GMRES Acceleration of Computational Fluid Dynamics Codes

L. B. Wigton,* N. J. Yu,* and D. P. Young**
The Boeing Company
Seattle, Washington

Abstract

The generalized minimal residual algorithm (GMRES) is a conjugate-gradient like method that applies directly to nonsymmetric linear systems of equations. In this paper, GMRES is modified to handle nonlinear equations characteristic of computational fluid dynamics. Attention is devoted to the concept of preconditioning and the role it plays in assuring rapid convergence. A formulation is developed that allows GMRES to be preconditioned by the solution procedures already built into existing computer codes. Examples are provided that demonstrate the ability of GMRES to greatly improve the robustness and rate of convergence of current state-of-the-art fluid dynamics codes. Theoretical aspects of GMRES are presented that explain why it works. Finally, the advantages GMRES enjoys over related methods such as conjugate gradients are discussed.

1. Introduction

Conjugate-gradient like methods have long been used to solve problems involving symmetric positive definite matrices (refs. 1,2). When preconditioned by other solution procedures such as line relaxation or approximate factorization schemes, conjugate-gradient methods provide rapid and reliable convergence. Moreover, the rate of convergence is often remarkably insensitive to the choice of relaxation parameters employed by the preconditioner.

In the past a few attempts have been made to use conjugate-gradient like methods to solve problems in computational fluid dynamics (CFD) (refs. 3,4,5). However, difficulties have been encountered in applying these methods to the nonsymmetric matrices that arise in CFD. Also, the preconditioning techniques employed during previous efforts did not lend themselves to immediate implementation in existing codes. As a result, conjugate-gradient like methods have not gained widespread use in the CFD community.

Recently, conjugate-gradient methods have been extended to apply directly to nonsymmetric matrices (ref. 6). A particularly effective variant is the generalized minimal residual algorithm (GMRES) developed by Saad and Schultz at Yale University (ref. 7). Given a large linear system of N equations in N unknowns, GMRES finds the best possible solution over a carefully constructed k dimensional subspace where k is a moderate number, say 20. The storage required by our nonlinear version of GMRES is large, roughly (k+4)*N. However, the new generation of supercomputers offers much larger memories than the machines they are replacing. For example, the new CRAY X-MP delivered to Boeing with a 128 million word solid state disk (SSD) effectively has 65 times the memory capacity of the CRAY-1/S it replaced. In view of the fact that the performance of many current solution algorithms degrades significantly when larger problems are considered, it may not be a bad idea to trade some of the increased memory capacity afforded by the new generation of supercomputers for increased speed and reliability of convergence.

* Lead Engineers, Members, AIAA
** Senior Specialist Engineer

The purpose of this paper is to serve as an introduction to the GMRES algorithm, to demonstrate its usefulness in CFD codes, and to encourage further research into GMRES and similar algorithms. Additional background information concerning computational linear algebra related to GMRES is readily available in the literature (refs. 8,9).

2. GMRES Algorithm

As described in the literature (ref. 7), GMRES is a method for solving nonsymmetric linear systems of equations. A modified version of GMRES that applies to nonlinear systems of equations will now be presented.

Consider a differentiable system

$$F(u) = 0 \tag{2.1}$$

of N nonlinear equations in N unknowns. The differential $\bar{F}(u;p)$ of F at u in the direction p is defined by

$$\bar{F}(u;p) = \lim_{\varepsilon \to 0} \frac{F(u + \varepsilon p) - F(u)}{\varepsilon} . \tag{2.2}$$

For computational purposes, we estimate $\bar{F}(u;p)$ by taking ε to be some small number and ensuring that the variables u and the component values of F(u) are reasonably scaled to permit an accurate evaluation.

Given u^n, an approximate solution to equation (2.1), one cycle of GMRES advances the solution by first choosing k orthonormal search directions $p_1, p_2, \ldots p_k$ as follows:

$$p_1 = F(u^n). \tag{2.3}$$

Normalize p_1

$$p_1 = p_1 / ||p_1||. \tag{2.4}$$

For j=1,2,...k−1 take

$$p_{j+1} = \bar{F}(u^n;p_j) - \sum_{i=1}^{j} b_{ij} p_i \tag{2.5}$$

where

$$b_{ij} = (\bar{F}(u^n;p_j), p_i)$$

so that

$$(p_{j+1}, p_i) = 0 \text{ for } i=1,2,\ldots j.$$

Normalize p_{j+1}

$$p_{j+1} = p_{j+1} / ||p_{j+1}||. \tag{2.6}$$

Now update u^n using

$$u^{n+1} = u^n + \sum_{j=1}^{k} a_j p_j, \tag{2.7}$$

where the coefficients a_j are chosen to minimize

$$||F(u^{n+1})||^2$$

$$= ||F(u^n + \sum_{j=1}^{k} a_j p_j)||^2$$

$$\cong ||F(u^n) + \sum_{j=1}^{k} a_j \bar{F}(u^n;p_j)||^2. \quad (2.8)$$

To ensure the overall success of GMRES, this least squares problem must be solved by a numerically stable procedure such as the QR algorithm available in LINPACK (ref. 10). In terms of storage and operation count, the version of the QR algorithm presented in Reference 7 is especially effective for solving this particular least squares problem.

Our implementation of the GMRES algorithm requires storage for the k search directions $p_1,p_2,...p_k$, and for $\bar{F}(u^n;p_k)$. Equation (2.5) is used to eliminate the need for storing $\bar{F}(u^n;p_j)$ for j=1,2,...k–1 (instead we gladly store the b_{ij} since k is much smaller than N). In order to use equation (2.2) to evaluate $\bar{F}(u^n;p_j)$ as needed, we provide storage for the three vectors $F(u^n)$, $u^n+\varepsilon p_j$, and $F(u^n+\varepsilon p_j)$. Thus the total memory requirement for GMRES(k) (GMRES with k search directions) is essentially (k+4)*N.

The operations required by GMRES are SDOT and SAXPY, which are part of the Basic Linear Algebra Subroutines (BLAS) available in LINPACK (ref. 10). SDOT forms the dot product of two vectors and SAXPY multiplies the vector by a scalar and adds it to another vector. Both SDOT and SAXPY are highly vectorizable. Since GMRES uses these operations over long vector lengths (equal to the number of unknowns in the problem), GMRES runs extremely well on vector computers. Also, since the information required by GMRES can be stored contiguously, it runs efficiently on virtual memory machines as well.

One cycle of the GMRES algorithm is an approximation to one cycle of Newton's iteration. Indeed with Newton's method one uses the linear approximation

$$F(u^n + p) \cong F(u^n) + \bar{F}(u^n;p) \quad (2.9)$$

to estimate a value of p which will enable $F(u^n + p) = 0$. Equation (2.9) suggests that we solve the following linear equation for p:

$$F(u^n) + \bar{F}(u^n;p) = 0. \quad (2.10)$$

The naive way to do this is to compute the entries of the NxN matrix associated with $\bar{F}$ and directly solve the system of linear equations. This is enormously expensive if N is at all large (as it is for most problems of practical interest). GMRES approximately solves equation (2.10) by finding the best possible solution over the k dimensional linear subspace spanned by the search directions $<p_1,p_2,...p_k>$. Of course if k=N, then GMRES finds the best possible solution to equation (2.10) over the whole space and therefore computes the exact solution. Unfortunately this is every bit as expensive as solving equation (2.10) directly. The key to efficiency is to arrange for GMRES to find a good solution to equation (2.10) using only a small number of search directions k. Preconditioning plays a vital role in achieving this goal.

3. Preconditioning

The available mathematical theorems (section 5) as well as much numerical experience indicate that the rate at which GMRES converges, measured by the value of k required to achieve a given level of accuracy, depends on the distribution of eigenvalues. The more the eigenvalues are clustered together, the faster GMRES will converge. The process of replacing a given problem with another equivalent problem (i.e. one with the same solution) enjoying a more favorable distribution of eigenvalues is called preconditioning.

Based on the observation that the identity operator has the most favorable distribution of eigenvalues (all the eigenvalues are clustered at 1), most methods for preconditioning invoke an approximate inverse to the operator in the equation one is solving. For example, if L is a linear operator and N^{-1} is an approximate inverse to L then the problem

$$L(x) - b = 0 \quad (3.1)$$

is equivalent to

$$N^{-1}(L(x) - b) = 0. \quad (3.2)$$

These equations have the same solution, but GMRES will more readily solve equation (3.2) than equation (3.1) because the eigenvalues of $N^{-1}L$ are more tightly clustered than those of L.

A formulation that allows GMRES to take advantage of preconditioners already built into existing codes will now be described.

Given a problem of the form

$$F(u) = 0, \quad (3.3)$$

most computer codes have a method M that takes a good approximation to the solution of equation (3.3) and creates a better approximation. Typical methods M might for example involve SLOR, ADI, a time marching scheme, or even multigrid. Whatever the method, M already invokes an approximate inverse to the operator in equation (3.3). The standard procedure for updating u is

$$u^{n+1} = M(u^n). \quad (3.4)$$

Convergence is achieved when $u^{n+1} = u^n$. Thus, solving equation (3.3) is equivalent to solving

$$u - M(u) = 0. \quad (3.5)$$

Applying GMRES to the preconditioned equation (3.5) is more effective than applying GMRES to equation (3.3) directly. As we shall see, in the examples of the next section, applying GMRES to equation (3.5) is also often considerably more effective than employing the standard iteration procedure equation (3.4).

4. Computational Results

In order to illustrate some of concepts presented above, consider the application of GMRES to the following problem. The equation

$$A\,U_{xx} + B\,U_{yy} + U_{zz} = 0 \quad (4.1)$$

is solved using the standard central difference discretization on a 50 by 50 by 20 uniform mesh with Dirichlet boundary conditions. The coefficients A and B vary randomly over space and lie between .001 and 1000. The preconditioning methods (denoted by M in the previous section) tested include point Gauss Seidel and line relaxation in one, two, and three directions. We arbitrarily choose to use 10 search directions in GMRES. The residual errors are plotted versus total CPU time in seconds (on the CRAY-1/S) in Figures 1 and 2. As can be seen, in all cases applying GMRES to equation (3.5) works much better than the standard iteration equation (3.4). In Figure 2, one notices that using GMRES directly with no preconditioner is a very poor strategy. However, using GMRES with a line relaxation preconditioner is very effective. In terms of total CPU time GMRES preconditioned with line relaxation in two directions is the most effective strategy of those tested. As a matter of interest in Figure 3 we plot the residual error versus the number of function calls (number of evaluations of equation (3.5)). As expected, GMRES preconditioned by line relaxation in all three directions requires the fewest

number of function evaluations. However, this was offset by the added expense of sweeping in three directions as compared to sweeping in just two directions. In general, the improved convergence rate produced by more elaborate preconditioners must be weighed against their cost.

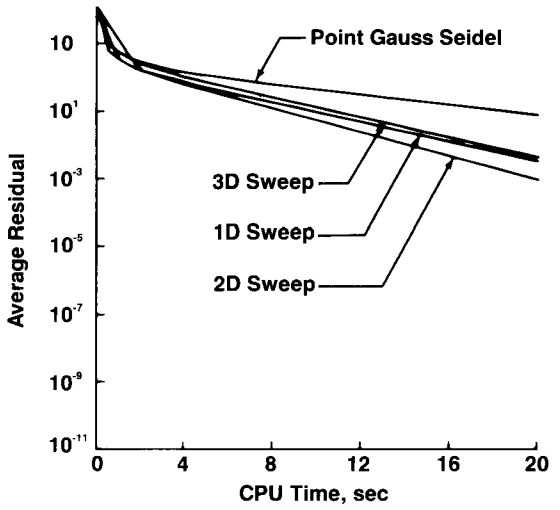

Figure 1. Convergence Histories in Terms of CPU Time for A U_{xx} + B U_{yy} + U_{zz} = 0 Using Point Gauss Seidel and Line Relaxation in One, Two, and Three Directions

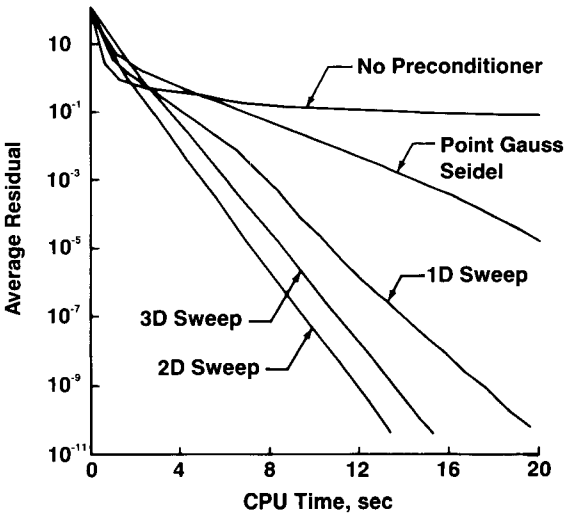

Figure 2. Convergence Histories in Terms of CPU Time for A U_{xx} + B U_{yy} + U_{zz} = 0 Using GMRES Preconditioned by Point Gauss Seidel and Line Relaxation in One, Two, and Three Directions

Application of GMRES to equation (3.5) is very easy to implement in existing codes. Accordingly, GMRES has been introduced into several contemporary flow codes with impressive results.

GMRES was added to T. Pulliam's ARC2D code (ref. 11), which solves both the Euler and Navier-Stokes equations around airfoils. In Figure 4 we show the typical performance gain achieved in solving the Euler equations for transonic flow around an airfoil using a 265 by 33 C-mesh. The addition of GMRES(20) improved the asymptotic convergence rate from .985 to .95, while the cost per iteration was increased by less than 15%. In fact, the run with GMRES shown in Figure 4 cost less than that made without GMRES. The memory required to add GMRES(20) to ARC2D was also more modest than one might first imagine. If G is the number of grid points, then ARC2D uses 45*G words of storage, of which 4*G are devoted to the solution. GMRES(20) requires (20+4)*4*G words of storage. Thus adding GMRES(20) increases the storage requirement from 45*G to 141*G, roughly a threefold increase. For G=265*33, this additional memory requirement was easily accommodated on our 2-million-word CRAY-1/S at no additional cost since we were charged by how long the CPU was tied up, not by how much memory was used.

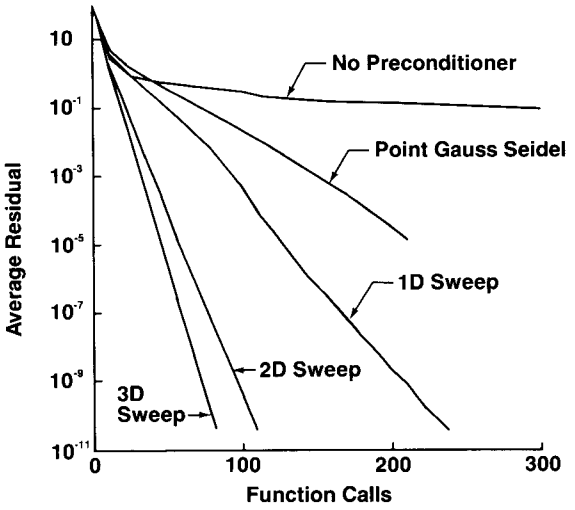

Figure 3. Convergence Histories in Terms of Function Calls for A U_{xx} + B U_{yy} + U_{zz} = 0 Using GMRES Preconditioned by Point Gauss Seidel and Line Relaxation in One, Two, and Three Directions

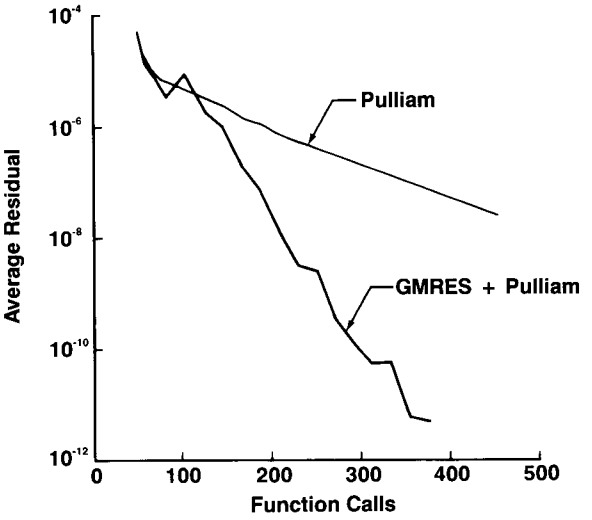

Figure 4. Convergence Histories for GMRES Applied to Pulliam's ARC2D Code. Euler Calculation for Transonic Flow Around RAE2822 Airfoil Using 265 x 33 C-Mesh. Minf=0.73, Alpha =2.79

It is interesting to note that the convergence history with GMRES in Figure 4 actually jumps up near the beginning. This is because our nonlinear version of GMRES approximates Newton's method, which can diverge unless one starts sufficiently close to the solution. We have found that a reliable cure to this problem is to introduce damping into equation (3.5). That is, given u^n we find u^{n+1} by using one GMRES cycle on

$$u^{n+1} - (1-\lambda)M(u^{n+1}) - \lambda M(u^n) = 0. \qquad (4.2)$$

The value of λ during early iterations is taken to be close to 1 and is gradually decreased to 0. The number of search directions k is taken to be small at first and gradually approaches its maximum value as $\lambda \to 0$. Employing this strategy allows GMRES to be started right away, even for flows involving strong shocks.

In Figure 5 we show the results of an experiment performed by Pulliam. He purposely set the time step parameter DT in his ARC2D code to 50. Even though Pulliam's implicit time marching scheme diverges when confronted with such a large DT parameter, it does not lose its ability to act as a good preconditioner for GMRES. In fact, GMRES+Pulliam(DT=50) converges just as fast as GMRES+Pulliam(DT=5). This example illustrates the ability of GMRES to improve the reliability of existing codes. In fact it is capable of stabilizing otherwise diverging algorithms.

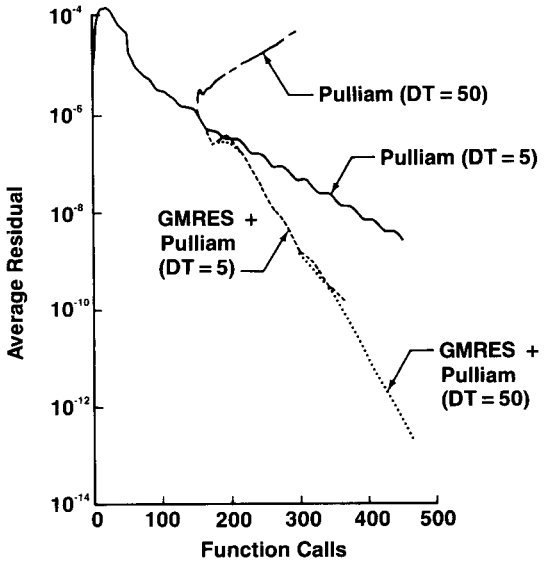

Figure 5. GMRES Applied to ARC2D Illustrating Ability To Stabilize Otherwise Diverging Algorithm

GMRES was also added to A. Jameson's 2D multigrid Euler code FLO53P (ref. 12). Convergence histories are shown in Figures 6 and 7. In the "easy" case shown in Figure 6, multigrid by itself performed quite well. GMRES did not improve the convergence rate until multigrid began to slow down after the residuals were reduced to 1.E-6. In the "hard" case shown in Figure 7 associated with a low Mach number and high angle of attack, multigrid by itself almost completely breaks down. However, as can be seen, it did not lose its ability to act as a good preconditioner for GMRES. An impressive aspect of this calculation is that GMRES enables the "hard" case shown in Figure 7 to converge just as fast as the "easy" case shown in Figure 6.

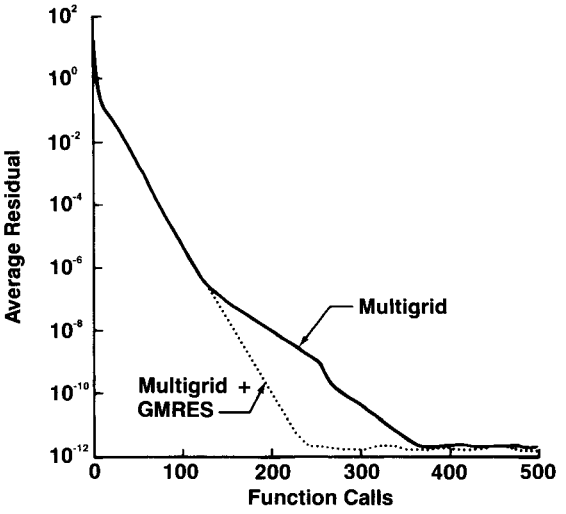

Figure 6. GMRES Applied to Jameson's Multigrid Euler Code FL053P Involving Transonic Flow Around NACA0012 Airfoil Using 193 x 33 C-Mesh. Minf=0.80, Alpha =1.25

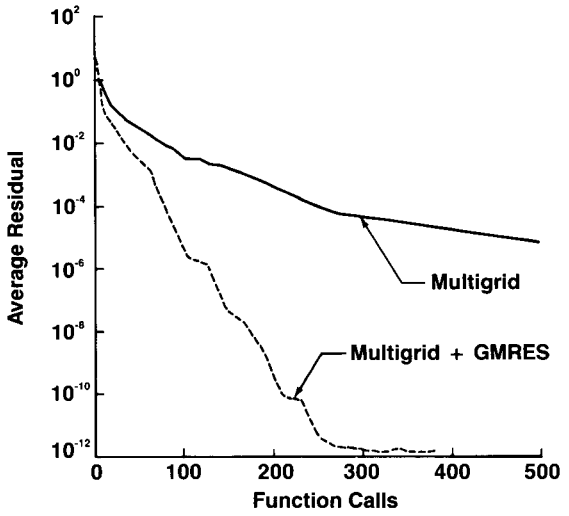

Figure 7. GMRES Applied to Jameson's Multigrid Euler Code FL053P Involving Subsonic Flow Around NACA0012 Airfoil Using 193 x 33 C-Mesh. Minf= 0.30, Alpha = 10.0

The recent acquisition of the CRAY X-MP by Boeing has allowed meaningful 3D calculations to be performed with the aid of GMRES. The convergence histories of FLO28 full-potential calculations for a 747 wing-body are shown in Figures 8a and 8b. These calculations were performed on a 145 by 17 by 33 mesh. The potential was initialized by first converging the solution on a coarser 73 by 9 by 17 mesh to machine error using GMRES. The four solution techniques being compared include the standard SLOR scheme that comes with the code, SLOR with a dominant eigenvalue extrapolation scheme (ref. 13), GMRES (30) preconditioned by SLOR, and GMRES (80) preconditioned by SLOR. The CRAY X-MP had no trouble accommodating the storage required by GMRES(80) for this problem. In fact, only 6% of the SSD was used. With the aid of GMRES, engineering accuracy was achieved after 200 iterations in the subsonic case and 600 iterations in the transonic case. The cost

per iteration was increased by 15% with GMRES(30) and 33% with GMRES(80). It is apparent from Figures 8a and 8b that even taking into account this additional overhead, GMRES offers a substantial improvement over our previous solution technology.

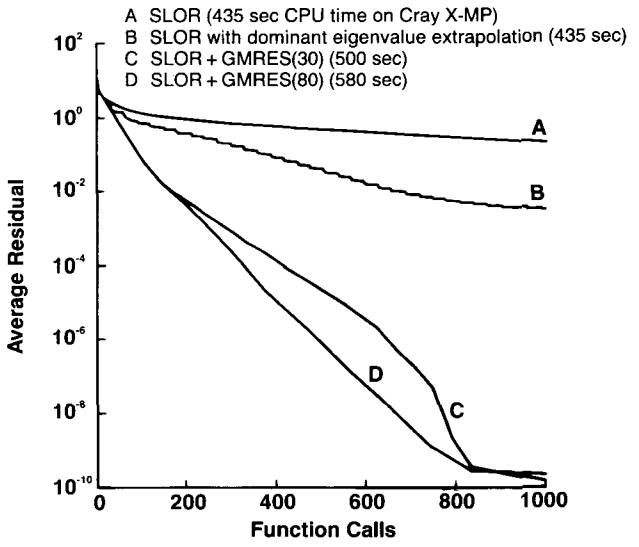

Figure 8a. 747 Wing-Body Full-Potential Calculation on 145 x 17 x 33 Mesh. Subsonic Case, Minf = 0.20, Alpha=2.69

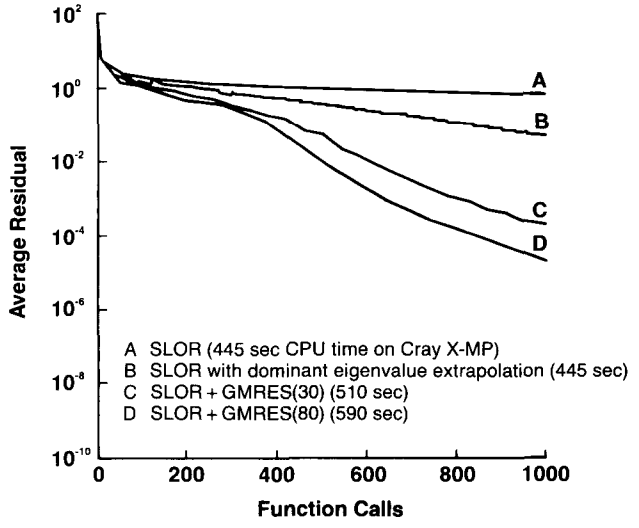

Figure 8b. 747 Wing-Body Full-Potential Calculation on 145 x 17 x 33 Mesh. Transonic Case, Minf =0.86, Alpha=2.69

These and other calculations indicate that GMRES can be easily retrofitted to existing flow solvers and can greatly improve their robustness and rate of convergence. Thus, we believe that GMRES constitutes a major advance in computational fluid mechanics.

5. Theoretical Aspects of GMRES

The heart of our nonlinear version of GMRES is of course the linear version. The latter has an interesting theory that we will discuss.

Let x_0 be an approximate solution of the linear system of equations

$$b + Ax = 0, \tag{5.1}$$

where A is an invertible matrix. Then a process mathematically equivalent to GMRES(k) constructs k orthogonal search directions $v_1,v_2,...v_k$ as follows. (The real GMRES algorithm makes the search directions orthonormal. The algebra in the argument we wish to present becomes a little simpler if we do not normalize the search directions.)

$$v_1 = b + Ax_0 = r_0 \tag{5.2}$$

For j = 1,2,..k–1 take

$$v_{j+1} = Av_j - \sum_{i=1}^{j} b_{ij}v_i, \tag{5.3}$$

where

$$b_{ij} = (Av_j,v_i)/(v_i,v_i). \tag{5.4}$$

The solution is now advanced from x_0 to x_k by choosing x_k to be the element of the affine space $x_0 + <v_1,v_2,...v_k>$, which minimizes the norm of the residual vector

$$r_k = b + Ax_k. \tag{5.5}$$

The only way this process can break down is if one of the search directions say, v_l, is zero. However, it is shown in Reference 7 that in this case the exact solution to (5.1) lies in the affine space $x_0 + <v_1,v_2,...v_{l-1}>$. Thus, the only time GMRES breaks down is when it has already computed the solution! We refer to this as "lucky" breakdown.

We now analyze the more common case where none of the search directions is zero. It follows from (5.2) and (5.5) that

$$||r_k|| = \underset{y}{\text{Min}}\ ||v_1 + Ay|| \text{ where } y \in <v_1,v_2,...v_k> \tag{5.6}$$

and, if this minimum occurs at $y=y_k$, then

$$x_k = x_0 + y_k.$$

Equation (5.3) and a simple inductive argument show that

$$<v_1,v_2...v_k> = <v_1,Av_1,..\ A^{k-1}v_1>. \tag{5.7}$$

From equations (5.6) and (5.7) we have

$$||r_k|| = \underset{y}{\text{Min}}||v_1 + Ay|| \text{ where } y \in <v_1,Av_1,...A^{k-1}v_1>. \tag{5.8}$$

That is, GMRES finds the best possible solution for y over the Krylov subspace $<v_1, Av_1,...A^{k-1}v_1>$ and is therefore referred to as a Krylov subspace method (ref. 7). Theoretically, once we had v_1 we could simply form Av_1, A^2v_1,...$A^{k-1}v_1$ and solve the least squares problem (5.8) directly. Unfortunately this process is numerically unstable. GMRES constructs an orthogonal basis $v_1,v_2...v_k$ of the Krylov subspace to ensure a numerically stable solution to the least squares problem (5.8).

Substituting the fact that $v_1=r_0$ (equation (5.2)) into equation (5.8) we find

$$||r_k|| = \underset{P(A)}{\text{Min}}\ ||P(A)r_0||, \tag{5.9}$$

where P(A) is of the form $I + a_1A + a_2A^2 + ... + a_kA^k$.

Thus, like all Krylov subspace methods, GMRES can also be thought of as an optimal polynomial acceleration scheme. It is important to note that the generation of this optimal polynomial occurs automatically without any user supplied information concerning A. The price for this service is the memory that is required.

Insight into the expected performance of optimal polynomial acceleration schemes such as GMRES can be obtained by considering the case where A is a diagonal matrix. It then follows from equation (5.9) that

$$||r_k|| \leq E(k)\,||r_0||, \tag{5.10}$$

$$\text{where } E(k) = \min_{p \in P(k)} \max_{\lambda \in \sigma(A)} |p(\lambda)|,$$

$P(k)$ is the set of polynomials p of degree less than or equal to k such that $p(0)=1$, and

$\sigma(A)$ is the spectrum of A = set of all eigenvalues of A.

Equation (5.10) shows why preconditioning (section 3) improves the performance of GMRES. Preconditioning decreases the size of the spectrum so that the optimal polynomial generated by GMRES can more easily annihilate the errors associated with each eigenvalue.

Determining the optimum polynomial for a given $\sigma(A)$ is in general a difficult problem in approximation theory. We will consider an interesting special case for which this problem has been solved.

Suppose that M is an affine transformation such that the spectrum of its associated linear operator $\bar{M}$ lies in an ellipse centered about the origin. The ellipse has semimajor axis "a" along the real axis and semiminor axis "b" along the imaginary axis ($b \leq a$). A picture of $\sigma(\bar{M})$ is shown in Figure 9a. The standard iteration scheme

$$u^{n+1} = M(u^n) \tag{5.11}$$

will converge at rate "a" provided $\sigma(\bar{M})$ lies within the unit circle. The convergence rate of GMRES applied to

$$u - M(u) = 0 \tag{5.12}$$

is determined by how well the optimum polynomial annihilates $\sigma(I - \bar{M})$ shown in Figure 9b. It is shown in Reference 14 that the optimum polynomial of degree k for this case is

$$p_k(z) = T_k((1-z)/c)/T_k(1/c), \tag{5.13}$$

$$\text{where } c = \text{sqrt}(a^2 - b^2) \tag{5.14}$$

and $T_k(z)$ is the Tchebychev polynomial of degree k, defined recursively by

$$T_0(z) = 1 \tag{5.15}$$

$$T_1(z) = z$$

$$T_{n+1}(z) = 2*z*T_n(z) - T_{n-1}(z).$$

For problem (5.12) the value of E(k) defined in (5.10) is given by

$$E(k) = T_k(a/c)/T_k(1/c). \tag{5.16}$$

Each cycle of GMRES requires k evaluations of M, so that the expected reduction in the residual per function evaluation is

$$r(k) = E(k)^{**}(1/k). \tag{5.17}$$

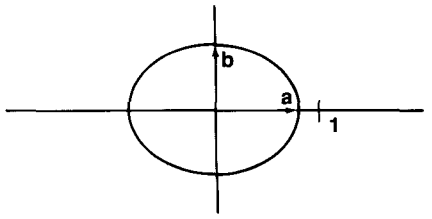

Figure 9a. Assumed Spectrum of $\bar{M}$

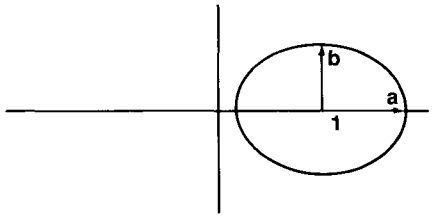

Figure 9b. Corresponding Spectrum of (I - $\bar{M}$)

r(k) is tabulated in Table 1 for various values of a,b, and k.

Table 1. Expected Rate of Convergence of GMRES (k) When σ(I-$\bar{M}$) Is Elliptical Region Shown in Figure 9b

Ellipse		Number of Search Directions (k)					
b/a	a	1	2	5	10	20	30
0.0	.900	.900	.825	.719	.672	.649	.641
0.0	.950	.950	.907	.825	.776	.749	.741
0.0	.980	.980	.961	.916	.874	.846	.836
0.0	.990	.990	.980	.954	.925	.898	.888
0.0	.995	.995	.990	.976	.957	.936	.926
.1	.900	.900	.826	.728	.694	.686	.685
.1	.950	.950	.908	.833	.798	.789	.788
.1	.980	.980	.962	.921	.893	.883	.882
.1	.990	.990	.980	.957	.938	.930	.929
.1	.995	.995	.990	.978	.966	.960	.959
.5	.900	.900	.853	.831	.830	.830	.830
.5	.950	.950	.923	.909	.909	.909	.909
.5	.980	.980	.969	.962	.961	.961	.961
.5	.990	.990	.984	.981	.980	.980	.980
.5	.995	.995	.992	.990	.990	.990	.990
1.0	.900	.900	.900	.900	.900	.900	.900
1.0	.950	.950	.950	.950	.950	.950	.950
1.0	.980	.980	.980	.980	.980	.980	.980
1.0	.990	.990	.990	.990	.990	.990	.990
1.0	.995	.995	.995	.995	.995	.995	.995

As can be seen, if b/a is small, then GMRES offers significant acceleration over the standard iteration scheme (5.11). This analysis explains why GMRES performs so well on the elliptic problems shown in Figures 1,2,3, and 8a. However, as $b/a \rightarrow 1$ (corresponding to a full circle of eigenvalues), Table 1 suggests that GMRES will perform no better than the standard iteration scheme. Fortunately this assessment is often overly pessimistic. Frequently there are gaps in the eigenvalue distribution that can be exploited by

GMRES. This ability is particularly useful when there are a few "wayward" eigenvalues that are larger than those contained within the main group. GMRES can devote some of the zeroes of the optimum polynomial to annihilate the errors associated with the wayward eigenvalues while using the remaining zeroes to annihilate the errors caused by the main group. The ability to "shepherd" wayward eigenvalues explains why GMRES improved the multigrid calculations shown in Figures 6 and 7. Furthermore, as exhibited in Figure 5, GMRES can even shepherd eigenvalues that lie outside the unit circle and that would otherwise cause the solution to diverge.

Thus we see that the secret to the success of GMRES is its ability to automatically generate optimum polynomials. Not only does this frequently accelerate convergence, but it also builds in flexibility to adapt to changing eigenvalue distributions, resulting in improved reliability.

6. Comparison Between GMRES and Similar Methods

As shown above, starting with x_0, one cycle of GMRES finds the best possible solution to the following linear equation for x:

$$b + Ax = 0 \tag{6.1}$$

over a k dimensional affine space $x_0 + \langle v_1, v_2, \ldots v_k \rangle$. In GMRES the search directions are chosen to be orthonormal. An alternative method, ORTHODIR , chooses the search directions to be A^tA orthogonal, that is,

$$(Av_i, Av_j) = 0 \quad \text{for} \quad i \neq j. \tag{6.2}$$

ORTHODIR computes the k search directions as:

$$v_1 = b + Ax_0 = r_0$$

for j=1,2,...k-1 take

$$u_j = Av_j$$

$$v_{j+1} = u_j - \sum_{i=1}^{j} b_{ij} v_i,$$

where

$$b_{ij} = (Au_j, Av_i)/(Av_i, Av_i).$$

The coefficients b_{ij} are computed to enforce equation (6.2). As in GMRES, ORTHODIR advances the solution from x_0 to x_k by choosing x_k to be the member of the affine space $x_0 + \langle v_1, v_2, \ldots v_k \rangle$, which minimizes the norm of

$$r_k = b + Ax_k,$$

which is equivalent to finding $a_1, a_2, \ldots a_k$, which minimize

$$||r_0 + \sum_{j=1}^{k} a_j Av_j||^2. \tag{6.3}$$

Now because of equation (6.2) this least squares problem can be explicitly solved:

$$a_j = -(r_0, Av_j)/(Av_j, Av_j).$$

It can be easily shown (ref. 7) that the search directions generated by ORTHODIR span the same k dimensional Krylov subspace as those generated by GMRES. Therefore, GMRES and ORTHODIR are mathematically equivalent. Since ORTHODIR solves the least squares problem more conveniently, it would appear to be preferable to GMRES. However, ORTHODIR requires that both the search directions v_j and the vectors Av_j be stored. Since GMRES only stores the search directions v_j and the last vector Av_k (section 2), GMRES uses only about half the storage required by ORTHODIR. Also, GMRES requires fewer arithmetic operations than ORTHODIR. Moreover, numerical tests (ref. 7) have shown GMRES to be more numerically stable. Of all the Krylov subspace methods for solving a nonsymmetric linear system of equations (ORTHODIR, ORTHOMIN, ORTHORES, and GCR are compared in ref. 7), GMRES appears to be the best in terms of storage, operation count, and numerical stability.

For problems involving symmetric positive definite matrices, algorithms more efficient than GMRES exist. Consider the linear system

$$Sx = b, \tag{6.4}$$

where S is a symmetric positive definite matrix. The conjugate residual (CR) method employs the equations (ref. 2):

Choose x_0

Set $r_0 = b - Sx_0$

$p_0 = r_0$.

Now recursively use

$a_i = (r_i, Sr_i)/(Sp_i, Sp_i)$

$x_{i+1} = x_i + a_i p_i$

$r_{i+1} = r_i - a_i Sp_i$

$b_i = (r_{i+1}, Sr_{i+1})/(r_i, Sr_i)$

$p_{i+1} = r_{i+1} + b_i p_i$

$Sp_{i+1} = Sr_{i+1} + b_i Sp_i$.

It then follows that (ref. 2)

$$(Sp_i, Sp_j) = 0 \text{ for } i \neq j$$

and that x_k minimizes

$$||Sx - b||^2$$

over the k dimensional affine space $x_0 + \langle p_0, p_1, \ldots p_{k-1} \rangle$.

Thus, CR operates very much in the same spirit as GMRES (in fact for symmetric positive definite linear systems they are mathematically equivalent), but explicit orthogonalizations such as equation (5.3) and the need for solving the least squares problem equation (5.6) are avoided. Moreover, storage is required for only the 5 vectors x, r, Sr, p, and Sp. This compares very favorably with the k+4 stored vectors required by our implementation of GMRES (typically k=20).

Another famous algorithm for solving equation (6.4) is the conjugate gradient method. It uses the relations (ref. 2):

Choose x_0

Set $r_0 = b - Sx_0$

$p_0 = r_0$.

Now recursively use

$$a_i = (r_i,r_i)/(p_i,Sp_i)$$

$$x_{i+1} = x_i + a_i p_i$$

$$r_{i+1} = r_i - a_i Sp_i$$

$$b_i = (r_{i+1},r_{i+1})/(r_i,r_i)$$

$$p_{i+1} = r_{i+1} + b_i p_i.$$

It then follows that (ref. 2)

$$(p_i,Sp_j) = 0 \text{ for } i \neq j$$

so that the search directions are (by definition) S-orthogonal (conjugate) to each other. Also, if y is the exact solution to equation (6.4), then x_k minimizes (ref. 2):

$$||y - x||$$

over the k dimensional affine space $x_0 + <p_0,p_1,...p_{k-1}>$.

Again CG enjoys enormous advantages over GMRES in terms of storage and operation count when applied to symmetric positive definite linear systems. Unfortunately, many of the advantages of CR and CG disappear when applied to nonsymmetric problems. Let G be a general (invertible!) nonsymmetric matrix. Then

$$Gx = b \tag{6.5}$$

can be solved with CG or CR if one considers

$$G^tGx = G^tb. \tag{6.6}$$

This involves the added expense and inconvenience of computing the adjoint operator G^t. The adjoints of any preconditioners must also be computed. This could be nearly impossible if we are using something like multigrid as a preconditioner. The elegant formulation shown in equation (3.5), which allows GMRES to be immediately retrofitted to existing codes, is lost. Also, the eigenvalues associated with equation (6.6) are more spread out than those of equation (6.5). This means that more iterations are required to solve equation (6.6) than to solve equation (6.5).

In view of the difficulties associated with applying CG or CR to equation (6.6) one must ask how much it really costs to apply GMRES to equation (6.5) directly. In a typical application, the cost of GMRES turns out not to be too burdensome. The operations required by GMRES are fully vectorizable over vectors of length equal to the number of unknowns in the problem and are therefore capable of efficient implementation. As we have seen in our applications of GMRES to CFD codes, the overhead due to the addition of GMRES is quite acceptable. Moreover, the extra storage required by GMRES can be accommodated by the new supercomputers. For these reasons we believe that in most cases applying GMRES to equation (6.5) is preferable to using CG or CR on equation (6.6)

7. Conclusion

We have reformulated a conjugate-gradient like method GMRES recently developed by Saad and Schultz (ref. 7) so that it applies to the nonlinear equations characteristic of CFD. We have demonstrated that GMRES can be easily retrofitted into existing codes and take full advantage of the solution procedures already built into the codes. Through theory and example we have shown that the introduction of GMRES frequently accelerates convergence and improves reliability. The price paid for the services of GMRES is the requirement for more memory. However, this resource is becoming increasingly available through the development of new supercomputers. Thus, algorithm and computer hardware technology have advanced together to provide a powerful new tool. After years of waiting in the background, conjugate-gradient like methods such as GMRES are finally ready for widespread use in CFD codes.

Acknowledgments

This work was partially supported by NASA contract NAS2-11851. We wish to thank Horst Simon of Boeing Computer Services for providing us with a Fortran implementaiton of the linear version of GMRES.

1. *Finite Element Solution of Boundary Value Problems, Theory and Computation,* O. Axelson and V. A. Barker, Academic Press Inc. 1984.

2. "Conjugate Gradient Methods for Partial Differential Equations," R. Chandra, Yale University Research Report 129, January 1978.

3. "Application of Conjugate Gradient Methods to Transonic Finite Difference and Finite Element Calculations," Y. S. Wong and M. Hafez, AIAA paper 81-1032.

4. "Calculations of Transonic Potential Flows by a Parameter Free Procedure," Y. S. Wong, AIAA paper 83-1886.

5. "Conjugate Gradients Methods for Solution of Finite Element and Finite Difference Flow Problems," T. C. Prince, AIAA paper 83-1923.

6. "Iterative Methods for Large, Sparse, Nonsymmetric Systems of Linear Equations," H. C. Elman, Yale University Department of Computer Science Research Report 229, April 1982.

7. "GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems," Y. Saad and M. H. Schultz, Research Report YALEU/DCS/RR-254, August 24, 1983.

8. *Matrix Computations,* G. H. Golub and C. F. Van Loan, The Johns Hopkins University Press, 1983.

9. *Spectral Approximation of Linear Operators,* F. Chatelin, Academic Press, 1983.

10. *LINPACK Users' Guide,* J.J. Dongarra, C. B. Moler, J. R. Bunch, and G. W. Stewart, SIAM, 1979.

11. "Euler and Thin Layer Navier Stokes Codes: ARC2D,ARC3D," T. H. Pulliam, Computational Fluid Dynamics Workshop at the University of Tennessee Space Institute, March 12-16, 1984.

12. "Transonic Flow Calculations," A. Jameson, Department of Mechanical and Aerospace Engineering, MAE Report 1651, Princeton University, July 1983.

13. "Transonic Flow Simulations for Complex Configurations With Surface-Fitted Grids," N. J. Yu, AIAA paper 81-1258.

14. "The Tchebychev Iteration for Nonsymmetric Linear Systems," T. A. Manteuffel, Numer. Math. *28*, 307-327, 1977.