Algorithm 881: A Set of Flexible GMRES Routines for Real and Complex Arithmetics on High-Performance Computers

VALÉRIE FRAYSSÉ
Kvasar Technology LLC
LUC GIRAUD
ENSEEIHT-IRIT
and
SERGE GRATTON
CNES

In this article we describe our implementations of the FGMRES algorithm for both real and complex, single and double precision arithmetics suitable for serial, shared-memory, and distributed-memory computers. For the sake of portability, simplicity, flexibility, and efficiency, the FGMRES solvers have been implemented in Fortran 77 using the reverse communication mechanism for the matrix-vector product, the preconditioning, and the dot-product computations. For distributed-memory computation, several orthogonalization procedures have been implemented to reduce the cost of the dot-product calculation, which is a well-known bottleneck of efficiency for Krylov methods. Furthermore, either implicit or explicit calculation of the residual at restart is possible depending on the actual cost of the matrix-vector product. Finally, the implemented stopping criterion is based on a normwise backward error.

Categories and Subject Descriptors: G.1.3 [**Numerical Analysis**]: Numerical Linear Algebra—*Sparse, structured and very large systems (direct and iterative methods)*; G.4 [**Mathematics of Computing**]: Mathematical Software

General Terms: Algorithms

Additional Key Words and Phrases: Linear systems, flexible Krylov methods, FGMRES, reverse communication, high-performance computing, distributed memory

The major part of this work was carried out while the three authors were working at CERFACS.
Authors' addresses: V. Frayssé, Kvasar Technology LLC, Boston, MA; L. Giraud, ENSEEIHT-IRIT, Toulouse, France; S. Gratton, CNES, Toulouse, France; email: gratton@cerfacs.fr.
Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or direct commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credits is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from the Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.
© 2008 ACM 0098-3500/2008/07-ART13 $5.00 DOI: 10.1145/1377612.1377617. http://doi.acm.org/10.1145/1377612.1377617.

ACM Reference Format:
Frayssé, V., Giraud, L., and Gratton, S. 2008. Algorithm 881: A set of flexible GMRES routines for real and complex arithmetics on high-performance computers. ACM Trans. Math. Softw. 35, 2, Article 13 (July 2008), 12 pages. DOI = 10.1145/1377612.1377617. http://10.1145/1377612.1377617.

1. INTRODUCTION

The FGMRES (flexible generalized minimum residual) method [Saad 1993] is among the most widely used Krylov solvers for the iterative solution of general, large, linear systems when variable preconditioning is considered. This numerical algorithm is embedded in many sophisticated package products that are either specialized packages for the solution of PDEs [Balay et al. 2004; Tuminaro et al. 1999] or general-purpose packages for the solution of sparse linear systems [Li et al. 2003]. Using these sophisticated packages requires one to comply with a predefined data structure (that is package dependent); furthermore, many of these packages do not support all the arithmetics and do not offer a wide range of orthogonalisation schemes. Even though a basic version of this numerical algorithm is fairly simple to describe and implement, we did not find any general implementation for complex matrices when we first looked for it a few years ago. This is the main reason why we decided to develop our own implementation, with the objectives of having a parallel portable package with a simple API (application program interface) easily understandable by nonspecialists in linear algebra, while incorporating all the features enabling one to play with the different possible variants for orthogonalization schemes, restarting strategies, and stopping criteria.

Four variants of the Gram-Schmidt orthogonalization procedure are implemented that permit the user to select the best-suited variant, depending on the target parallel computer and the conditioning of the linear system. We made the dot-product calculation available in reverse communication to comply with parallel distributed-memory usages, where this calculation generally requires a global reduction. In many cases, because of the limited amount of memory or prohibitive cost of orthogonalization, a restart should be performed and a new initial residual computed. To address those situations where the matrix-vector product is inexpensive, as in finite-element calculation for instance, or expensive, as in boundary-element calculation where the matrix is dense and often approximated via a fast multipole technique, the user can select either an explicit or implicit calculation of the residual. As suggested in Saad [1993], we implement a mechanism enabling us to release a contigious part of the workspace so that the user might employ it to implement the preconditioning step. For instance, the user can implement a few steps of the restarted GMRES [Saad and Schultz 1986], where the size of restart is set according to the amount of memory available at each step. Finally, the stopping criterion is a key component for iterative solvers. We have decided to enable the user to choose among all the possible criteria, based on normwise backward error in the Euclidean norm that are commonly admitted as relevant for iterative solvers [Barrett et al. 1994; Strakoš and Tichy 2006].

The purpose of this article is to present the API of the FGMRES routines and to describe several choices that have been made in order to get an efficient and reliable implementation suitable for real and complex arithmetic on any scientific computer. The package was first made available in the public domain in 1998 and has so far been downloaded more than 1100 times, both by academic and research organizations and by industrial companies. The most popular arithmetics are the double real and the double complex. The latter seems to be fairly popular with those solving wave propagation and convection diffusion problems, as illustrated by the list of papers in that field that reference our package [Batiste et al. 2004; Carpentieri et al. 2005; Meca et al. 2004a; 2004b; Mercader et al. 2005; 2004; Warsa et al. 2004]. The selection of Fortran 77 as programming language was mainly governed by its portability and interoperability with other languages. Many users employing more sophisticated languages like C or C++ have, at the very most, written a wrapper to encapsulate our solvers in their software. We indicate that our FGMRES package is slightly less popular than our GMRES [Frayssé et al. 2005; 2003]. We believe a main reason to be that GMRES is often used when FGMRES should (because the preconditioner is varied at each step, e.g., the preconditioning step is computed by solving iteratively an auxilary problem). However, GMRES still provides the user with a satisfactory solution thanks to its robustness to innacurate matrix-vector products [Bouras and Frayssé 2005; Giraud et al. 2007; Simoncini and Szyld 2003; van den Eshof and Sleijpen 2004].

2. THE FGMRES ALGORITHM

2.1 General Description

The generalized minimum residual (GMRES) method was proposed by Saad and Schultz [1986] in order to solve large, sparse, and non-Hermitian linear systems. GMRES belongs to the class of Krylov-based iterative methods. In 1993, Saad introduced a variant of the GMRES method with right preconditioning that enables the use of a different preconditioner at each step of the Arnoldi process [Saad 1993].

For sake of clarity and readability, we first describe the GMRES algorithm with right preconditioner and derive the FGMRES algorithm. Furthermore, for sake of generality we describe this method for linear systems that are complex; everything also specializes to real arithmetic calculation. Let A be a square nonsingular $n \times n$ complex matrix, and b a complex vector of length n, together defining the linear system

$$Ax = b\,. \tag{1}$$

Let M be a square nonsingular $n \times n$ complex matrix, we define the right preconditioned linear system

$$AMt = b\,, \tag{2}$$

where $x = Mt$ is the solution of the unpreconditioned linear system. Let $t_0 \in \mathbb{C}^n$ be an initial guess for this linear system and $r_0 = b - AMt_0$ be its corresponding residual.

The GMRES algorithm builds an approximation of the solution of Eq. (2) of the form

$$t_k = t_0 + V_k y, \tag{3}$$

where the columns of V_k form an orthonormal basis for the Krylov space of dimension $\mathcal{K}$ defined by

$$\mathcal{K}_k = \text{span}\left\{r_0, AMr_0, \ldots, (AM)^{k-1} r_0\right\},$$

and where y belongs to $\mathbb{C}^k$. The vector y is determined so that the 2-norm of the residual $r_k = b - AMt_k$ is minimal over $\mathcal{K}_k$.

The basis V_k for the Krylov subspace $\mathcal{K}_k$ is obtained via the well-known Arnoldi process. The orthogonal projection of A onto $\mathcal{K}_k$ results in an upper Hessenberg matrix $H_k = V_k^H A V_k$ of order k. The Arnoldi process satisfies the relationship

$$A[Mv_1, \cdots, Mv_k] = AMV_k = V_k H_k + h_{k+1,k} v_{k+1} e_k^H, \tag{4}$$

where e_k is the k^{th} canonical basis vector. Eq. (4) can be rewritten as

$$AMV_k = V_{k+1} \bar{H}_k,$$

where

$$\bar{H}_k = \begin{bmatrix} H_k \\ 0 \cdots 0 \; h_{k+1,k} \end{bmatrix}$$

is an $(k+1) \times k$ matrix.

Let $v_1 = r_0/\beta$, where $\beta = \|r_0\|_2$. The residual r_k associated with the approximate solution defined by Eq. (3) verifies

$$\begin{aligned} r_k &= b - AMt_k = b - AM(t_0 + V_k y) \\ &= r_0 - AMV_k y = r_0 - V_{k+1} \bar{H}_k y \\ &= \beta v_1 - V_{k+1} \bar{H}_k y \\ &= V_{k+1}(\beta e_1 - \bar{H}_k y). \end{aligned} \tag{5}$$

Since V_{k+1} is a matrix with orthonormal columns, the residual norm $\|r_k\|_2 = \left\|\beta e_1 - \bar{H}_k y\right\|_2$ is minimal when y solves the linear least-squares problem

$$\min_{y \in \mathbb{C}^k} \left\|\beta e_1 - \bar{H}_k y\right\|_2. \tag{6}$$

We will denote by y_k the solution of (6). Therefore, $t_k = t_0 + V_k y_k$ is an approximate solution of (2) for which the residual is minimal over $\mathcal{K}_k$. The GMRES method owes its name to this minimization property that is its key feature, as it ensures the decrease of the residual norm. We depict in Algortihm 1. the sketch of the modified Gram-Schmidt (MGS) variant of the GMRES method with right preconditioner.

In practice, the storage of the orthonormal basis V_k may become prohibitive. The restarted GMRES method is designed to cope with this memory drawback. Given a fixed m, the restarted GMRES method computes the sequence of approximate solutions t_j until t_j is acceptable or $j = m$. If the solution is not found,

Algorithm 1. Right Preconditioned GMRES

1: Choose a convergence threshold ε
2: Choose an initial guess t_0
3: $r_0 = b - AMt_0 = b$; $\beta = \|r_0\|$
4: $v_1 = r_0/\|r_0\|$;
5: **for** $k = 1, 2, \ldots$ **do**
6: $w = AMv_k$;
7: **for** $i = 1$ to k **do**
8: $h_{i,k} = v_i^H w$
9: $w = w - h_{i,k} v_i$
10: **end for**
11: $h_{k+1,k} = \|w\|$
12: $v_{k+1} = w/h_{k+1,k}$
13: Solve the least-squares problem $\min \|\beta e_1 - \bar{H}_k y\|$ for y
14: Exit if convergence is detected
15: **end for**
16: Set $x_m = M(t_0 + V_m y)$

then a new starting vector is chosen, on which GMRES is applied again. Often, GMRES is restarted from the last computed approximation, that is, $t_0 = t_m$, to ensure the monotonicity property of residual norms, even when restarting. The process is iterated until an accurate enough approximation is found. We will denote by GMRES(m) the restarted GMRES algorithm for a projection of size at most m. One possible benefit of using restarted GMRES is that it alleviates the cost of the orthogonalization procedure that can become very time consuming when the size of Krylov space becomes large.

If the preconditioner involved at step 6 in Algorithm 1. varies at each step, we can still write an equality similar to Eq. (4) as

$$\begin{aligned} A[M_1 v_1, \cdots, M_k v_k] &= A[z_1, \cdots, z_k] \\ &= AZ_k \\ &= V_k H_k + h_{k+1,k} v_{k+1} e_k^H \\ &= V_k \bar{H}_k, \end{aligned}$$

which enables us to get a relation similar to (5). Using $x_k = x_0 + Z_k y$, we have

$$\begin{aligned} r_k &= b - Ax_k = b - A(x_0 + Z_k y) \\ &= r_0 - AZ_k y = r_0 - V_{k+1} \bar{H}_k y \\ &= \beta v_1 - V_{k+1} \bar{H}_k y \\ &= V_{k+1}(\beta e_1 - \bar{H}_k y), \end{aligned}$$

where y is the solution of a least-squares problem similar to Eq. (6). Because this GMRES variant allows for flexible preconditioners, it is referred to as flexible GMRES. From an implementation point-of-view the main difference between right preconditioned GMRES and FGMRES lies in the memory requirement. In the latter algorithm, both V_k and Z_k need to be stored. We remind that only happy breakdowns might occur in GMRES (i.e., at step 11

Algorithm 2. Flexible GMRES

1: Choose a convergence threshold ε
2: Choose an initial guess x_0
3: $r_0 = b - Ax_0 = b$; $\beta = \|r_0\|$
4: $v_1 = r_0/\|r_0\|$;
5: **for** $k = 1, 2, \ldots$ **do**
6: $z_k = M_k v_k$; % M_k is the preconditioner used at step k
7: $w = Az_k$;
8: **for** $i = 1$ to k **do**
9: $h_{i,k} = v_i^H w$
10: $w = w - h_{i,k} v_i$
11: **end for**
12: $h_{k+1k} = \|w\|$
13: $v_{k+1} = w/h_{k+1,k}$
14: Solve the least-squares problem $\min \|\beta e_1 - \bar{H}_k y\|$ for y
15: Exit if convergence is detected
16: **end for**
17: Set $x_m = x_0 + Z_m y$

of Algorithm 1. if $h_{k+1,k}$ is zero, the algorithm would breakdown, but it does not care because this also means it has found the solution [Saad and Schultz 1986]). This is no longer true for FGMRES, which can break at step 12 before it has computed the solution. We describe the MGS variant of this method in Algorithm 2 and refer to Saad [1993] for a complete description of the convergence theory.

In the following paragraphs, we highlight the main points for FGMRES:

—the solution of the least-squares problem (6);

—the construction of the orthonormal basis V_k;

—stopping criteria for the iterative scheme; and

—the calculation of the residual at restart.

2.2 The Least-Squares Problem

At each step k of FGMRES, one needs to solve the least-squares problem (6). The matrix $\bar{H}_k$ involved in this least-squares problem is a $(k+1) \times k$ complex matrix which is upper Hessenberg. We wish to use an efficient algorithm for solving (6) which exploits the structure of $\bar{H}_k$.

First, we base the solution of (6) on the QR factorization of the matrix $[\bar{H}_k,\ \beta e_1]$: If $QR = [\bar{H}_k,\ \beta e_1]$, where Q is an orthonormal matrix and $R = (r_{i\ell})$ is a $(k+1) \times (k+1)$ upper triangular matrix, then the solution y_k of (6) is given by

$$y_k = R(1:k, 1:k)^{-1} R(1:k, k+1). \tag{7}$$

Here, $R(1:k, 1:k)$ denotes the $k \times k$ top left submatrix of R and $R(1:k, k+1)$ stands for the last column of R. Moreover, as V_{k+1} is an orthonormal matrix,

$$\|r_k\|_2 = \|b - Ax_k\|_2 = \left\|\beta e_1 - \bar{H}_k y_k\right\|_2 = \left|r_{k+1,k+1}\right|. \tag{8}$$

Therefore, the value of the norm of the residual of the linear system is a by-product value of the algorithm and can be obtained without explicitly evaluating the residual vector.

The QR factorization of upper Hessenberg matrices can be efficiently performed using Givens rotations, because they enable one to sequentially zero out all elements $\bar{H}_{\ell+1,\ell}$, $\ell = 1, \ldots k$. However, since $[\bar{H}_{k+1},\ \beta e_1]$ is obtained from $[\bar{H}_k,\ \beta e_1]$ by adding one column c, the R factor R_{k+1} of $[\bar{H}_{k+1},\ \beta e_1]$ is obtained by updating the R factor R_k of $[\bar{H}_k,\ \beta e_1]$ using an algorithm that we briefly outline now, for $k = 3$ [Bindel et al. 2002; Björck 1996; Blackford et al 2002].

(1) Let

$$R_k = \begin{pmatrix} + & + & + & + \\ 0 & + & + & + \\ 0 & 0 & + & + \\ 0 & 0 & 0 & + \end{pmatrix}$$

and $Q_k \in \mathbb{C}^{(k+1)\times(k+1)}$ be such that $[\bar{H}_k,\ \beta e_1] = Q_k R_k$. The matrix Q_k is not explicitly computed; only the sine and cosine of the Givens rotations are stored. The vector $w = Q_k^H c$ is then computed by applying the stored Givens rotations, and w is inserted in between the k and $k+1$ columns of R_k, to yield

$$\widetilde{R}_k = \begin{pmatrix} + & + & + & * & + \\ 0 & + & + & * & + \\ 0 & 0 & + & * & + \\ 0 & 0 & 0 & * & + \\ 0 & 0 & 0 & * & 0 \end{pmatrix}.$$

(2) A Givens rotation that zeros element $\widetilde{R}_k(k+2, k+1)$ is computed and applied to $\widetilde{R}_k$ to produce the matrix

$$R_{k+1} = \begin{pmatrix} + & + & + & + & + \\ 0 & + & + & + & + \\ 0 & 0 & + & + & + \\ 0 & 0 & 0 & + & + \\ 0 & 0 & 0 & 0 & + \end{pmatrix}.$$

The computation of the sine and cosine involved in the Givens QR factorization use the BLAS routines *ROTG.

In a parallel distributed environment this calculation, whose complexity is $\mathcal{O}(m)$, is performed redundantly on each processor so that no extra communication is required.

2.3 Computation of V_k

The orthogonality quality of the V_k plays a central role in FGMRES. It ensures that the convergence is not slowed or delayed. However, ensuring a very good orthogonality might be expensive and useless for some applications. Consequently, a tradeoff has to be found to balance the numerical efficiency of the orthogonalization scheme and its inherent efficiency on a given target computer.

Most of the time, the Arnoldi algorithm is implemented through the modified Gram-Schmidt process for the computation of V_k and H_k. In finite precision arithmetic, there might be a severe loss of orthogonality; this loss can be compensated by selectively iterating the orthogonalization scheme [Björck 1994; Hoffmann 1989], and the resulting QR factorization algorithm is called the iterative modified Gram-Schmidt (IMGS). The drawback of IMGS is the increased number of dot products.

The classical Gram-Schmidt (CGS) algorithm can be implemented in an efficient manner by gathering the dot products into one dense matrix-vector product, but it is well known that CGS is numerically worse than MGS. However, CGS with selective reorthogonalization (ICGS) results in an algorithm of the same numerical quality as IMGS. Therefore, ICGS is particularly attractive in a parallel distributed environment, where the global reduction involved in computation of the dot products is a well-known bottleneck [Frank and Vuik 1999; Frayssé et al. 1998; Lehoucq and Salinger 2001; Shadid and Tuminaro 1994].

In our FGMRES implementation, we have chosen to give the user the possibility of using any of the four different aforesaid schemes: CGS, MGS, ICGS and IMGS. We follow Rutishauser [1967] to define the criterion for the selective reorthogonalization and set $K = \sqrt{2}$ for the value for the threshold, as in Balay et al. [2004]; Maschho and Sorensen [1996].

2.4 Stopping Criteria

We have chosen to base our stopping criterion on the normwise backward error. The backward error analysis, introduced by Wilkinson [1963], is a powerful concept for analyzing the quality of an approximate solution that enjoys the following two properties.

(1) *Independent of the Details of Round-Off Propagation*. The errors introduced during the computation are interpreted in terms of perturbations of the initial data, and the computed solution is considered as exact for the perturbed problem.
(2) *Comparability*. Because round-off errors are seen as data perturbations, they can be compared with errors due to numerical approximations (consistency of numerical schemes) or to physical measurements (e.g., uncertainties on data coming from experiments).

The backward error defined by Eq. (9) measures the distance between the data of the initial problem and that of a perturbed problem. Dealing with such a distance both requires to choose data that is perturbed and a norm to quantify the perturbations. For the first choice, the matrix and the right-hand side of linear systems are natural candidates. In the context of linear systems, classical choices are the normwise and componentwise perturbations [Chaitin-Chatelin and Frayssé 1996; Higham 2002]. These choices lead to explicit formulas for the backward error (often a normalized residual), which is then easily evaluated. For iterative methods, it is generally admitted that the normwise model of perturbation is appropriate [Barrett et al. 1994].

Let x_k be an approximation to the solution $x = A^{-1}b$. Then for given α and β (we refer to the end of this section for a discussion on how to choose these parameters),

$$\begin{aligned}\eta(x_k) &= \min_{\Delta A, \Delta b} \{\varepsilon > 0 : \|\Delta A\|_2 \le \varepsilon\alpha,\ \|\Delta b\|_2 \le \varepsilon\beta \\ &\qquad\qquad \text{and } (A + \Delta A)x_k = b + \Delta b\} \\ &= \frac{\|b - Ax_k\|_2}{\alpha\|x_k\|_2 + \beta}\end{aligned} \tag{9}$$

is called the *normwise backward error* associated with x_k. It measures the norm of the smallest perturbations ΔA on A and Δb on b such that $(A + \Delta A)x_k = b + \Delta b$. The best one can require from an algorithm is a backward error of the order of the machine precision. In practice, the approximation of the solution is acceptable when its backward error is lower than the uncertainty of the data. Therefore, there is no gain in iterating after the backward error has reached machine precision (or data accuracy). Thanks to equality (8), we see that the 2-norm of the residual is given directly in the algorithm during the solution of the least-squares problem. Therefore, the backward error can be obtained at a low cost and we can use

$$\eta_A(x_k) = \frac{|r_{j+1,j+1}|}{\alpha\|x_k\|_2 + \beta}$$

as the stopping criterion of the FGMRES iterations. However, it is well known that in finite precision arithmetic, the computed residual (8) given from the Arnoldi process may differ significantly from the true residual. Therefore, it is not safe to exclusively use $\eta_A(x_k)$ as the stopping criterion. Our strategy is the following: First we iterate until $\eta_A(x_k)$ becomes lower than the tolerance, then we iterate until $\eta(x_k)$ itself becomes lower than the tolerance. We hope in this way to minimize the number of explicit residual computations (involving computation of matrix-vector products) necessary to evaluate $\eta(x_k)$, while still having a reliable stopping criterion. Finally, we mention that if our code is used in a parallel heterogeneous computing environment, because $\eta_A(x_k)$ is a by-product of the least-squares solution, processors using different arithmetics might detect a convergence while others do not. Such a situation would unfortunately lead to a deadlock as the different processes would no longer be performed within the SPMD (single-program multiple data) execution model assumed in our implementation. It was not our purpose to address the situation of parallel heterogeneous computing.

How to choose α and β ? Classical choices for α and β that appear in the literature are $\alpha = \|A\|_2$ and $\beta = \|b\|_2$. Any other choice that reflects the possible uncertainty in the data can also be plugged in. In our implementation, default values are used when the user's input is $\alpha = \beta = 0$. Table I explains the output information given to the user on the unpreconditioned linear system on return from FGMRES.

Table I. Stopping Criterion for the FGMRES Method

α	β	Information on the linear system
0	0	$\frac{\|Ax_k - b\|_2}{\|b\|_2}$
0	$\neq 0$	$\frac{\|Ax_k - b\|_2}{\beta}$
$\neq 0$	0	$\frac{\|Ax_k - b\|_2}{\alpha \|x_k\|_2}$
$\neq 0$	$\neq 0$	$\frac{\|Ax_k - b\|_2}{\alpha \|x_k\|_2 + \beta}$

2.5 Computation of the Residual at Restart

In some applications, the computation of each matrix-vector product can be extremely expensive, as, for instance, in some domain decomposition techniques, or in electromagnetism when a fast multipole expansion is used to evaluate the matrix-vector product. In such cases, one would like to avoid explicit calculation of the residual at each restart of GMRES. Since we then set $x_0 = x_m$, we have $r_0 = b - Ax_m$ with $x_m = x_0 + V_m y_m$. We can then observe that

$$\begin{aligned} r_0 &= b - A(x_0 + Z_m y_m) \\ &= V_{m+1}(\beta e_1 - \bar{H} y_m) \\ &= V_{m+1} Q_m (Q_m^H \beta e_1 - \begin{bmatrix} R(1:m, 1:m) \\ 0 \end{bmatrix} y_m) \\ &= V_{m+1} Q_m \begin{bmatrix} 0 \\ r_{m+1,m+1} \end{bmatrix}. \end{aligned}$$

It follows that the calculation of the residual amounts to computing a linear combination of the $(m+1)$ Arnoldi vectors. The coefficients of the linear combination are computed by applying the Givens rotations in reverse order to the vector which has all its entries equal to zero (except its last, which is equal to $r_{m+1,m+1}$). This nonzero value is a by-product of the solution of the least-squares problem. This calculation of the residual requires $n(2m+1)+2m$ floating-point operations (flops) and is to be preferred over an explicit calculation whenever the matrix-vector product involving A would use more than $2n(m+1)$ flops. We should mention that in some circumstances, such as when the required backward error is close to the machine precision, the use of this trick might slightly delay the convergence (although it might still enable one to get the solution within an overall shorter amount of computational time). Notice that the implementation of this trick requires storage of $(m+1)$ Arnoldi vectors, while only m have to be stored otherwise. For sake of robustness, even if this calculation of the residual is selected by the user, we enforce an explicit residual calculation if, in the previous restart, the convergence was detected by $\eta_A(x_k)$ but not assessed by $\eta(x_k)$.

REFERENCES

BALAY, S., BUSCHELMAN, K., EIJKHOUT, V., GROPP, W. D., KAUSHIK, D., KNEPLEY, M. G., MCINNES, L. C., SMITH, B. F., AND ZHANG, H. 2004. PETSc users manual. Tech. Rep. ANL-95/11 - Revision 2.1.5, Argonne National Laboratory.

BARRETT, R., BERRY, M., CHAN, T. F., DEMMEL, J., DONATO, J., DONGARRA, J., EIJKHOUT, V., POZO, R., ROMINE, C., AND DER VORST, H. V. 1994. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, 2nd ed. SIAM, Philadelphia, PA.

BATISTE, O., ALONSO, A., AND MERCADER, I. 2004. Hydrodynamic stability of binary mixtures in Bénard and thermogravitational cells. *J. Non-Equilib. Thermodyn. 29*, 359–375.

BINDEL, D., DEMMEL, J., KAHAN, W., AND MARQUES, O. 2002. On computing Givens rotations reliably and efficiently. *ACM Trans. Math. Softw. 28,* 2 (Jun.), 206–238.

BJÖRCK, Å. 1994. Numerics of Gram-Schmidt orthogonalization. *Linear Algebra Appl. 197-198*, 297–316.

BJÖRCK, Å. 1996. *Numerical Methods for Least Squares Problems*. SIAM, Philadelphia, PA.

BLACKFORD, L. S., DEMMEL, J., DONGARRA, J., DUFF, I., HAMMARLING, S., HENRY, G., HEROUX, M., KAUFMAN, L., LUMSDAINE, A., PETITET, A., POZO, R., REMINGTON, K., AND WHALEY, R. C. 2002. An updated set of basic linear algebra subprograms (BLAS). *ACM Trans. Math. Softw. 28,* 2 (Jun.), 135–151.

BOURAS, A. AND FRAYSSÉ, V. 2005. Inexact matrix-vector products in Krylov methods for solving linear systems: A relaxation strategy. *SIAM J. Matrix Anal. Appl. 26,* 23, 660–678.

CARPENTIERI, B., DUFF, I. S., GIRAUD, L., AND SYLVAND, G. 2005. Combining fast multipole techniques and an approximate inverse preconditioner for large electromagnetism calculations. *SIAM J. Sci. Comput. 27,* 3, 774–792.

CHAITIN-CHATELIN, F. AND FRAYSSÉ, V. 1996. *Lectures on Finite Precision Computations*. SIAM, Philadelphia, PA.

FRANK, J. AND VUIK, C. 1999. Parallel implementation of a multiblock method with approximate subdomain solution. *Appl. Numer. Math. 30*, 403–423.

FRAYSSÉ, V., GIRAUD, L., GRATTON, S., AND LANGOU, J. 2003. A set of GMRES routines for real and complex arithmetics on high performance computers. Tech. Rep. TR/PA/03/03, CERFACS, Toulouse, France. http://www.cerfacs.fr/algor.

FRAYSSÉ, V., GIRAUD, L., GRATTON, S., AND LANGOU, J. 2005. Algorithm 842: A set of GMRES routines for real and complex arithmetics on high performance computers. *ACM Trans. Math. Softw. 31,* 2, 228–238.

FRAYSSÉ, V., GIRAUD, L., AND KHARRAZ-AROUSSI, H. 1998. On the influence of the orthogonalization scheme on the parallel performance of GMRES. Tech. Rep. TR/PA/98/07, CERFACS, Toulouse, France. Preliminary version of the paper published in the *Proceedings of the EuroPar*. Lecture Notes in Computer Science, Springer, vol. 1470, 751–762.

GIRAUD, L., GRATTON, S., AND LANGOU, J. 2007. Convergence in backward error of relaxed GMRES. *SIAM J. Sci. Comput. 29,* 2, 710–728.

HIGHAM, N. J. 2002. *Accuracy and Stability of Numerical Algorithms*, 2nd ed. Society for Industrial and Applied Mathematics, Philadelphia, PA.

HOFFMANN, W. 1989. Iterative algorithms for Gram-Schmidt orthogonalization. *Comput. 41*, 335–348.

LEHOUCQ, R. B. AND SALINGER, A. G. 2001. Large-Scale eigenvalue calculations for stability analysis of steady flows on massively parallel computers. *Int. J. Numer. Methods Fluids 36*, 309–327.

LI, Z., SAAD, Y., AND SOSONKINA, M. 2003. pARMS: A parallel version of the recursive multilevel solver. *Numer. Linear Algebra Appl. 10*, 485–509.

MASCHHO, K. J. AND SORENSEN, D. C. 1996. A portable implementation of ARPACK for distributed memory parallel computers. In *Proceedings of the Copper Mountain Conference on Iterative Methods*.

MECA, E., MERCADER, I., BATISTE, O., AND REZ DE LA PISCINA, L. R. 2004a. A blue sky catastrophe in double-diffussive convection. *Phys. Rev. Lett. 92*, 23, 1–4.

MECA, E., MERCADER, I., BATISTE, O., AND REZ DE LA PISCINA, L. R. 2004b. Complex dynamics in double-diffusive convection. *Theor. Comput. Fluid Dynam. 18*, 231–238.

MERCADER, I., ALONSO, A., AND BATISTE, O. 2004. Numerical analysis of the Eckhaus instability in travelling-wave convection in binary mixtures. *Eur. Phys. J. E 15*, 319–333.

MERCADER, I., BATISTE, O., REZ DE LA PISCINA, L. R., RUIZ, X., RÜDIGER, S., AND CASADEMUNT, J. 2005. Bifurcations and chaos in single-roll natural convection with low Prandtl number. *Phys. Fluids*. submitted.

RUTISHAUSER, H. 1967. Description of algol 60. handbook for automatic computation. Springer, Berlin, 1.a.

SAAD, Y. 1993. A flexible inner-outer preconditioned GMRES algorithm. *SIAM J. Sci. Comput. 14*, 461–469.

SAAD, Y. AND SCHULTZ, M. 1986. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Statis. Comput. 7*, 856–869.

SHADID, J. N. AND TUMINARO, R. S. 1994. A comparison of preconditioned nonsymmetric Krylov methods on a large-scale MIMD machine. *SIAM J. Sci. Comput. 14,* 2, 440–459.

SIMONCINI, V. AND SZYLD, D. B. 2003. Theory of inexact Krylov subspace methods and applications to scientific computing. *SIAM J. Sci. Comput. 25*, 454–477.

STRAKOŠ, Z. AND TICHY, P. 2006. Error estimation in preconditioned conjugate gradients. *BIT*. accepted.

TUMINARO, R. S., HEROUX, M., HUTCHINSON, S. A., AND SHADID, J. 1999. Official Aztec user's guide - version 2.1. Tech. Rep. 99-8801J, Sandia National Laboratories.

VAN DEN ESHOF, J. AND SLEIJPEN, G. L. G. 2004. Inexact Krylov subspace methods for linear systems. *SIAM J. Matrix Anal. Appl. 26,* 1, 125–153.

WARSA, J. S., BENZI, M., WAREING, T. A., AND MOREL, J. E. 2004. Preconditioning a mixed discontinuous finite element method for radiation diffusion. *Numer. Linear Algebra Appl. 11*, 795–811.

WILKINSON, J. H. 1963. *Rounding Errors in Algebraic Processes*, vol. 32. Her Majesty's Stationery Office, London, UK.

Received February 2006; revised August 2007; accepted October 2007