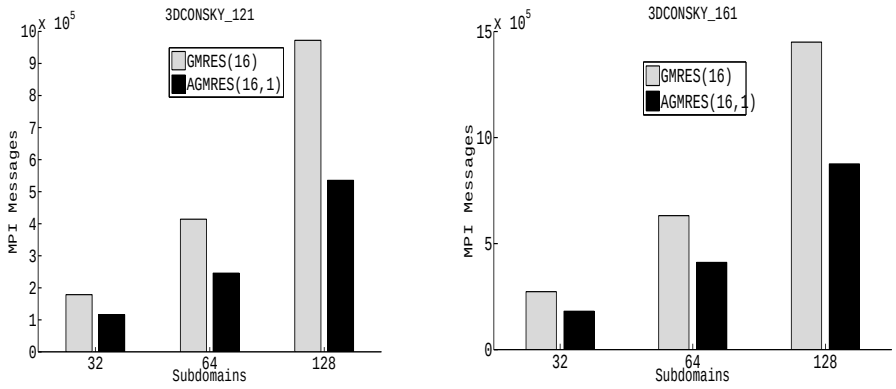

FIG. 4.7. *Amount of MPI Messages in the iterative phase of GMRES and AGMRES for the convective SkyScraper problems on the 3D* $121 \times 121 \times 121$ *and* $161 \times 161 \times 161$ *grids.*

faster than GMRES(m). In order to compare the methods with similar memory requirements, we choose $m = 24$ for AGMRES and $m = 48$ for GMRES, since AGMRES needs to store the two systems W_s and V_s. For all but one value of D, AGMRES$(24, r)$ is faster than GMRES(48) for both matrices VV11R and IM07R. This is also true for AGMRES$(32, r)$ compared to GMRES(64) for the matrix RM07R.

4.7. Analysis of parallelism. Another advantage of AGMRES over GMRES is the communication volume. In Tables 4.5 and 4.6 and in Figure 4.7, we have reported the number of MPI messages exchanged. The counts are done on the Send/Receive routines as well as the collective communications (Reduce and Broadcast). We do not take into account the MPI message lengths. It appears that the number of messages is a function of the num-

ETNA
Kent State University
http://etna.math.kent.edu

ber of subdomains. This is generally reduced by allowing many subdomains to be assigned to a unique CPU. In the problems under study, this is not feasible in practice as it will induce more iterations as the number of subdomains increases. The communication volume is obviously proportional to the number of iterations as well. The second observation is that AGMRES communicates less than GMRES for the same number of subdomains and the same basis size. As more subdomains are used, the gap between the two methods increases. For instance, in Table 4.5, GMRES on 64 subdomains produces nearly a ratio of 1.5 more messages than AGMRES. In the augmented basis, the situation is different. At each cycle, AGMRES(m, r) communicates more than AGMRES(m) because of the computation of the eigenvectors. However, since a substantial number of iterations is saved by using the augmented basis, we actually observe a better communication in AGMRES(m, r). Now between GMRES and AGMRES(m, r), the previous analysis holds as well but there are two cases: when the restart length is very close to the number of subdomains, the communication for the computation of eigenvectors may dominate if there is no substantial acceleration in the convergence rate of AGMRES(m, r). This is observed in Table 4.6 for VV11R and IM07R. With a substantial gain in the convergence rate as in Table 4.5, AGMRES(m, r) benefits from that and the communication volume decreases proportionally to the number of iterations. The second situation occurs when the number of subdomains is very large with respect to the basis size. Even if there is no substantial acceleration in AGMRES(m, r), the kernel computations of AGMRES will produce less communication volume than that of GMRES. This is observed in Figure 4.7. As the number of subdomains increases, the difference between the two methods is more and more pronounced. A fine-tuned adaptive strategy is still required to determine whether or not to augment the basis.

5. Concluding remarks. We have proposed the AGMRES(m, r) implementation, which combines the Newton basis GMRES implementation with an augmented subspace technique. This approach benefits from the high level of parallelism during the kernel computation of the Krylov basis. The proposed augmented basis reduces negative effects due to restarting and due to a large number of subdomains.

The numerical results on the VARGAS supercomputer (IBM Power 6 processors) confirm that AGMRES communicates less than GMRES and produces a faster solution of large linear systems. Moreover, for the proposed test cases, AGMRES gives a fairly good convergence rate when a few Ritz vectors are added to the Krylov basis. The proposed implementation is done in the PETSc package. It thus benefits from the optimized routines for the usual linear algebra operations on matrices and vectors. Its object-oriented interface allows us to use transparently any parallel preconditioner implemented in the package based on algebraic domain decomposition methods or multilevel methods. Indeed it can be used as a smoother for algebraic multigrid methods [18].

Although the proposed augmented basis behaves well for the presented test cases, there are some cases where it may not be useful to use. Careful analysis is still needed of the adaptive strategy to avoid the computation of Ritz vectors.

The proposed augmented basis is combined with a fixed preconditioning matrix. We plan to combine it with flexible GMRES in a future work.

Acknowledgments. This work was funded by the French National Agency of Research under the contract ANR-TLOG07-011-03 LIBRAERO. Numerical experiments have been done on the VARGAS supercomputer from GENCI-IDRIS (Grand Equipement National de Calcul Intensif-Institut du Développement et des Ressources en Informatique Scientifique). Preliminary tests were carried out using the GRID'5000 experimental testbed[7]. We are very

[7] https://www.grid5000.fr

ETNA
Kent State University
http://etna.math.kent.edu

grateful to Bernard Philippe and François Pacull for many suggestions and helpful discussions during this work. We would like to thank Guy A. Atenekeng for providing us with the test matrices of the convection-diffusion equation. We are grateful to Roger B. Sidje for giving us the implementation of the RODDEC method.

REFERENCES

[1] Y. ACHDOU AND F. NATAF, *Low frequency tangential filtering decomposition*, Numer. Linear Algebra Appl., 14 (2007), pp. 129–147.
[2] P. R. AMESTOY, I. S. DUFF, J.-Y. L'EXCELLENT, AND J. KOSTER, *A fully asynchronous multifrontal solver using distributed dynamic scheduling*, SIAM J. Matrix Anal. Appl., 23 (2001), pp. 15–41.
[3] P. R. AMESTOY, I. S. DUFF, D. RUIZ, AND B. UÇAR, *A parallel matrix scaling algorithm*, in High Performance Computing for Computational Science VECPAR 2008, J. M. Palma, P. R. Amestoy, M. Daydé, M. Mattoso, and J. A. C. Lopes, eds., Springer, Berlin, 2008, pp. 301–313.
[4] G.-A. ATENEKENG-KAHOU, *Parallélisation de GMRES préconditionné par une itération de Schwarz multiplicatif*, PhD Thesis, Institut de Recherche en Informatique et Automatisme, University of Rennes I and University of Yaounde I, 2008.
[5] J. BAGLAMA, D. CALVETTI, G. H. GOLUB, AND L. REICHEL, *Adaptively preconditioned GMRES algorithms*, SIAM J. Sci. Comput., 20 (1998), pp. 243–269.
[6] Z. BAI, D. HU, AND L. REICHEL, *A Newton basis GMRES implementation*, IMA J. Numer. Anal., 14 (1994), pp. 563–581.
[7] S. BALAY, J. BROWN, K. BUSCHELMAN, V. EIJKHOUT, W. D. GROPP, D. KAUSHIK, M. G. KNEPLEY, L. C. MCINNES, B. F. SMITH, AND H. ZHANG, *PETSc users manual*, Tech. Report ANL-95/11-Revision 3.2.0, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL, 2011.
[8] S. BALAY, K. BUSCHELMAN, W. D. GROPP, D. KAUSHIK, M. G. KNEPLEY, L. C. MCINNES, B. F. SMITH, AND H. ZHANG, *PETSc Web page*, 2011. http://www.mcs.anl.gov/petsc
[9] M. BENZI, *Preconditioning techniques for large linear systems: a survey*, J. Comput. Phys., 182 (2002), pp. 418–477.
[10] K. BURRAGE AND J. ERHEL, *On the performance of various adaptive preconditioned GMRES strategies*, Numer. Linear Algebra Appl., 5 (1998), pp. 101–121.
[11] X.-C. CAI AND M. SARKIS, *A restricted additive Schwarz preconditioner for general sparse linear systems*, SIAM J. Sci. Comput., 21 (1999), pp. 792–797.
[12] A. CHAPMAN AND Y. SAAD, *Deflated and augmented Krylov subspace techniques*, Numer. Linear Algebra Appl., 4 (1997), pp. 43–66.
[13] R. DA CUNHA, D. BECKER, AND J. PATTERSON, *New parallel (rank-revealing) QR factorization algorithms*, in Euro-Par 2002 Parallel Processing, B. Monien and R. Feldmann, eds., vol. 2400 of Lecture Notes in Computer Science, Springer, Berlin, 2002, pp. 209–237.
[14] T. A. DAVIS AND Y. HU, *The University of Florida sparse matrix collection*, ACM Trans. Math. Software, 38 (2011), Art. 1 (25 pages).
[15] E. DE STURLER, *Iterative methods on distributed memory computers*, PhD Thesis, Faculty of Technical Mathematics and Informatics, Delft University of Technology, Delft, The Netherlands, October 1994.
[16] J. DEMMEL, L. GRIGORI, M. HOEMMEN, AND J. LANGOU, *Communication-optimal parallel and sequential QR and LU factorizations*, SIAM J. Sci. Comput., 34 (2012), pp. A206–A239.
[17] M. EIERMANN, O. G. ERNST, AND O. SCHNEIDER, *Analysis of acceleration strategies for restarted minimal residual methods*, J. Comput. Appl. Math., 123 (2000), pp. 261–292.
[18] H. C. ELMAN, O. G. ERNST, AND D. P. O'LEARY, *A multigrid method enhanced by Krylov subspace iteration for discrete Helmhotz equations*, SIAM J. Sci. Comput., 23 (2001), pp. 1291–1315.
[19] J. ERHEL, *A parallel GMRES version for general sparse matrices*, Electron. Trans. Numer. Anal., 3 (1995), pp. 160–176. http://etna.math.kent.edu/vol.3.1995/pp160-176.dir
[20] ———, *A parallel preconditioned GMRES algorithm for sparse matrices*, in The Mathematics of Numerical Analysis, J. Renegar, M. Shub, and S. Smale, eds., vol. 32 of Lectures in Appl. Math., AMS, Providence, 1996, pp. 345–355.
[21] ———, *Some properties of Krylov projection methods for large linear systems*, Comp. Tech. Rev., 3 (2011), pp. 41–70.
[22] J. ERHEL, K. BURRAGE, AND B. POHL, *Restarted GMRES preconditioned by deflation*, J. Comput. Appl. Math., 69 (1996), pp. 303–318.
[23] L. GIRAUD, S. GRATTON, X. PINEL, AND X. VASSEUR, *Flexible GMRES with deflated restarting*, SIAM J. Sci. Comput., 32 (2010), pp. 1858–1878.

ETNA
Kent State University
http://etna.math.kent.edu

[24] M. HOEMMEN, *Communication-avoiding Krylov subspace methods*, PhD Thesis, Department of Mathematics, University of California, Berkeley, 2010.
[25] W. JALBY AND B. PHILIPPE, *Stability analysis and improvement of the block Gram-Schmidt algorithm*, SIAM J. Sci. Statist. Comput., 12 (1991), pp. 1058–1073.
[26] W. JOUBERT AND G. CAREY, *Parallelizable restarted iterative methods for nonsymmetric linear systems. part II: parallel implementation*, Int. J. Comp. Math., 44 (1992), pp. 269–290.
[27] S. A. KHARCHENKO AND A. Y. YEREMIN, *Eigenvalue translation based preconditioners for the* GMRES(k) *method*, Numer. Linear Algebra Appl., 2 (1995), pp. 51–77.
[28] S. KIM AND A. CHRONOPOULOS, *An efficient parallel algorithm for extreme eigenvalues of sparse nonsymmetric matrices*, Int. J. High Perform. Comp. Appl., 6 (1992), pp. 407–420.
[29] P. KUMAR, L. GRIGORI, F. NATAF, AND Q. NIU, *Combinative preconditioning based on Relaxed Nested Factorization and Tangential Filtering preconditioner*, Research Report RR-6955, INRIA, Paris, 2009.
[30] M. MOHIYUDDIN, M. HOEMMEN, J. DEMMEL, AND K. YELICK, *Minimizing communication in sparse matrix solvers*, in SC '09: Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis, New York, ACM, 2009, Art. 36 (12 pages).
[31] R. B. MORGAN, *A restarted GMRES method augmented with eigenvectors*, SIAM J. Matrix Anal. Appl., 16 (1995), pp. 1154–1171.
[32] ———, *GMRES with deflated restarting*, SIAM J. Sci. Comput., 24 (2002), pp. 20–37.
[33] D. NUENTSA WAKAM AND G.-A. ATENEKENG-KAHOU, *Parallel GMRES with a multiplicative Schwarz preconditioner*, ARIMA Rev. Afr. Rech. Inform. Math. Appl., 14 (2011), pp. 81–99.
[34] D. NUENTSA WAKAM, J. ERHEL, E. CANOT, AND G.-A. ATENEKENG KAHOU, *A comparative study of some distributed linear solvers on systems arising from fluid dynamics simulations*, in Parallel Computing: From Multicores and GPU's to Petascale, B. Chapman, F. Desprez, G. R. Joubert, A. Lichnewsky, F. Peters, and T. Priol, eds., vol. 19 of Advances in Parallel Computing, IOS Press, Amsterdam, 2010, pp. 51–58.
[35] D. NUENTSA WAKAM, J. ERHEL, AND W. D. GROPP, *Parallel adaptive deflated GMRES*, in Domain Decomposition Methods in Science and Engineering XX, R. Bank, M. Holst, O. Widlund, and J. Xu, eds., Lecture Notes in Computational Science and Engineering, 91, Springer, Berlin, 2013, pp. 631–638.
[36] F. PACULL, S. AUBERT, AND M. BUISSON, *Study of ILU factorization for Schwarz preconditioners with application to computational fluid dynamics*, in Proceedings of the Second International Conference on Parallel, Distributed, Grid and Cloud Computing for Engineering, P. Iványi, B. H. V. Topping, eds., Civil-Comp Press, Stirlingshire, UK, Paper 39, 2011.
[37] B. PHILIPPE AND L. REICHEL, *On the generation of Krylov subspace bases*, Appl. Numer. Math., 62 (2012), pp. 1171–1186.
[38] L. REICHEL, *Newton interpolation at Leja points*, BIT, 30 (1990), pp. 332–346.
[39] Y. SAAD, *A flexible inner-outer preconditioned GMRES algorithm*, SIAM J. Sci. Comput., 14 (1993), pp. 461–469.
[40] ———, *Iterative Methods for Sparse Linear Systems*, 2nd ed., SIAM Philadelphia, 2003.
[41] Y. SAAD AND M. H. SCHULTZ, *GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.
[42] R. B. SIDJE, *Alternatives for parallel Krylov subspace basis computation*, Numer. Linear Algebra Appl., 4 (1997), pp. 305–331.
[43] R. B. SIDJE AND B. PHILIPPE, *Parallel Krylov subspaces basis computation*, in Proc. CARI'94, J. Tankoano, ed., ORSTOM Editions, Paris, 1994, pp. 421–440.
[44] V. SIMONCINI, *On a non-stagnation condition for GMRES and application to saddle point matrices*, Electron. Trans. Numer. Anal., 37 (2010), pp. 202–213.
http://etna.math.kent.edu/vol.37.2010/pp202-213.dir
[45] V. SIMONCINI AND D. B. SZYLD, *Recent computational developments in Krylov subspace methods for linear systems*, Numer. Linear Algebra Appl., 14 (2007), pp. 1–59.
[46] ———, *New conditions for non-stagnation of minimal residual methods*, Numer. Math., 109 (2008), pp. 477–487.
[47] M. SOSONKINA, L. T. WATSON, R. K. KAPANIA, AND H. F. WALKER, *A new adaptive GMRES algorithm for achieving high accuracy*, Numer. Linear Algebra Appl., 5 (1998), pp. 275–297.
[48] H. A. VAN DER VORST AND C. VUIK, *The superlinear convergence behaviour of GMRES*, J. Comput. Appl. Math., 48 (1993), pp. 327–341.