Algorithm 842: A Set of GMRES Routines for Real and Complex Arithmetics on High Performance Computers

VALÉRIE FRAYSSÉ
Kvasar Technology LLC
LUC GIRAUD and SERGE GRATTON
CERFACS
and
JULIEN LANGOU
University of Tennessee, Knoxville

In this article we describe our implementations of the GMRES 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 GMRES 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 the Krylov methods. Either implicit or explicit calculation of the residual at restart are 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.1.4 [**Numerical Analysis**]: Numerical Linear Algebra—*Sparse, structured and very large systems (direct and iterative methods)*

General Terms: Algorithms

Additional Key Words and Phrases: Linear systems, Krylov methods, GMRES, reverse communication, distributed memory

This work was performed while V. Frayssé and J. Langou were researchers at CERFACS.
Authors' addresses: V. Frayssé, Kvasar Technology LLC, 372 Mariborough Street, No. 10, Boston, MA 02115; email: valerie@spydre.com; L. Giraud and S. Gratton, CERFACS, 42 Avenue Gaspard Coriolis, 31057 Toulouse Cedex, France; email: {giraud,gratton}@cerfacs.fr; J. Langou, The University of Tennessee, Department of Computer Science, 1122 Volunteer Blvd., Knoxville, TN 37996-3450; email: langou@cs.utk.edu.

1. INTRODUCTION

The GMRES (Generalized Minimum Residual) method [Saad and Schultz 1986] is among the most widely used Krylov solvers for the iterative solution of general large linear systems. 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 to 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 we decided to develop our own implementation with the objective 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 preconditioner location, orthogonalization schemes, restarting strategies and stopping criteria. Our implementation enables the user to choose the location of the preconditioner, which can be on the left, on the right or on both sides. Four variants of the Gram-Schmidt orthogonalization procedure are implemented, which permits 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 because the cost of the orthogonalization becomes prohibitive, a restart should be performed and a new initial residual should be computed. To address the situations where the matrix-vector product is cheap, as in finite element calculation for instance, or expensive, as in boundary element calculation where the matrix is dense and is often approximated via a fast multipole technique, the user can select either an explicit or an implicit calculation of the residual. Finally, the stopping criterion is a key component for the 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, which are commonly admitted as relevant for iterative solvers. The purpose of this article is to present the API of the GMRES routines and to describe several choices that have been made in order to get an efficient and reliable implementation of the GMRES [Saad and Schultz 1986] algorithm suitable for real and complex arithmetic on any scientific computer. The package was first made available in the public domain in 1997 and has so far been downloaded more than a thousand times both by academic and research institutions and by industrial companies. The most popular arithmetics are the double real and the double complex. This latter seems to be fairly popular in the scientific community of wave propagation as illustrated by the list of papers in that field that reference our package [Buchau and Rucker 2002; Hustedt et al. 2003;

monga Made 2001; monga Made and Beauwens 2000; monga Made et al. 2000; Sylvand 2002; West and Sturm 1999; Yoshida et al. 2000, 2001]. The selection of Fortran 77 as programming language was mainly governed by its qualities of portability and interoperability with other languages. Many users using more sophisticated languages like C or C++ have written a wrapper to encapsulate our solvers in their software [Cross et al. 1999; Lewis et al. 1997; Masters et al. 1997; Warsa et al. 2004]. Finally, to illustrate the success of our package we should point out that it has been appearing for many months in the top 10 items returned by Google when a search is performed on the word "GMRES."

2. THE GMRES ALGORITHM

2.1 General Description

The Generalized Minimum RESidual (GMRES) method was proposed by Saad and Schultz in 1986 [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.

For the sake of generality we describe this method for linear systems that are complex, and intended for real arithmetic calculation. Let A be a square nonsingular $n \times n$ complex matrix, and b be a complex vector of length n, defining the linear system

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

to be solved. Let $x_0 \in \mathbb{C}^n$ be an initial guess for this linear system and $r_0 = b - Ax_0$ be its corresponding residual.

The GMRES algorithm builds an approximation of the solution of (1) of the form

$$x_m = x_0 + V_m y \tag{2}$$

where V_m is an orthonormal basis for the Krylov space of dimension m defined by

$$\mathcal{K}_m = \text{span}\{r_0, Ar_0, \ldots, A^{m-1}r_0\},$$

and where y belongs to $\mathbb{C}^m$. The vector y is determined so that the 2-norm of the residual $r_m = b - Ax_m$ is minimal over $\mathcal{K}_m$.

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

$$AV_m = V_m H_m + h_{m+1,m} v_{m+1} e_m^H, \tag{3}$$

where e_m is the mth canonical basis vector. Equation (3) can be rewritten as

$$AV_m = V_{m+1} \bar{H}_m$$

where

$$\bar{H}_m = \begin{bmatrix} H_m \\ 0 \cdots 0 \; h_{m+1,m} \end{bmatrix}$$

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

Let $v_1 = r_0/\beta$ where $\beta = \|r_0\|_2$. The residual r_m associated with the approximate solution (2) verifies

$$\begin{aligned} r_m &= b - Ax_m = b - A(x_0 + V_m y) \\ &= r_0 - AV_m y = r_0 - V_{m+1}\bar{H}_m y \\ &= \beta v_1 - V_{m+1}\bar{H}_m y \\ &= V_{m+1}(\beta e_1 - \bar{H}_m y). \end{aligned}$$

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

$$\min_{y \in \mathbb{C}^m} \|\beta e_1 - \bar{H}_m y\|_2. \tag{4}$$

We will denote by y_m the solution of (4). Therefore, $x_m = x_0 + V_m y_m$ is an approximate solution of (1) for which the residual is minimal over $\mathcal{K}_m$. GMRES owes its name to this minimization property that is its key feature since it ensures the decrease of the residual norm.

In exact arithmetic, GMRES needs a number of steps equal to the degree of the minimal polynomial of $r_0 = Ax_0 - b$ associated with A. Even if this degree is theoretically lower than or equal to n, it can be very large in practice, and the storage of the orthonormal basis V_m 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 sequel of approximate solutions x_j until x_j is acceptable or $j = m$. If the solution was not found, then a new starting vector is chosen on which GMRES is applied again. Often, GMRES is restarted from the last computed approximation, $x_0 = x_m$ to insure the monotonicity property of the residual norms even when restarting. The process is iterated until a good enough approximation is found. We will denote by GMRES(m) the restarted GMRES algorithm for a projection size of at most m. One possible benefit of using restarted GMRES is that it alleviates the cost of the orthogonalization procedure, which can become very time consuming when the size of the Krylov space becomes large.

In the following paragraphs, we highlight the key points for GMRES:

—the solution of the least-squares problem (4),

—the construction of the orthonormal basis V_m,

—the stopping criteria for the iterative scheme, and

—the calculation of the residual at the restart.

2.2 The Least-Squares Problem

At each step j of GMRES, one needs to solve the least-squares problem (4). The matrix $\bar{H}_j$ involved in this least-squares problem is a $(j+1) \times j$ complex matrix which is upper Hessenberg. We wish to use an efficient algorithm for solving (4) that exploits the structure of $\bar{H}_j$.

First, we base the solution of (4) on the QR factorization of the matrix $[\bar{H}_j,\ \beta e_1]$: if $QR = [\bar{H}_j,\ \beta e_1]$ where Q is an orthonormal matrix and $R = (r_{ik})$ is a $(j+1)\times(j+1)$ upper triangular matrix, then the solution y_j of (4) is given by

$$y_j = R(1:j, 1:j)^{-1} R(1:j, j+1). \tag{5}$$

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

$$\|r_j\|_2 = \|b - Ax_j\|_2 = \|\beta e_1 - \bar{H}_j y_j\|_2 = |r_{j+1,j+1}|. \tag{6}$$

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 $\bar{H}_{k+1,k}$, $k = 1, \dots j$. However, since $[\bar{H}_{j+1},\ \beta e_1]$ is obtained from $[\bar{H}_j,\ \beta e_1]$ by adding one column c, the R factor R_{j+1} of $[\bar{H}_{j+1},\ \beta e_1]$ is obtained by updating the R factor R_j of $[\bar{H}_j,\ \beta e_1]$ using an algorithm that we briefly outline now, for $j = 3$ (see Bindel et al. [2002]; Björck [1996]; Blackford et al. [2002]):

(1) Let

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

and $Q_k \in C^{(j+1)\times(j+1)}$ be such that $[\bar{H}_j,\ \beta e_1] = Q_k R_k$. 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 j and $j+1$ columns of R_k, to yield

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

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

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

The computation of the sine and cosine involved in the givens QR factorization use the BLAS routines *ROTG, and we refer the reader to Bindel et al. [2002] and Blackford et al. [2002] for questions related to the reliability of these transformations.

2.3 Computation of V_j

The orthogonality quality of the V_j plays a central role in GMRES. It ensures that the convergence is not slowed down or delayed. However, ensuring a very good orthogonality might be expensive and useless for some applications. Consequently a trade-off 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 (MGS) process for the computation of V_m and H_m. In finite precision arithmetic, there might be a severe loss of orthogonality in the computed basis, which does not however prevent the method from being backward stable; this loss can be compensated for by selectively iterating the orthogonalization scheme [Björck 1994; Hoffmann 1989]. The resulting QR factorization algorithm is called 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 and gives rise to a GMRES implementation that may not be backward stable. 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 the computation of the dot products is a well-known bottleneck [Lehoucq and Salinger 2001; Frank and Vuik 1999; Frayssé et al. 1998; Shadid and Tuminaro 1994].

In our GMRES implementation, we have chosen to give the user the choice of any of the four different schemes quoted above: CGS, MGS, ICGS and IMGS. We follow Rutishauser [1967] in defining the criterion for the selective reorthogonalization and set $K = \sqrt{2}$ as suggested by Daniel et al. [1976] as the value for the threshold.

2.4 Preconditioning

The convergence of GMRES or GMRES(m) to solve (1) might be slow. To overcome this drawback, one often prefers to solve a transformed linear system that is referred to as the preconditioned linear system. More precisely if $A \approx M_1 M_2$ we actually solve the linear system

$$M_1^{-1} A M_2^{-1} z = M_1^{-1} b \tag{7}$$

with $x = M_2^{-1} z$. In our implementation we allow the user to select left and/or right preconditioning. The use of preconditioners has some consequences for the stopping criterion. We discuss these points in the next paragraph.

2.5 Stopping Criteria

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

(1) it is 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) 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 (uncertainties on data coming from experiments for instance).

The backward error defined by (8) measures the distance between the data of the initial problem and those of a perturbed problem. Dealing with such a distance both requires choosing the data that are perturbed and a norm to quantify the perturbations. For the first choice, the matrix and the right-hand side of the linear systems are natural candidates. In the context of linear systems, classical choices are the normwise and the 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_j be an approximation to the solution $x = A^{-1}b$. Then for α and β given (we refer to the end of this section for a discussion on how to choose those parameters),

$$\begin{aligned}\eta(x_j) &= \min_{\Delta A, \Delta b}\{\varepsilon > 0 : \|\Delta A\|_2 \leq \varepsilon\alpha,\ \|\Delta b\|_2 \leq \varepsilon\beta \text{ and } (A + \Delta A)x_j = b + \Delta b\} \qquad (8)\\ &= \frac{\|b - Ax_j\|_2}{\alpha\|x_j\|_2 + \beta}\end{aligned}$$

is called the *normwise backward error* associated with x_j. It measures the norm of the smallest perturbations ΔA on A and Δb on b such that $(A + \Delta A)x_j = b + \Delta b$. 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 (6), 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

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

as the stopping criterion of the GMRES iterations. However, it is well-known that, in finite precision arithmetic, the computed residual (6) given from the Arnoldi process may differ significantly from the true residual. Therefore, it is not safe to exclusively use $\eta_A(x_j)$ as the stopping criterion. Our strategy is the following: first we iterate until $\eta_A(x_j)$ becomes lower than the tolerance, then afterwards, we iterate until $\eta(x_j)$ itself becomes lower than the tolerance. We hope in this way to minimize the number of explicit residual computations (involving the computation of matrix-vector products) necessary to evaluate $\eta(x_j)$, while still having a reliable stopping criterion.

When GMRES is used in conjunction with preconditioning, then our stopping criterion is based on the backward error for the preconditioned system (7):

$$\eta^P(x_j) = \|M_1^{-1}AM_2^{-1}z_j - M_1^{-1}b\|_2/(\alpha^P\|x_j\|_2 + \beta^P)$$

with $x_j = M_2^{-1}z_j$. We denote by

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

the stopping criterion for the preconditioned GMRES. As previously, we stop the iterations when the computed values of $\eta_A^P(x_j)$ and then $\eta^P(x_j)$ satisfy the prescribed tolerance. We prefer to stop the iterations on the preconditioned linear system and not on the original linear system because the residual, which is readily available in the algorithm is that of the *preconditioned* system. It would be too expensive to compute the residual of the unpreconditioned system at each iteration. For the user's information, we also give the value of the backward error for the unpreconditioned system on return from the solver.

We should notice that for a right preconditioner $\eta = \eta^P$ (or $\eta_A = \eta_A^P$): this is the reason why right preconditioning is often preferred in many applications. Otherwise there is no a priori relationship between the backward error of the preconditioned system and that of the unpreconditioned system. Nevertheless, we noticed in our experiments that η (or η_A) is usually smaller than η^P (or η_A^P). It is therefore recommended to use a larger tolerance for the preconditioned system than one would have used on the unpreconditioned one.

How to choose α, β, α^P and β^P ? Classical choices for α and β that appear in the literature are $\alpha = \|A\|_2$ and $\beta = \|b\|_2$. Similarly, α^P and β^P should be chosen such as $\alpha^P \sim \|M_1^{-1}A\|_2$ and $\beta^P \sim \|M_1^{-1}b\|_2$. 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 $\alpha = \beta = 0$ or $\alpha^P = \beta^P = 0$. Table I lists the stopping criteria for different choices of α^P and β^P. Similarly, Table II explains the output information given to the user on the unpreconditioned linear system on return from GMRES.

2.6 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 that case, one would like to avoid the explicit

Table I. Stopping Criterion for the Preconditioned GMRES Method

α^P	β^P	Stopping Criterion
0	0	$\dfrac{\left\lVert M_1^{-1}AM_2^{-1}z_j - M_1^{-1}b\right\rVert_2}{\left\lVert M_1^{-1}b\right\rVert_2}$
0	$\neq 0$	$\dfrac{\left\lVert M_1^{-1}AM_2^{-1}z_j - M_1^{-1}b\right\rVert_2}{\beta^P}$
$\neq 0$	0	$\dfrac{\left\lVert M_1^{-1}AM_2^{-1}z_j - M_1^{-1}b\right\rVert_2}{\alpha^P \lVert x_j \rVert_2}$
$\neq 0$	$\neq 0$	$\dfrac{\left\lVert M_1^{-1}AM_2^{-1}z_j - M_1^{-1}b\right\rVert_2}{\alpha^P \lVert x_j \rVert_2 + \beta^P}$

Table II. Stopping Criterion for the Unpreconditioned GMRES Method

α	β	Information on the Unpreconditioned System
0	0	$\dfrac{\lVert Ax_j - b\rVert_2}{\lVert b\rVert_2}$
0	$\neq 0$	$\dfrac{\lVert Ax_j - b\rVert_2}{\beta}$
$\neq 0$	0	$\dfrac{\lVert Ax_j - b\rVert_2}{\alpha\lVert x_j\rVert_2}$
$\neq 0$	$\neq 0$	$\dfrac{\lVert Ax_j - b\rVert_2}{\alpha\lVert x_j\rVert_2 + \beta}$

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 $x_m = x_0 + V_m y$. We can then observe that

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

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 the reverse order to the vector that has all its entries equal to zero except the last, which is equal to $r_{m+1,m+1}$. This non-zero 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 to 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, for instance 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 us to get the solution within an overall shorter amount of computational time). Notice that the implementation of this trick requires the storae of $(m+1)$ Arnoldi vectors, while only m have to be stored otherwise. For the 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 $\eta_A^P(x_j)$ but not assessed by $\eta^P(x_j)$.

For more details on the implementation and the application protocol interface we also refer to Frayssé et al. [2003].

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*, Second ed. SIAM, Philadelphia, PA.

BINDEL, D., DEMMEL, J., KAHAN, W., AND MARQUES, O. 2002. On computing givens rotations reliably and efficiently. *ACM Transactions on Mathematical Software (TOMS) 28*, 2 (June), 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.

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. (TOMS) 28*, 2 (June), 135–151.

BUCHAU, A. AND RUCKER, W. M. 2002. Preconditioned fast adaptive multipole boundary element method. *IEEE Trans. Magn. 38*, 2, 461–464.

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

CROSS, J. T., MASTERS, I., AND LEWIS, R. W. 1999. Why you should consider object-oriented programming techniques for finite element methods. *Int. J. Numer. Meth. Heat Fluid Flow 9*, 333–347.

DANIEL, W., GRAGG, W. B., KAUFMAN, L., AND STEWART, G. W. 1976. Reorthogonalization and stable algorithms for updating the Gram-Schmidt QR factorization. *Math. Comp. 30*, 772–795.

FRANK, J. AND VUIK, C. 1999. Parallel implementation of a multiblock method with approximate subdomain solution. *Appl. Num. 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. Available on http://www.cerfacs.fr/algor.

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 EuroPar'98. *Lecture Notes in Computer Science, Springer-Verlag*, vol. 1470, pp. 751–762.

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

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

HUSTEDT, B., OPERTO, S., AND VIRIEUX, J. 2003. A multi-level direct-iterative solver for seismic wave propagation modelling: space and wavelet approaches. *Geophys. Int. J. 155*, 953–980.

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. Meth. Fluids 36*, 309–327.

LEWIS, R. W., MASTERS, I., AND CROSS, J. T. 1997. Automatic timestep selection for the super-time-stepping acceleration on unstructured grids using object-oriented programming. *Comm. Numer. Meth. Eng. 13*, 4, 249–260.

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

MASTERS, I., USMANI, A. S., CROSS, J. T., AND LEWIS, R. W. 1997. Finite element analysis of solidification using object-oriented and parallel techniques. *Int. J. Numer. Meth. Eng. 40*, 15, 2891–2909.

MONGA-MADE, M. M. 2001. Incomplete factorization based preconditionings for solving the Helmholtz equation. *Int. J. Numer. Meth. Eng. 50*, 5, 1077–1101.

MONGA-MADE, M. M. AND BEAUWENS, R. 2000. Imaginary diagonal relaxations for highly indefinite linear systems. In *Proceedings of the 16th IMACS World Congress 2000*. IMACS, Department of Computer Science, Rutgers University.

MONGA-MADE, M. M., BEAUWENS, R., AND WARZÉE, G. 2000. Preconditioning of discrete Helmholtz operators perturbed by a diagonal complex matrix. *Comm. Numer. Meth. Eng. 16*, 2, 801–817.

RUTISHAUSER, H. 1967. *Description of Algol 60*. Handbook for Automatic Computation, vol. 1. Springer-Verlag, Berlin.

SAAD, Y. AND SCHULTZ, M. 1986. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. 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. Comp. 14*, 2, 440–459.

SYLVAND, G. 2002. Résolution itérative de formulation intégrale pour Helmholtz 3D: Applications de la méthode multipôle à des problèmes de grande taille. Ph.D. thesis, Ecole Nationale des Ponts et Chaussées.

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.

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

WEST, J. C. AND STURM, J. M. 1999. On iterative approaches for electromagnetic rough-surface scattering problems. *IEEE Trans. Antennas and Propagation 47*, 8, 1281–1288.

WILKINSON, J. H. 1963. *Rounding errors in algebraic processes*. Vol. 32. Her Majesty's stationery office, London.

YOSHIDA, K., NISHIMURA, N., AND KOBAYASHI, S. 2000. Analysis of three dimensional scattering of elastic waves by cracks with fast multipole boundary integral equation method. *J. Appl. Mech. JSCE 3*, 145–150.

YOSHIDA, K., NISHIMURA, N., AND KOBAYASHI, S. 2001. Analysis of three dimensional scattering of scalar waves by cracks with fast multipole boundary integral equation method. *Trans JSME A*, 16–22.

Received July 2003; revised December 2003, December 2004; accepted December 2004