Implementation of Generalized Minimum Residual Krylov Subspace Method for Chemically Reacting Flows

Matthew MacLean[1]
Calspan-University at Buffalo Research Center, Buffalo, NY, 14225

Todd White[2]
ERC, Inc., Moffett Field, CA 94035

Two independent implementations of the Generalized Minimum Residual (GMRES) technique have been tested in the chemically reacting DPLR CFD code. The first implementation utilizes the existing Jacobian matrices by writing a new GMRES solver that is integrated directly into the DPLR CFD code time integration subroutine. The second implementation makes calls to the Portable, Extensible Toolkit for Scientific Computation (PETSc), an external linear and non-linear solver package, to perform the GMRES calculation. Several important aspects of developing the built-in GMRES solver are explored including basis vector orthogonalization technique, linear system pre-conditioning, parallelization, and convergence criteria. The resulting performance is compared with the PETSc implementation as well as line- and point- relaxation schemes for hypersonic cone, $25^O/55^O$ double cone, and spherical capsule test cases. Overall, line-relaxation is typically the most efficient solution technique when elapsed time is measured, but GMRES can, in some cases, provide more accurate solution of the matrices by measuring residual decrease per iteration. In the double cone test case at high enthalpy, the GMRES method is shown to be numerically more stable at larger timestep size. Comparisons between PETSc and the built-in GMRES solvers show that choice of convergence criteria drive the relative performance of the solvers.

Nomenclature

A	=	sparse Jacobian matrix of target system
b	=	right-hand side vector of target system
c	=	Givens rotation cosine
d	=	diagonal elements of ILU preconditioner, Eqn (29)
$e_{i,j}$	=	unit vector of length j, with one non-zero element, i
E	=	lower off-diagonal elements of sparse A
F	=	upper off-diagonal elements of sparse A
G	=	Givens rotation matrix
H	=	Hessenberg matrix
k	=	number of line- relaxation steps
L	=	lower diagonal matrix for Householder transformation, Eqn (9)
m	=	number of GMRES restarts
M	=	dimension/rank of Krylov subspace (size of each GMRES restart)
N	=	dimension/rank of linear system
P	=	preconditioner matrix
p	=	order of ILU fill
Q	=	matrix of orthogonal basis vectors
r	=	residual
R	=	transformed Hessenberg matrix
s	=	Givens rotation sine

[1] Senior Research Scientist, AIAA Senior Member, maclean@cubrc.org.
[2] Research Scientist, AIAA Member

u = Householder vector
U = matrix of Householder column vectors
w = raw (unorthogonalized) Krylov subspace vector
x = target linear system solution vector
y = target linear system solution vector in Krylov space

Greek

β = initial residual magnitude
l = Courant–Friedrichs–Lewy (CFL) number

subscript

M = condition after calculation of Krylov subspace K_M
0 = initial condition

I. Introduction

EFFICIENT solution of the large-scale, sparse matrix systems generated from multi-dimensional, chemically-reacting Computational Fluid Dynamics (CFD) codes is a critical part of the iterative convergence of the discretization of the Navier-Stokes equations. For high-speed flows, the time-scales for chemical and thermal relaxation processes as well as the potential for very large gradients of the flow variables (e.g. normal to an adjacent solid surface) means that the terms in the resulting linear mathematical system can be poorly conditioned. An inherent complexity in solving these systems is the need to consider parallelization in the solution scheme since both memory and performance concerns require that practical CFD problems will be solved on large-scale, parallel cluster systems.

A number of effective strategies for applying point-relaxation and line-relaxation methods to these matrices have been developed such as the full-matrix data-parallel method[1] (FMDP) or the data-parallel line-relaxation method[2] (DPLR). The DPLR method recognizes that a strong coupling occurs with the gradients normal to a solid surface boundary relative to the gradients in other directions in hypersonic flows where the viscous boundary layer is very thin[3]. If the unknowns are simultaneously solved along lines of hexahedral or prismatic cells while relaxing the other terms of the system, the sparse matrix may be replaced by a small number (typically 4) of relaxation steps where a significantly less expensive, block tri-diagonal solution is required. This approach is robust for a wide range of high-speed flows with little or no reduction in convergence performance. However, there may be certain circumstances where the assumption that the off-diagonal terms may be relaxed can result in significantly poorer convergence properties. Situations include conditions where the surface-normal gradients do not dominate the flow or where there is no preferential gradient direction (e.g. wakes or separated flows) or conditions where the simplified, diagonal matrix is not well conditioned (e.g. low speed flows, large timesteps). For these cases, it may be better to retain all terms of the sparse matrix even though the cost of the solution is generally higher.

In this work, a type of method called Generalized Minimum Residual[4] (GMRES) is considered, which is part of a general class of linear solution algorithms collectively referred to as Krylov subspace methods. The Krylov subspace methods seek an approximate solution to the linear system by relying on matrix/vector products to transform the matrix into a reduced dimension that reflects the most dominant eigenvectors of the original system. Methods like GMRES are attractive for CFD applications since they generally do not require storage or manipulation of the zero entries of the matrix. Several comprehensive software library packages like PETSc[5] and Trilinos[6] are freely available from the Internet and they have been utilized previously for CFD applications[7-10]. Such packages have several advantages, namely (1) they are well-debugged and have significant development time already invested in them, (2) they have a large number of options and methods available that typically can be selected or activated with a few function calls, (3) they can be utilized without a detailed understanding of how the internal algorithms work, and (4) they have been tested with a wide variety of linear systems that presumably encompass the limits of what might be observed from the CFD simulation. However, there are a few drawbacks to using these packages, such as (1) they require users of the CFD code to download, compile, and link to additional dependencies in order to get the CFD code to run, (2) if the solution fails for some reason, it can be difficult to debug without familiarity with the external package source code, and (3) there can be an inherent storage redundancy if the existing CFD code and the external package do not have compatible formats which can waste significant memory or force major re-writing of the CFD code.

Since there are some advantages and disadvantages to using an external linear solver package, this work considers both the option to use the PETSc package (version 3.1p8) as well as the development of a new GMRES solver that is an integral part of the NASA Ames Research Center DPLR CFD code. As the proper name of the code is the same as the line-relaxation algorithm that it implements, this paper uses the convention “DPLR” to indicate

the line-relaxation algorithm and "DPLR CFD code" to indicate the CFD code package to avoid confusion. The two GMRES options are toggled via a pre-processor definition so that either the PETSc formulation or the built-in formulation may be selected at compile time. The development of the built-in GMRES solver is considered in the subsequent sections, followed by several examples that compare the two formulations with each other and with the existing line- and point- relaxation schemes. These two new solver options are released as part of V4.03.0 of the DPLR CFD code ("Big Bend").

The DPLR CFD code is a multi-block, structured, finite-volume code that solves the reacting Navier-Stokes equations including finite rate chemistry and finite rate vibrational non-equilibrium effects. This code is named for the data-parallel line relaxation method[2] and implements a modified (low dissipation) Steger-Warming flux splitting approach[11] for the convection terms and central differencing for the diffusion terms. Finite rate vibrational relaxation is modeled via a simple harmonic oscillator vibrational degree of freedom[12] using the Landau-Teller model[13]. Vibrational energy relaxation rates are computed by default from the semi-empirical expression due to Millikan and White[14], but rates from the work of Camac[15] and Park, et al[16] are substituted for specific collisions where experimental data exists. Vibration-dissociation coupling is currently modeled using the *T-Tv* approach of Park[17] or with some preliminary implementation of CVDV coupling[18]. Transport properties are appropriately modeled in the DPLR CFD code for high enthalpy flow[19,20] using the binary collision-integral based mixing rules from Gupta, et al[21]. Diffusion fluxes can be modeled using either Fick's law of diffusion or the self-consistent effective binary diffusion (SCEBD) method[22].

II. Derivation of GMRES Algorithm Applied to the DPLR CFD Code

A. Introduction

The premise of the GMRES method and other Krylov techniques is the projection of a matrix of size NxN defining a linear system of N unknowns into a smaller subspace of size M (where M<<N), defined by a series of vectors that capture the dominant eigenvectors of the original matrix. Saad[23] provides a comprehensive description of the theory of Krylov techniques as well as implementation details. The method starts from an initial guess for the solution vector, x_0, which defines an initial residual in Eqn(1) with a magnitude of $\beta = \|r_0\|$.

$$r_0 = b - Ax_0 \tag{1}$$

The Krylov subspace is defined by a series of vectors that form the columns of Eqn(2), which span the Krylov subspace K_M.

$$\left\{ r_0 \middle| Ar_0 \middle| A^2 r_0 \middle| \ldots \middle| A^{M-1} r_0 \right\} \tag{2}$$

In general, these vectors are not very orthogonal to each other so they do not form a good basis on their own; therefore, an orthogonalization procedure is necessary. Typically, Gram-Schmidt orthogonalization[24] is used for this, although Householder transformations have been investigated as an alternative[25]. The transformation of any matrix, A, into the M^{TH} Krylov subspace, K_M, using this orthogonal basis vector set, Q_M, will be an upper Hessenberg matrix, H_M, given in Eqn(3) where the sizes of the matrices are indicated below each element for clarity.

$$\underset{(MxN)}{Q_M^T} \underset{(NxN)}{A} \underset{(NxM)}{Q_M} = \underset{(MxM)}{H_M} \tag{3}$$

The upper Hessenberg is guaranteed to have a form that is almost an upper triangular matrix with one additional row of non-zero values adjacent to the main diagonal.

B. Subspace Vector Orthogonalization

Orthogonalization of the set of vectors spanning the Krylov subspace described in Eqn(2) can be accomplished by several methods. Gram-Schmidt orthogonalization[24] is one popular choice. Classical Gram-Schmidt is obtained via the Arnoldi method from Eqn(4), where w_m is obtained by multiplying the previously computed basis vector by the matrix A.

$$q_m = w_m - \sum_{i=1}^{m-1} \left(w_m \bullet \hat{q}_i \right) \hat{q}_i \tag{4}$$

Modified Gram-Schmidt updates the new vector immediately as it is orthogonalized against each previous vector. Numerically, this process is less susceptible to numerical loss of precision than classical Gram-Schmidt. Modified Gram-Schmidt is obtained via Eqn(5).

$$\begin{aligned} q_m^{(0)} &= w_m \\ q_m^{(i)} &= q_m^{(i-1)} - \left(q_m^{(i-1)} \bullet \hat{q}_i \right) \hat{q}_i \qquad for \quad i = 1, m-1 \end{aligned} \tag{5}$$

Mathematically, both types of Gram-Schmidt give the same result. Modified Gram-Schmidt has significantly better numerical behavior but it is more expensive to implement in parallel because a summation operation must be performed for each dot product immediately rather than at the end. In either case, each vector in the basis, Q_M, is normalized to unit magnitude as it is used in GMRES.

An alternate method of orthogonalization is the use of Householder transformations as suggested by Walker[25]. The Householder transformation technique orthogonalizes the basis through a series of matrix transformations of the form given in Eqn(6), where $\hat{u}_m$ decreases in effective length such that it contains non-zero elements only greater than or equal to m and the first m-1 rows and columns of the Householder matrix, Q_m, will be equal to the identity matrix.

$$Q_m = I - 2\hat{u}_m \hat{u}_m^T \tag{6}$$

Each vector to be orthogonalized is defined by the products defined in Eqn(7).

$$w_m = \left(\prod_{i=1}^{m-1} Q_i \right) \left(\prod_{i=1}^{m-1} A \right) r_0 \tag{7}$$

As defined by Eqn(7), computing vectors becomes increasing expensive for large systems; however, Walker (algorithm 3.1)[25] notes that that this technique can be more practically implemented by solving the system given in Eqn(8).

$$w_m = \left\{ I - 2U_m L_m^{-1} U_m^T \right\} \{A\} \left\{ I - 2U_m \left(L_m^T \right)^{-1} U_m^T \right\} \hat{e}_m \tag{8}$$

Although this system looks complex, it can be solved using only a single forward-substitution (the inversion of L_m) and a single backward-substitution (the inversion of L_m^T) with appropriate intermediate multiplications. The matrix U_m is simply the matrix of Householder vectors that define the series of transformations from Eqn(6) while the matrix L_m is a lower diagonal that is defined by Eqn(9):

$$L_m = \begin{bmatrix} 1 & & & \\ 2u_2^T u_1 & 1 & & \\ \vdots & & \ddots & \\ 2u_m^T u_1 & \cdots & 2u_m^T u_{m-1} & 1 \end{bmatrix} \tag{9}$$

The Householder transformation vector, u_m, is found by evaluating Eqn(10), where the accent on the vector indicates that the first *m-1* components of w_m are zeroed out before computing the Householder transformation.

$$u_m = \hat{w}_m - \left\|\hat{w}_m\right\| \hat{e}_{m,N} \tag{10}$$

Both the Gram-Schmidt (classical and modified) and the Householder orthogonalization methods require a similar amount of storage. Gram-Schmidt requires the storage of (M+1) vectors of N-length while Householder requires (M+1) vectors that decrease in length from N to 1 (an upper-diagonal) as well as the lower diagonal matrix L_m. The Householder U_m and L_m matrices can be jointly stored, so both methods require the storage of (M+1)*N elements of double precision. This is in addition to temporary storage arrays, minor variables like Hessenberg elements, etc. which do not significantly contribute to the memory footprint. For use in intrinsically parallel environments like CFD, however, the Householder method requires significantly more message passing between processes. In parallel environments, Householder transformations are typically implemented using a block approach, where the Householder transformations are performed locally on each block of unknowns assigned to that process and then combined in a tree pattern[26,27]. Gram-Schmidt is significantly easier to implement and significantly faster for CFD applications if the loss of orthogonality can be tolerated. The stability of the GMRES method has been proven with Gram-Schmidt orthogonalization[28-29], but the practical behaviors for systems of equations generated from the DPLR CFD code are considered in the next section.

Improvement to the quality of the orthogonalization has also been observed for some generic vector systems when the original monomial basis (the set of vectors in Eqn(2)) is modified by introducing scaling[30] or Leja re-ordering to form, for example, a Newton basis[26,31-32]. Such a basis has been shown to reduce the possibility of numerical loss of orthogonalization for ill-conditioned problems, but forming such a basis comes at significant extra expense. The use of any basis other than the monomial basis has not been explored here since it would likely be prohibitively expensive for CFD code integration.

C. GMRES Solution Technique

After M-steps of orthogonalization, GMRES provides M+1 orthogonal vectors (including the original residual vector) with the following relationship in Eqn(11), where the tilde on the Hessenberg matrix indicates that it contains one extra row so it is not a square matrix:

$$\underset{(NxN)}{A}\ \underset{(NxM)}{Q_M} = \underset{(NxM+1)}{Q_{M+1}}\ \underset{(M+1xM)}{\tilde{H}_M} \tag{11}$$

GMRES finds the new solution x_M that minimizes the residual from its original value β after M steps of orthogonalization, defined in Eqn(12):

$$\left\|r_M\right\| = \left\|r_0 - A\left(x_M - x_0\right)\right\| \tag{12}$$

Eqn(5) may be transformed to a new variable, y_M, defined by Eqn(13):

$$Q_M y_M = \left(x_M - x_0\right) \tag{13}$$

This allows the substitution of the modified Hessenberg matrix in Eqn(14):

$$\left\|r_M\right\| = \left\|r_0 - AQ_M y_M\right\| = \left\|r_0 - Q_{M+1}\tilde{H}_M y_M\right\| \tag{14}$$

This is equivalent to minimizing a modified residual obtained by pre-multiplying by the basis vector set as in Eqn(15):

$$\left\|\tilde{r}_M\right\| = \left\|Q_{M+1}^T r_M\right\| = \left\|Q_{M+1}^T r_0 - \tilde{H}_M y_M\right\| \tag{15}$$

Since the first basis vector of the transformed M subspace was chosen to be the initial residual, the first term in Eqn(15) will always be the magnitude of the initial residual vector, β, times the unit vector $\langle 1, 0, \ldots, 0\rangle^T$, designated

$\hat{e}_{1,M+1}$. The GMRES algorithm is thus the least-squares minimization of the residual in the K_M subspace from M+1 equations, given in Eqn(16):

$$\left\|\tilde{r}_M\right\| = \left\|\beta\hat{e}_{1,M+1} - \tilde{H}_M y_M\right\| \tag{16}$$

Saad and Schultz note that the system of M+1 equations may be most readily solved by modifying the upper Hessenberg matrix with a series of Givens rotations of the form of Eqn(17) where the rotation, G_i, occupies the i and i+1 rows and columns of the matrix, which can be performed at each step of the process.

$$G_i = \begin{bmatrix} \ddots & & & & & & \\ & 1 & & & & & \\ & & \cos\theta_i & -\sin\theta_i & & & \Leftarrow row(i) \\ & & \sin\theta_i & \cos\theta_i & & & \Leftarrow row(i+1) \\ & & & & 1 & & \\ & & & & & \ddots & \end{bmatrix} \tag{17}$$

The least-squares system is solved by a series of these Givens rotations to transform the Hessenberg matrix as in Eqn(18) so that the M+1 row of the transformed R_M becomes trivially zero and may be dropped.

$$R_M = \left(\prod_{i=1}^{M} G_i\right)\tilde{H}_M \tag{18}$$

At each step, i, the Givens rotation can be found by applying each preceding rotation to the new column of Hessenberg elements. The new rotations are then defined by Eqn(19) and Eqn(20).

$$c_i = \cos\theta_i = \frac{r_{i,i}}{\sqrt{r_{i,i}^2 + h_{i+1,i}^2}} \tag{19}$$

$$s_i = \sin\theta_i = \frac{-h_{i+1,i}}{\sqrt{r_{i,i}^2 + h_{i+1,i}^2}} \tag{20}$$

Here, it is implied that any element, $r_{i,i}$, has already been transformed by the (i-1) successive transformations of the Hessenberg element(s) as in Eqn(21), where the initial elements $r_{n,i}$ are equal to those of the Hessenberg, $h_{n,i}$ (only the elements of R_M are ever stored in memory).

$$\left.\begin{aligned} r_{n,i}^{(n)} &= c_n r_{n,i}^{(n-1)} - s_n r_{n+1,i}^{(n-1)} \\ r_{n+1,i}^{(n)} &= s_n r_{n,i}^{(n-1)} + c_n r_{n+1,i}^{(n-1)} \end{aligned}\right\} \quad for \quad n = 1, i \tag{21}$$

In the same way, the elements of the first term of the least-squares problem corresponding to the initial residual β are transformed according to Eqn(22):

$$b_m = c_m\left(\prod_{i=1}^{m-1} s_i\right)\beta \tag{22}$$

The transformed system of the least-squares problem is a system of M equations and M unknowns with an upper diagonal matrix R_M:

$$R_M y_M = b_M \tag{23}$$

With the transformed Hessenberg matrix R_M, the solution to the least-squares problem for the vector y_M can be found by back-substitution.

$$y_m = \frac{1}{r_{m,m}}\left[c_m \beta \left(\prod_{i=1}^{m-1} s_i \right) - \sum_{i=m+1}^{M} r_{m,i} y_i \right] \qquad for \quad m = M,1 \tag{24}$$

The change to the solution vector x_M can be found through use of Eqn(13). One advantage of GMRES that was recognized by Saad and Schultz is that the residual of the solution can be monitored as each rotation is generated via Eqn(20) so that the method can be stopped when it reaches some threshold of acceptability.

$$\left\| r_M \right\| = \left(\prod_{i=1}^{M} s_i \right) \beta \tag{25}$$

As Eqn(25) implies, the calculation of each vector results in a Givens rotation that will multiply the existing residual by something between 0.0 and 1.0, so the residual decreases by some percentage at each iteration. The rate of convergence will be a function of the conditioning of the system. A property of GMRES is that it will converge in one iteration if M is chosen equal to N, but the advantage of GMRES is exploited by choosing M to be much less than N. GMRES is most commonly implemented with restart, where, after M steps, the method is restarted using the last computed solution x_M as an improved initial guess for the next restart cycle. This keeps memory usage to a minimum. Unlike the condition of running GMRES through the full N steps, there is no mathematical guarantee of convergence with restarted GMRES, but the technique behaves well for most problems.

D. Linear System Preconditioning

The convergence rate of the GMRES method can be significantly enhanced by modifying the system with a preconditioner matrix as in Eqn(26). The matrix P is chosen to be an approximation of the matrix A for which the inverse can be computed cheaply.

$$P^{-1}\left(Ax\right) = P^{-1}\left(b\right) \tag{26}$$

Saad[23] discusses a number of preconditioning techniques. Incomplete Lower/Upper decomposition (ILU) has been considered here. ILU approximates the factorization of the sparse matrix A into lower (L) and upper (U) matrices for inversion. In general, an exact LU decomposition of the sparse matrix A will result in L and U matrices which are not sparse or at least do not have the non-zero pattern of A. In ILU decomposition, L and U are approximated by retaining only a non-zero pattern related to A and neglecting the remaining terms. A level of fill, p, is used to annotate the amount of terms retained in addition to the non-zero pattern of A. Thus, ILU(0) factors A into L and U that have the same non-zero pattern as A. Additional levels of fill ILU(p) retain more terms and are more accurate factorizations at higher preconditioner costs (both in terms of operations and memory).

The CFD system of equations for a particular unknown cell (i,j,k) are of the form given in Eqn(27). The subscript on the block elements of A indicate the row in the matrix that the term appears in while the superscript is used to indicate the column that the term appears in, where indices not listed in the superscript are assumed to be (i), (j), or (k).

$$\begin{aligned} &a_{i,j,k}^{i-1}\Delta^n x_{i-1,j,k} + a_{i,j,k}^{j-1}\Delta^n x_{i,j-1,k} + a_{i,j,k}^{k-1}\Delta^n x_{i,j,k-1} + a_{i,j,k}\Delta^n x_{i,j,k} \\ &+a_{i,j,k}^{i+1}\Delta^n x_{i+1,j,k} + a_{i,j,k}^{j+1}\Delta^n x_{i,j+1,k} + a_{i,j,k}^{k+1}\Delta^n x_{i,j,k+1} = b_{i,j,k} \end{aligned} \tag{27}$$

For ILU(0), Saad explains that the 5-point and 7-point stencils used by structured CFD codes like the DPLR CFD code can be exploited to produce an efficient factorization of the form given in Eqn(28).

$$P = \left(D + E\right)\left(D^{-1}\right)\left(D + F\right) \tag{28}$$

The matrices E and F are the off-diagonal elements of A, where E is the set of elements below the diagonal [e.g. the $(i-1)$, $(j-1)$, and $(k-1)$ Jacobian terms] and F is the set of elements above the diagonal [e.g. the $(i+1)$, $(j+1)$, and $(k+1)$ Jacobian terms]. The approximate preconditioner, P, is inverted by noting that, in Eqn(28), D+F is upper-diagonal and D+E is lower diagonal and may be solved via the standard forward/backward substitution technique of LU decomposition. The elements of the diagonal matrix D are computed as given in Eqn(29).

$$d_{i,j,k} = a_{i,j,k} - a^{i-1}_{i,j,k}\left[d_{i-1,j,k}\right]^{-1} a^{i+1}_{i-1,j,k} - a^{j-1}_{i,j,k}\left[d_{i,j-1,k}\right]^{-1} a^{j+1}_{i,j-1,k} - a^{k-1}_{i,j,k}\left[d_{i,j,k-1}\right]^{-1} a^{k+1}_{i,j,k-1} \tag{29}$$

In the context of CFD, a system of equations is solved at each cell location, so the terms appearing in Eqn(29) are written with a "-1" exponent to indicate that they will actually be block matrix inverses rather than simple divisions. ILU(0) is implemented by storing the additional matrix D, which is of the same size as one of the Jacobians and therefore will add $1/7^{TH}$ to the memory storage requirement for three-dimensional simulations.

Preconditioning in parallel deserves special consideration. It is possible to implement ILU(0) decomposition in parallel environments, but significant additional message passing overhead is incurred. Various relaxation preconditioners exist targeted toward parallel environments. Here, we consider a simple modification to standard ILU(0), where the preconditioner is applied just to the block of unknowns assigned to that process and terms requiring messages to be passed are neglected. This leads to additional approximation of the LU decomposition beyond the terms already neglected by ILU(0). The effect of this approach on the solution will be explored in the next section.

III. Mathematical Test Problems to Investigation of GMRES Algorithm Techniques

An independent test code was written to investigate the attributes of the GMRES algorithm without the complexity of the CFD code or parallelization to assess the viability and efficiency of the choices outlined in the previous section. To make this assessment, two linear systems have been selected to test GMRES features. The first test case is a simple system of 10 unknowns that is given in Eqn(30). The matrix is sparse and banded with a structure similar to a structured partial differential equation (PDE). The system has been arbitrarily created to be well posed and simple with a solution vector that contains elements that are roughly on the order of 10^0.

$$\begin{bmatrix} 1 & 2 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 3 & 2 & -1 & 0 & 0 & 0 & -2 & 0 & 0 & 0 \\ 0 & 2 & 3 & -2 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 2 & 4 & 2 & 0 & 0 & 0 & -2 & 0 \\ 0 & 0 & 0 & 1 & 5 & 0 & 0 & 0 & 0 & -1 \\ -1 & 0 & 0 & 0 & 0 & 6 & 0 & 0 & 0 & 0 \\ 0 & -2 & 0 & 0 & 0 & -2 & 3 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & -5 & 4 & 3 & 0 \\ 0 & 0 & 0 & -2 & 0 & 0 & 0 & 1 & 2 & 1 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 3 & 4 \end{bmatrix} \vec{x} = \begin{Bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \end{Bmatrix}, \qquad \vec{x} = \begin{Bmatrix} 5.2905 \\ -1.2044 \\ 4.1560 \\ 2.2268 \\ 0.0575 \\ 1.8818 \\ 3.6534 \\ 2.6055 \\ 6.6670 \\ -2.4859 \end{Bmatrix} \tag{30}$$

The first test that is considered is the use of Gram-Schmidt compared to Householder transformations for the orthogonalization of the monomial basis. As detailed in the previous section, Householder transformations are considerably more expensive, particularly considering parallelization, so it is desirable to employ Gram-Schmidt orthogonalization if possible. The serial test code takes about twice as long to solve the test systems using

Householder transformations as with Gram-Schmidt orthogonalization, which is consistent with the operation count given by Saad. The comparison of the residual history of the two techniques solving the 10x10 matrix with a restart number M=5 is shown in Fig. 1, plotted against the number of restarts. So, for this case, GMRES was restarted 100 times, deriving 5 basis vectors each time and solving a 5x5 transformed system like Eqn(23). The residual has been calculated after each restart of GMRES using Eqn(1) rather than Eqn(25) for two reasons. First, Eqn(25) can have issues with machine precision once the method gets close to convergence. Second, for tests with preconditioning, Eqn(25) computes a transformed residual of the preconditioned system and is therefore not equal to the true residual of the system. In this case, no preconditioner has been applied, so it was noted that both Eqn(1) and Eqn(25) did produce the same result until the method got very close to convergence.

The two techniques perform identically. For double precision math on an x86_64 processor (64-bits), the minimum number that can be resolved is approximately 2.2×10^{-16}. Given that the magnitude of the solution vector is 19.6214, it would be expected that machine precision loss could start occurring somewhere around 4×10^{-15} or so. Once the solutions get near machine zero, the Householder transformation technique seems to perform slightly better, indicating that the Gram-Schmidt technique may be suffering from loss of orthogonality. The test case was recompiled and run with all variables declared as *long double* (128-bits). While this improved the convergence floor to approximately 1×10^{-18} for both methods, it seems that both orthogonalization techniques begin to suffer from loss of orthogonality before reaching machine zero, which should be smaller than 1×10^{-30} for long double declarations.

The effect of the restart number, M, is shown in Fig. 2 for restart numbers of M=2, M=4, and M=8. It is clear that the choice of M has a significant impact on the convergence behavior of this small system. The case where M=2 actually stagnated after decreasing the residual about one order of magnitude and could not improve the solution any further. Higher values of the restart number improve the convergence behavior. As noted already, GMRES will converge in a single iteration (apart from numerical rounding and precision loss issues) if the restart number is equal to the size of the matrix, but this is not a practical situation since the computational and storage savings of the Krylov technique are effectively lost at that point.

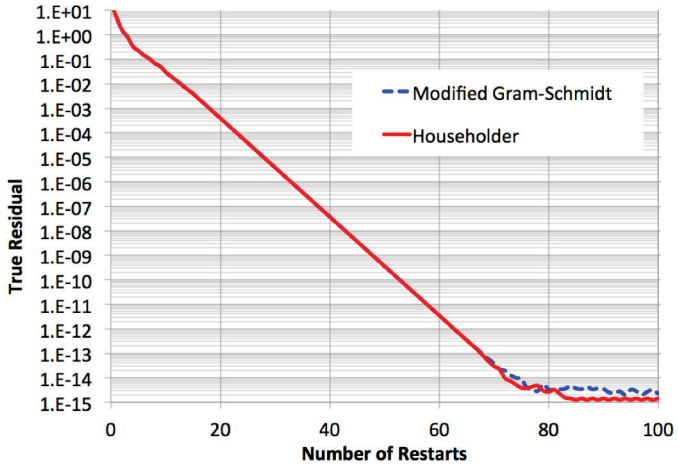


Figure 1. Comparison of Gram-Schmidt and Householder Orthogonalization for 10x10 Test Case from Eqn(30)

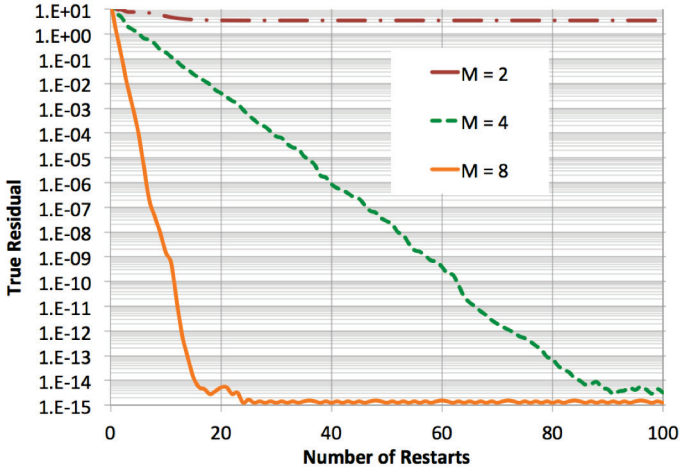


Figure 2. Comparison of Choice of Restart Number, M, for 10x10 Test Case from Eqn(30) using Gram-Schmidt Orthogonalization

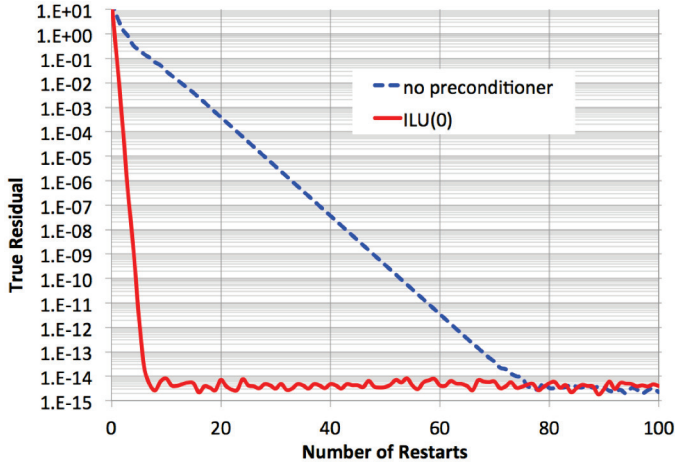


Figure 3. Effect of ILU(0) Preconditioner for 10x10 Test Case from Eqn(30) using Gram-Schmidt Orthogonalization with M=5

The effect of the ILU(0) preconditioner is shown in Fig. 3 compared to the original GMRES solution with Gram-Schmidt orthogonalization and a restart number M=5. The impact of the preconditioner is dramatic, resulting in convergence to the observed machine zero in 6 restart cycles compared to 75 with the original system. The effect of increasing the level of fill of the preconditioner is shown in Fig. 4, where ILU(1) and ILU(2) are compared to the ILU(0) result. Increasing the level of fill does have some positive impact on the solution since ILU(1) reaches machine zero after 4 restarts and ILU(2) reaches machine zero after 3 restarts compared to the 6 cycles by ILU(0). However it is clear that the largest benefit comes from treating the original system with any level of preconditioning, so the extra expense and storage of increasing the level of fill beyond p=0 must be considered on a more practical problem.

Finally, since the preconditioner will actually be implemented in parallel for the real CFD application, the effect of splitting the ILU(0) preconditioner into blocks is considered. This has been accomplished by preconditioning the system of Eqn(30) in pieces. For example, a 2 block preconditioner applies the ILU(0) technique to the upper left 5x5 sub-matrix (consisting of rows and columns 1–5) and separately to the lower-right 5x5 sub-matrix (consisting of rows and columns 6–10) in Eqn(30). This simulates the effect of dropping terms that span processors in the parallel decomposition. The results of considering up to 5 blocks for the preconditioned system with Gram-Schmidt orthogonalization, ILU(0), and a restart number of M=5 are shown in Fig. 5. It is clear that breaking the preconditioner for this case does have a negative effect on the convergence rate, but the system still converges significantly faster than the original solution (which required 75 iterations to reach machine zero). It also seems that increasing the number of breaks in the system beyond two does not further affect the convergence in an adverse way. This result makes sense because it may be noted from Eqn(30) that the entire outer bands will be lost for a break into two blocks, so additional breaks would not result in the loss of many additional terms near the diagonal.

The second test case is a matrix that has been selected as a representative linear system generated by the DPLR CFD code. The DPLR CFD code was run for a number of iterations with a relatively crude grid and the Jacobian and residual matrices were extracted for a typical intermediate iteration from the transient flowfield. The flowfield used is shown in Fig. 6, where the solution at the state where the linear system was extracted is shown on top and the final steady-state solution is shown on the bottom of the image. The case is a simple hemisphere run on a 24x64 cell grid with axisymmetric perfect gas flow (4 equations solved per cell). This results in a linear system composed of 6,144 unknowns. An accurate (to machine zero) solution of the extracted

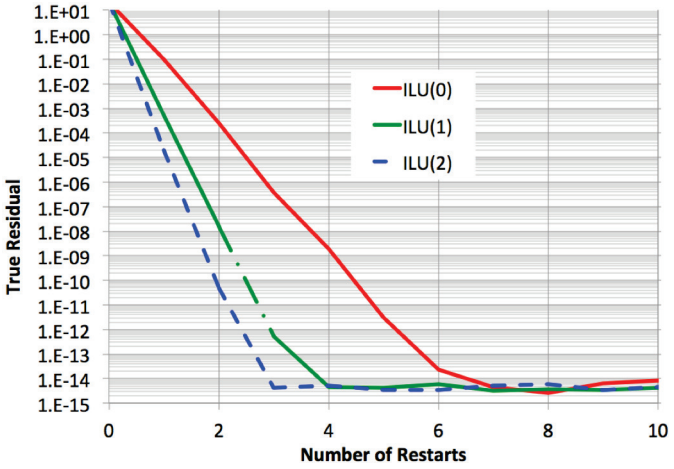


Figure 4. Effect of Level of Fill on ILU(p) Preconditioner for 10x10 Test Case from Eqn(30) using Gram-Schmidt Orthogonalization with M=5

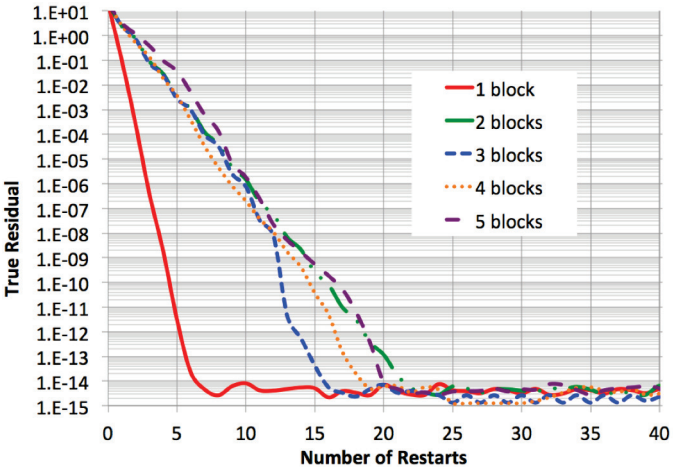


Figure 5. Effect of Applying ILU(0) Preconditioner in Blocks for 10x10 Test Case from Eqn(30) using Gram-Schmidt Orthogonalization with M=5

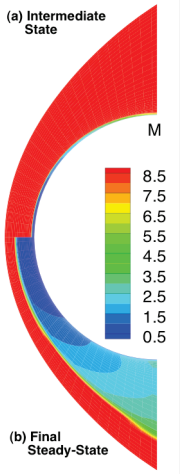


Figure 6. Flowfield Used to Generate Linear System for Hemisphere Test Case

system using LU decomposition shows that the largest unknown has a value of 1.0734×10^{6} and the smallest non-zero unknown has a value of 1.9516×10^{-18}, though several unknowns have a solution of machine zero as well. There is considerable range in the magnitude of the terms of the solution vector. The matrix is sparse with 7504 non-zero blocks out of 2.36 million, or 0.32% filled.

For this larger system, computational cycles took significantly longer. As with the previous test problem, 100 restarts were computed with both Gram-Schmidt and Householder orthogonalizations using a restart number M=10. The result is shown in Fig. 7. In Fig. 7 and all subsequent plots of residual, the residual at each cycle of the test is normalized by the initial residual value of 4.747848×10^{6} to better capture when machine precision loss is observed. After 100 restarts, the magnitude of the relative residual has decreased only to 6×10^{-4}, so it is clear that GMRES would require significantly more cycles to reach machine zero residual convergence. Obtaining the first 100 cycles of both methods required nearly 2 hours of total CPU time on one core of an AMD-2382 processor, so it was deemed unnecessary to continue running the simulation to convergence. Based on the slope of the convergence plot, it might be expected to take at least 1000 more cycles provided GMRES does not stagnate. However, it is observed again that the Gram-Schmidt and Householder methods give identical convergence results and we would expect them to do so until the methods reach near machine zero.

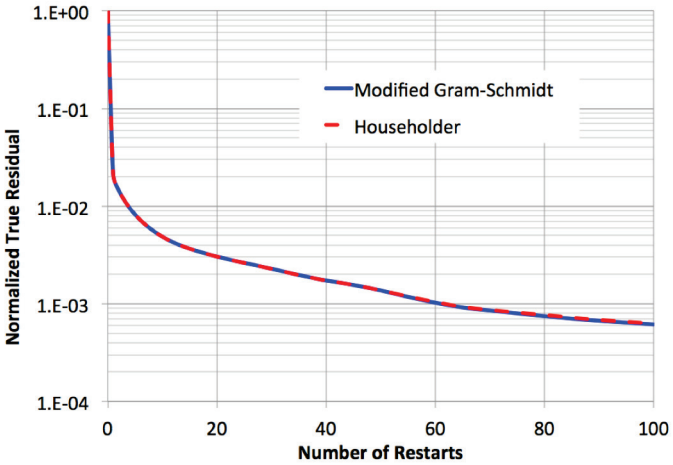


Figure 7. Comparison of Gram-Schmidt and Householder Orthogonalization for Hemisphere Test Case with M=10

The effect of the restart number is shown in Fig. 8, which plots normalized residuals for values ranging from M=2 to M=128 with Gram-Schmidt orthogonalization. After 100 restarts of GMRES, increasing the restart number does have a significant beneficial effect on the convergence of the residual. However, given that the cost of each restart cycle increases with increased M, the convergence has been replotted versus the total number of steps in Fig. 8(b). The total number of steps is defined as the total number of vectors computed, which is equal to the number of vectors computed per cycle (M) times the number of restart cycles performed. When the convergence behavior is viewed in this way, it is clear that increasing the restart number still has a somewhat beneficial effect, but the magnitude of that benefit at any given number of steps is not as large as it seemed from Fig. 8(a). It is also noted that for this case, even a restart number of M=2 shows convergent behavior as far as the case was run. Only the M=128 case showed any significant residual stagnation, which lasted for several restart cycles before subsequently converging at a more rapid rate. This observation emphasises the well-known property of restarted GMRES that numerically it is not guaranteed to have any particular convergence behavior.

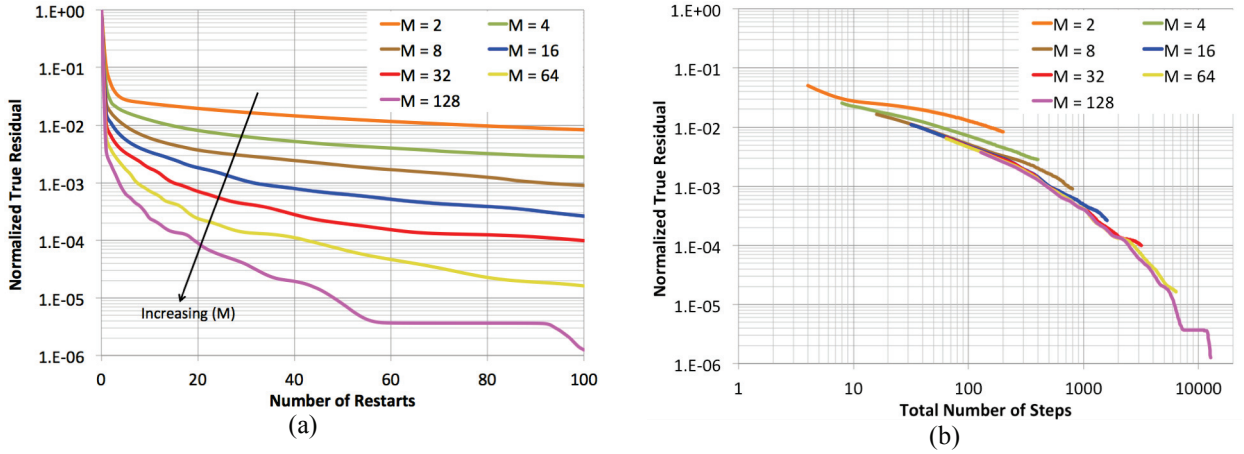


Figure 8. Effect of Restart Number, M, on Hemisphere Test Case using Gram-Schmidt Orthogonalization

The effect of preconditioning on the hemisphere test system is shown in Fig. 9. Preconditioning was shown in the previous test to have a significant effect on the behavoir of GMRES and the effect here is even more dramatic because of the sparseness of the linear system. While the baseline method could only be estimated to require in excess of 1000 restart cycles to reach machine zero, the preconditioned systems require only 2 restart cycles. In addition to ILU(0) preconditioning, ILU(p) solutions up to a level of fill of p=5 are shown, where it is observed that all ILU(p) methods perform approximately equally well. The effect of ILU(p) preconditioning can be more easily seen by looking at the number of steps (vectors computed) for the very first restart cycle only. This comparison is shown in Fig. 10, which plots the normalized residual for the first 10 steps after which the method would be restarted. For GMRES with preconditioning, only the preconditioned residual is readily available as defined in Eqn(25) as each vector is computed. This preconditioned residual is different than the true residual found by applying Eqn(1) to the current estimate of the solution vector, where the difference amounts to multiplying the true residual by the inverse of the preconditioned matrix P. Nevertheless, even the preconditioned residual provides a way to approximately compare the methods at each step as each new basis vector is generated. As with the previous test case, the effect of increasing the level of fill, p, is to improve the convergence behavior of GMRES by marginally reducing the number of steps required to reduce the residual to machine zero. For example, ILU(0) required 8 steps to reach $1x10^{-14}$ while ILU(5) required only 5 steps. However, compared to the improvement of a factor of 1000 times over the non-preconditioned solution that was observed by employing ILU(0), the effect of level of fill is minor.

Finally, the effect of simulating parallelization with ILU(0) is considered by breaking the linear system into sub-blocks and dropping all ILU(0) terms that appear outside of each block. Since the linear system consists of 1536 cells composed of 4 equations each, the splitting of the blocks has been selected to only divide into blocks of whole cells as would be the case for real CFD applications. The results are shown in Fig. 11, which again plots the preconditioned residual against the step number for a single restart cycle. The effect of splitting the system is not as obvious as it was for the first test case since the matrix is so sparse in this case, resulting in a smaller

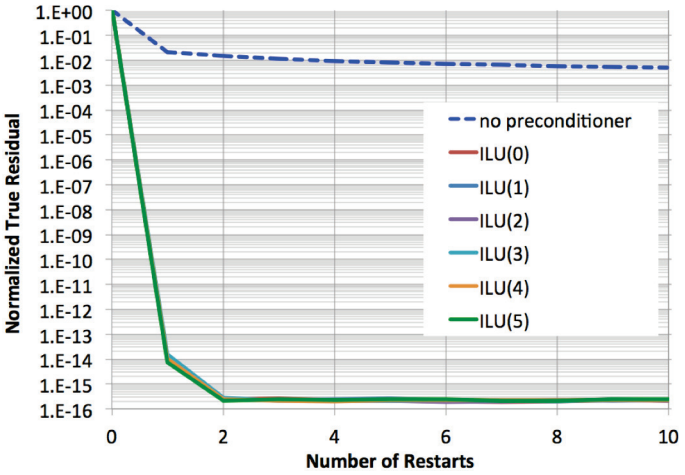


Figure 9. Effect of ILU(p) Preconditioning on True Residual for Hemisphere Test Case with M=10

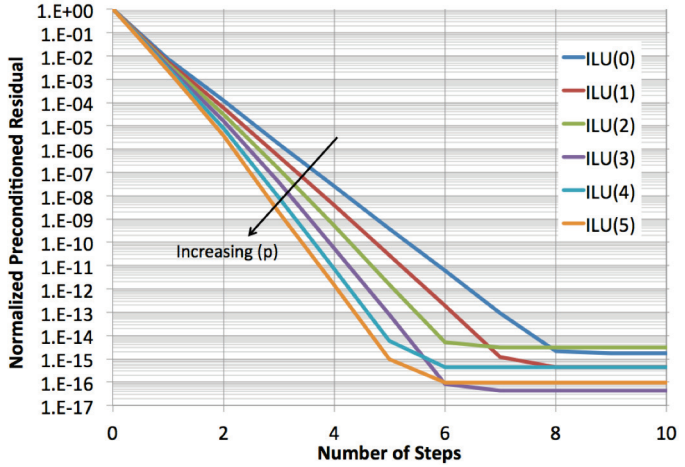


Figure 10. Effect of ILU(p) Preconditioning on Hemisphere Test Case Preconditioned Residual with M=10 for the First Restart Cycle

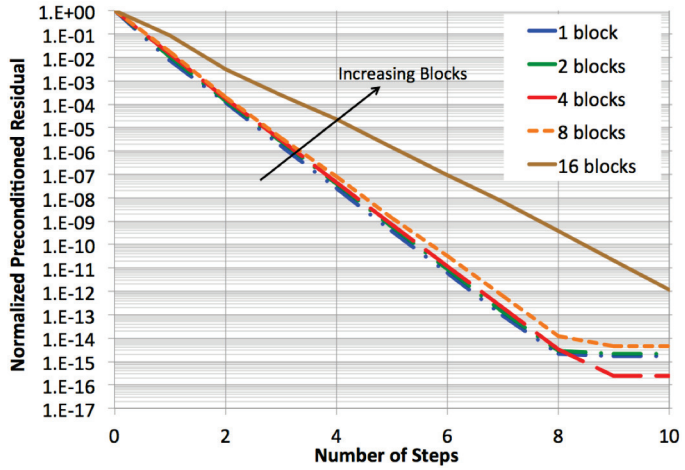


Figure 11. Effect of Simulated Parallelization on Preconditioned Residual for Hemisphere Test Case with M=10 and ILU(0) Preconditioner

fraction of the terms being dropped by the breaking process. Breaking the system into 8 pieces and 16 pieces does produce a noticable increase in the number of steps required to solve the system. This is particularly notable for 16 blocks, which does require about 50% more steps to reach the same level of convergence. However, given that the method with ILU(0) preconditioning converges in only a few steps anyway means that the simple parallelization of the preconditioner does not carry a significant penalty in performance.

The two test cases considered in this section have demonstrated a few things. First, given the significant calculation overhead and inherant complexity of parallelization, there appears to be no advantage to using Householder transformations to orthogonalize the basis vectors. Second, the ILU(0) preconditioner has such a dramatic effect on the convergence behavior of GMRES that it has been decided that increasing the level of fill beyond p=0, while offering a marginal benefit, is not worth the added cost and programming complexity. Third, a simplistic parallelization of ILU(0) obtained by neglecting fill terms that do not reside on a given processor appears to have an acceptable performance penalty that does not outweigh the significant computational cost of fully parallelizing ILU(0) with the extra required message passing. Fourth, to first order, restarted GMRES seems to converge at a rate loosely proportional to the total number of basis vectors computed (number of vectors per restart times the number of restarts). Therefore, since the memory footprint of the algorithm can increase so drastically, with larger M, a default value of M=10 has been set at compile time with a user-defined parameter to control the maximum number of restarts at runtime. These choices have formed the basis for the method coded in the DPLR CFD code that is tested in the next section.

IV. Example CFD Cases with GMRES Implementations

In this section, several CFD cases are examined to compare the performance of the built-in GMRES implementation discussed in the last section to the PETSc implementation of GMRES and the existing line (DPLR) and point (FMDP) relaxation schemes in the DPLR CFD code. Unless otherwise noted, the PETSc GMRES solver uses the default settings provided by the package.

A. Hypersonic Cone

The first case considered is a hypersonic cone with 7^{O} half-angle, a nose radius of 2.5-mm, and approximately 1-m total length. This test case is one that was formerly considered by MacLean, et al.[33] as part of the HIFiRE-1 ground test program. The conditions are from Run 4 of that program, with a freestream specification of ρ=0.125041 kg/m^3, U=1928 m/s, T=213 K, and M=6.59 and with an isothermal wall temperature of T_W=299 K. Perfect gas air and laminar flow are assumed and the geometry is axisymmetric. The grid has been significantly coarsened from the nominal resolution in the original study to 104 cells along the surface by 64 cells normal to the surface with 4 equations solved per cell so that the solution can be computed on a single processor. This corresponds to 26,624 unknowns. The grid is heavily stretched near the wall to resolve heat transfer to the surface with a peak cell aspect ratio of approximately 34,000:1. The converged Mach number field and the mesh used are shown in Fig. 12. Solutions were computed using one, two, and four processors where the mesh was divided into the required number of subdomains by breaking the grid along the axial coordinate (e.g. keeping body normal lines continuous).

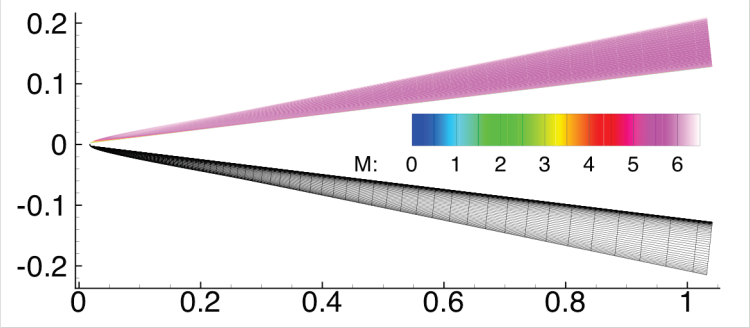
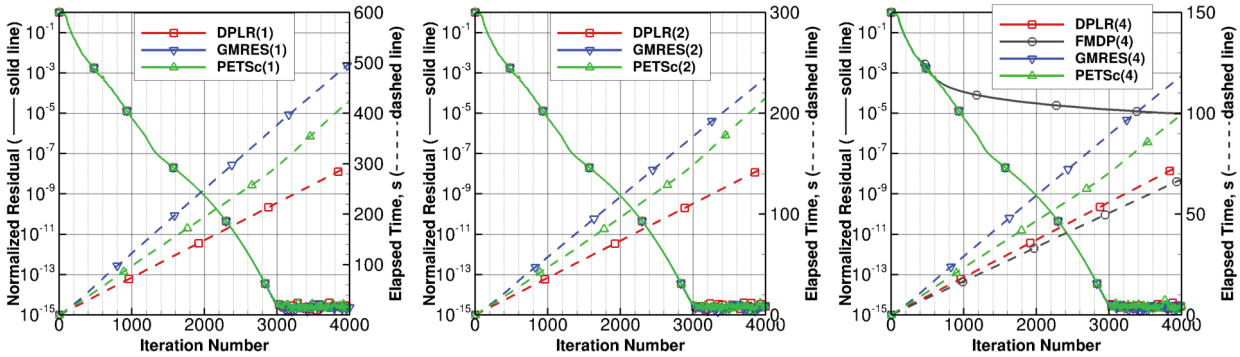


Figure 12. HIFiRE-1 Mach Number Flowfield and Mesh

Since this problem is a hypersonic case with a very thin boundary layer near the surface, the line-relaxation method works very well to converge the system. The usual number of four relaxation steps were used for the DPLR line-relaxation solutions and it was found that increasing this value beyond four had no effect on the convergence rate of the residual. It was found that, in contrast, the point relaxation technique (FMDP) did not perform well at all for this case. Even after increasing the number of relaxation steps to as high as 20, it was found that the FMDP method could not be converged at any CFL higher than 500 at most, whereas the line-relaxation and both GMRES techniques all converged using a CFL of 5000. Therefore, an FMDP solution was only generated on four processors. The fact that both GMRES methods could be converged at the same CFL as the line-relaxation verifies that GMRES is at least as stable as line-relaxation is and helped to validate the implementations against programming or formulation errors.



(a) 1 processor (b) 2 processors (c) 4 processors

Figure 13. Normalized Residuals and Elapsed Times on 1, 2, and 4 Processors for HIFiRE-1 Cone Case

The CFD solver residual histories as a function of iteration number are shown in Fig. 13 for one, two, and four processor parallelizations. The residual histories are plotted on a log scale and are normalized by the initial residual, where the flowfield for all cells is initialized to the freestream state. The residual decreases to machine zero, approximately 1.0×10^{-15}, in approximately 3000 iterations for all cases except the single FMDP case. For each case, the elapsed time is plotted as a function of the iteration number on the right-hand axis of each plot. In general, line-relaxation is the fastest method (as it should be), with the PETSc implementation of GMRES taking approximately 25-35% longer and the built-in GMRES implementation taking about 60-70% longer. The approximate solution time required to drive the residual to machine zero has been extracted for all cases and summarized in Table 1. From this, it can be observed that the line-relaxation method as well as both GMRES implementations scale ideally for this small number of processors. The overhead required to compute a global solution with the GMRES method is not prohibitively expensive, and the neglected terms of the ILU(0) preconditioner do not seem to affect the performance of the built-in GMRES in any way. Again it may be noted that the FMDP method takes one order of magnitude more elapsed time to converge the case than any of the other methods. It is also apparent that both implementations of GMRES are reasonably competitive with line relaxation when considering the inherent extra overhead of calculation involved.

Table 1. Elapsed Times (s) to Converged to Machine Zero for HIFiRE-1 Cone Case

	1 processor	2 processors	4 processors
DPLR	222	110	56
Built-in GMRES	377	177	89
PETSc GMRES	295	149	70
FMDP	-	-	620

One difference between the PETSc implementation of GMRES and the built-in implementation is that PETSc uses classical Gram-Schmidt (CGS) orthogonalization by default instead of modified Gram-Schmidt (MGS). However, PETSc can also be made to use MGS by calling a single function. Since MGS does require more coordinated message passing, it can slow solution speed in parallel applications. The comparison of CGS and MGS using PETSc as the GMRES solver on the four processor grid are shown in Fig. 14. It seems from this particular application that the cost of MGS is not particularly significant. The residual histories of the two orthogonalization techniques are very similar, though there is a very small elapsed time penalty of about 3% to MGS until the solver gets close to machine zero. Once the solver is near machine zero, the CGS solution actually begins to take longer per iteration, which implies that it may be suffering from loss of orthogonality as we might expect. However, since the flowfield is already converged, this loss of orthogonality is unimportant.

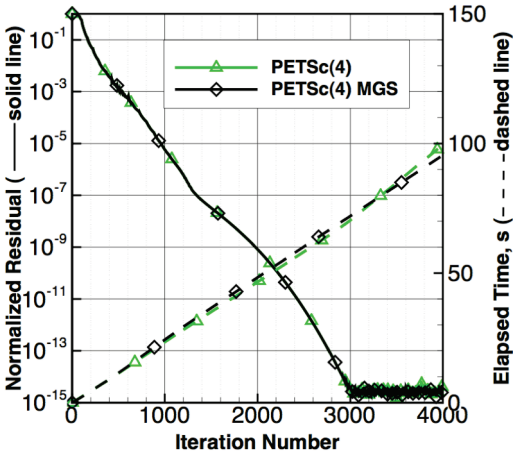
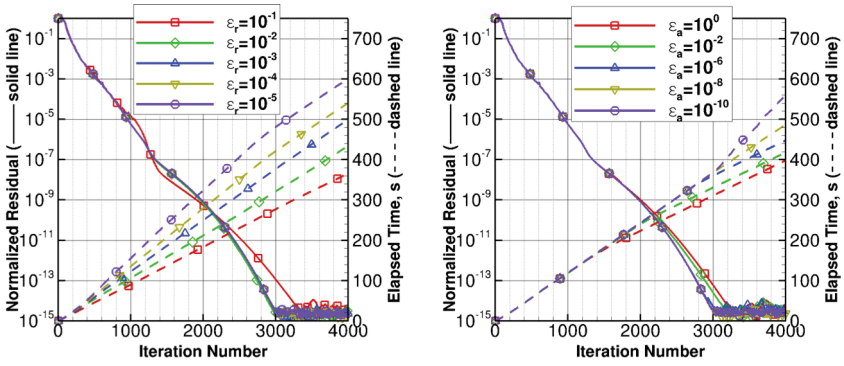


Figure 14. Residual History and Elapsed Time for Classical and Modified Gram-Schmidt Orthogonalization Techniques using PETSc GMRES Solver

The convergence criteria are of greater importance since the number of vectors computed by GMRES is proportional to its elapsed time per iteration. By default, the built-in GMRES implementation operates until either of two criteria is met based on the initial residual of the first iteration of the solution. It should be noted that this initial residual is computed for the preconditioned system and is thus not equivalent to the true residual of the system. These criteria are monitored by computing Eqn (25) for each orthogonal basis vector. The first criterion is that the preconditioned residual decreases by a specified ratio from its initial value. The second criterion is that the preconditioned residual decreases to a specified absolute threshold. This second criterion is employed so that GMRES will not cycle indefinitely when the solution is near machine zero convergence and the initial residual is so small that it cannot be decreased any further. Obviously it is desirable to set these two tolerances to be as large as possible and still maintain good behavior of the solver so that elapsed solution time can be minimized. By default, the built-in GMRES solver has been tested with a relative residual stopping criterion of $1x10^{-3}$ and an absolute residual convergence criterion of $1x10^{-8}$ for this case.

These two criteria have been independently varied by several orders of magnitude in Fig. 15, which again shows CFD residual history and elapsed time for solution on one processor with the built-in GMRES method. The relative convergence criterion shown in Fig. 15(a) shows that increasing the threshold results in less elapsed time per iteration since the method computes less basis vectors. However, the largest threshold, $1x10^{-1}$, begins to display somewhat erratic behavior during CFD solver convergence and does take approximately 10% more iterations to reach machine zero. The rest of the solutions show close agreement in convergence history, noting that the next largest threshold, $1x10^{-2}$, is just visibly different than the rest. This indicates that, for relative convergence thresholds of $1x10^{-3}$ and less, that GMRES is solving the system fully at each flow iteration. A small amount of additional speed could potentially be gained by relaxing this convergence criterion up to one order of magnitude from its default value, but this conclusion is certainly problem dependent and finding a general setting that works for all problems requires more careful testing.



(a) relative convergence (b) absolute convergence

Figure 15. Residual History and Elapsed Time for Absolute and Relative Convergence Criteria using Built-in GMRES Solver

The absolute convergence threshold tests are shown in Fig. 15(b). Again, it is observed that larger values for this threshold results in shorter elapsed times for each flow iteration. This convergence criterion is intended only to affect the solution when the flow is near convergence and the initial residual values considered by GMRES are already near zero. The largest two thresholds are observed to change the solution behavior before the flow reaches machine zero tolerance. While they do require less elapsed time per iteration, this benefit is largely mitigated by the fact that it takes more iterations to reach machine zero. The tightest threshold of $1x10^{-10}$ shows that it affects the behavior of the solution only after 3000 iterations where machine zero residuals are encountered. The value of $1x10^{-8}$ seems to be a reasonable choice although it could potentially be relaxed some as well. As with the relative convergence threshold, testing on other types of problems is required to find a value that best balances the performance of the solver. However, it is apparent that these two convergence criteria heavily influence the overall speed of the method.

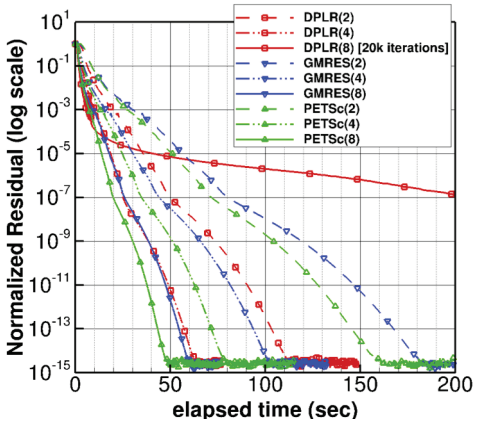


Figure 16. Normalized Residual History over 4,000 Iterations on 2, 4, and 8 Processors Using Body Normal Grid Breaking

One known limitation of the line-relaxation (DPLR) method occurs when the grid is broken for parallel processing along planes normal to the viscous wall (e.g. along constant "J" lines) such that the boundary layer profile lines are not solved on a single processor. For this simple, single-block grid, there is no

need and no reason to break the grid up in such a way, but, for complex topologies with multiply interconnected blocks, breaking a viscous boundary layer zone up across processors is sometimes unavoidable. The hypersonic cone case was repeated by breaking the body-normal lines into two, four and eight processes and solved using line-relaxation, built-in GMRES, and the PETSc GMRES solvers. The normalized residuals are plotted for all cases in Fig. 16 against the elapsed time. With DPLR line-relaxation, the solution converges on two or four processors as before using a maximum CFL of 5000, but eight processors breaks up the boundary layer sufficiently that the solution fails to converge as robustly. For this case, the maximum CFL that could be run was 100 and, after 20,000 flow iterations, the solution has only decreased by approximately seven orders of magnitude. In contrast, both the built-in GMRES and the PETSc solvers behave well for a grid broken into eight processors and converge to machine zero within 4,000 iterations at the same maximum CFL of 5,000. As before, the PETSc solver is somewhat faster than the built-in GMRES, but both perform well. Both methods show good parallel scaling by observing the approximate elapsed time to machine zero, where the built-in GMRES reaches this threshhold in approximately 180s, 100s, and 60s and the PETSc solver reaches machine zero in approximately 160s, 80s, and 50s. This test shows that the GMRES solvers can provide benefit for cases where the viscous boundary layer must be broken across processors.

B. Laminar $25^O/55^O$ Double Cone

A more complex test case is the laminar shock interaction over a $25^O/55^O$ axisymmetric double cone model that has been considered as part of a large, on-going code validation and development study[34]. The interaction of the shock dominated flowfield and the separated region which spans the junction between the two cones has been shown to be exceptionally sensitive to a number of effects including freestream gas state[35], grid resolution, numerics, dissipation[36], non-equilibrium thermodynamics[37], non-equilibrium chemistry[38], and thermochemical coupling[39]. The initial set of experimental test cases were primarily obtained in low energy nitrogen gas which was in vibrational non-equilibrium in the freestream of the facility but otherwise thermally and chemically benign. Run 35 from this test series was investigated in detail by Nompelis, et al.[37] and near perfect agreement between the experimental surface measurements and the CFD simulation was demonstrated by incorporating fine grid density, good numerics, vibrational non-equilibrium, and vibrational accommodation slip. The same Run 35 case is used as a test of the GMRES solver here, where a pointwise specification of the freestream has been incorporated with centerline values of: ρ=0.0006081 kg/m^3, U=2576 m/s, T=102 K, T_V=2711 K, and with an isothermal wall temperature of T_W=298 K. The nozzle calculation to obtain the freestream pointwise distribution has been performed independently of the calculation by Nompelis, et al., but the profile is consistent with their distribution. A computational Schlieren image created by computing the magnitude of the flowfield density gradient is shown in Fig. 17, demonstrating the features of the boundary layer separation, shock interaction, and reattachment process.

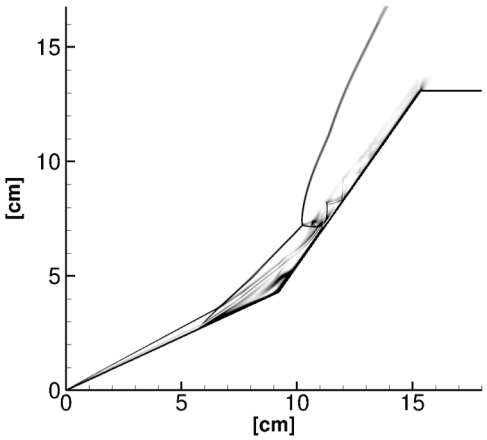


Figure 17. Computational Schlieren Image of Double Cone Flowfield with Run 35 Conditions (dimensions in cm)

On a very dense grid of 1250 x 256 cells solved on 46 processors, the PETSc solver, the built-in GMRES solver, and the DPLR line-relaxation solver were all found to exhibit stable convergence up to a CFL of 2,000 using the same ramp up to the maximum value for all methods. Setting the absolute convergence criterion at $1x10^{-8}$ as described for the previous case, it was quickly found that normalized residual stalling occurred at approximately 10^{-13} for this case as shown in Fig. 18. Increasing the absolute convergence criterion threshhold to $1x10^{-10}$ corrected this behavior. It is also interesting to note that, while the tighter

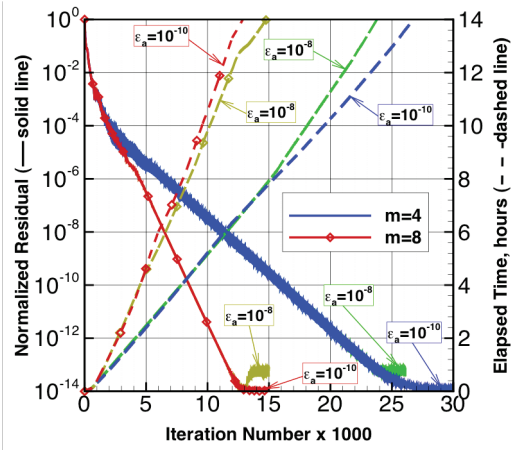
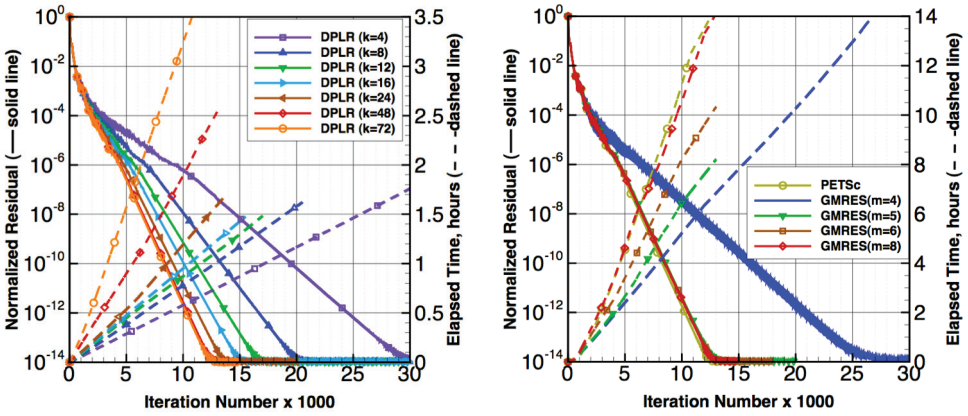


Figure 18. Residual History and Elapsed Time for Absolute Convergence Criterion using Built-in GMRES Solver

absolute convergence criterion took somewhat longer to reach machine zero when using four restarts, it was actually a bit faster when using eight restarts. The tighter convergence criterion has been used for all subsequent comparisons with the double cone case.

For all methods, the code remarkably converged to machine zero. Unlike the previous case, the convergence behavior of the solution techniques was not identical for this flowfield. For the line-relaxation method, a series of tests had to be performed to determine the optimum number of relaxation steps required to converge the flowfield.



(a) line-relaxation (b) built-in GMRES and PETSc

Figure 19. Residual History and Elapsed Time for Run 35 Double Cone using Built-in GMRES Solver, PETSc Solver, and Line-relaxation Solver

This is shown in Fig. 19(a), which plots the normalized continuity residual and the elapsed time for specified relaxation steps ranging from 4 to 72. It is observed that additional relaxation steps up to 48 results in significantly fewer required iterations to converge to machine zero. However, considering the elapsed time to converge to machine zero (each elapsed time line has been truncated at the iteration when this occurs), it is clear that there is an optimum around 16 relaxation steps, where the use of fewer steps requires more convergence time because of the increased number of required iterations and the use of more steps requires more convergence time because of the increased cost per flow iteration. For the built-in GMRES method, the parameter controlling the maximum number of restarts allowed also had to be optimized as shown in Fig. 19(b). The PETSc library uses a much larger number of restarts by default, whereas the built-in GMRES method displayed a poorer convergence rate for fewer than 5 restarts. This is evident both in the larger number of iterations required to reach machine zero and also the small-scale oscillations that occur as the method converges when using 4 restarts. The use of 5 or more restarts gives convergence behavior that is consistent with the PETSc solution. The elapsed times for the most optimal GMRES method requires approximately 8 hours to reach machine zero. The comparison with the optimum convergence with the line-relaxation method of approximately 1.5 hours shows that, for this particular case, the GMRES technique is not competitive in terms of elapsed time because of the large number of restarts required to solve the ill-posed system. The resulting predicted heat transfer distribution for both methods shows the same excellent level of agreement with the experiment that Nompelis, et al. observed for the Run 35 case as shown in Fig. 20.

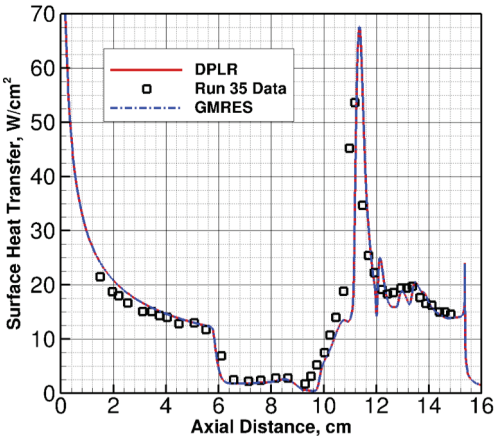
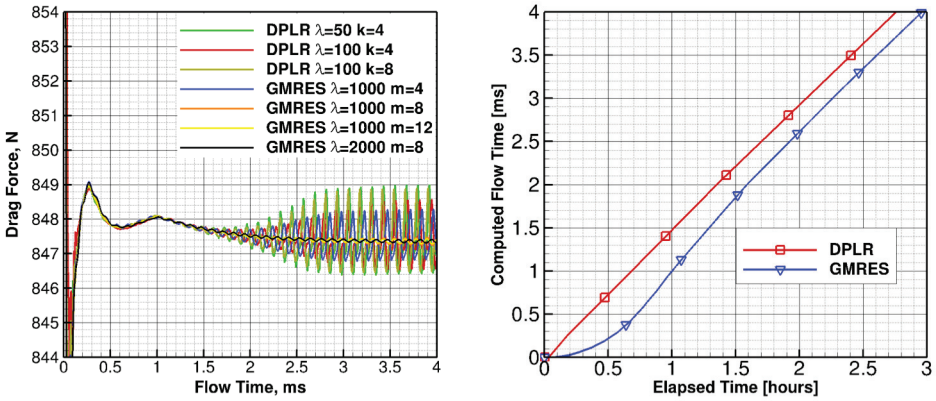


Figure 20. Comparison of Measured Heat Transfer to Experimental Data for Run 35 Using Built-In GMRES and DPLR Methods

Based on the Run 35 case, the GMRES solver does display different, but not better, convergence behavior than the line-relaxation method. However, in testing higher enthalpy cases in preparation for future experiments with the double cone model in the LENS-XX expansion tunnel facility, GMRES was found to exhibit much different convergence behavior in some instances. Assuming freestream conditions of ρ=0.00127 kg/m^3, U=4163 m/s, T=464 K, T_V=464 K with pure nitrogen gas, it was found that the line-relaxation technique could run at a maximum CFL of only 100 without crashing the solver; increasing the number of relaxation steps did not influence this behavior. Using the GMRES method, the same grid and condition was stable at a CFL of 2,000, or a factor of twenty times higher timestep. For these higher enthalpy, higher density conditions, the global density residual does not decrease to machine zero because of shock oscillation in the interaction region. In such cases, another metric of convergence is the monitoring of a global flow metric such as the drag force on the double cone surface. Since both the line-

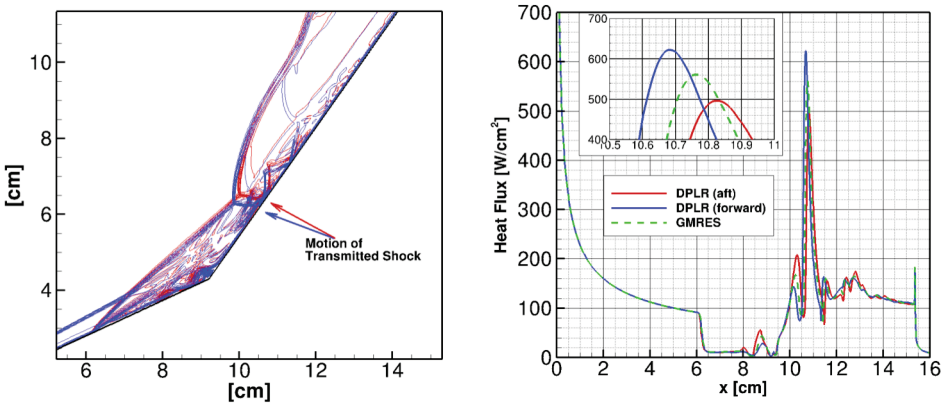
relaxation and GMRES methods use a global timestep of first-order accuracy, the drag force can be plotted against the elapsed flow time. Solutions over 4 ms of flow time are plotted for several methods in Fig. 21(a) covering a number of options. Line relaxation and GMRES solutions were computed using a CFL of 100 (the maximum sustainable for the DPLR technique). It can be observed that in these cases, the drag force on the model undergoes a cyclic oscillation maximizing around 2.75ms that does not damp with flow time. Increasing the number of relaxation steps or GMRES restarts does not affect this behavior. The peak-to-peak magnitudes of the oscillations are 0.12% of the mean value. Unlike the line-relaxation method, GMRES remains stable at a higher CFL up to 2000, which results in a significantly smaller peak-to-peak magnitude of 0.01%. Thus, the GMRES technique shows more stable, steady-state behavior when using twenty times larger timesteps per iteration. The elapsed time for the two methods in Fig. 21(b) shows that DPLR and GMRES at their maximum CFL level compute flow time at approximately the same overall rate for this case.



(a) drag force (b) rate of flow establishment

Figure 21. Convergence of Drag Force for High Enthalpy Double Cone using Line-Relaxation and Built-in GMRES Solver

The finding in this case is that, not only has the built-in GMRES solution method been found to be as fast as line-relaxation, it also converges to a significantly steadier and more stable solution than the line-relaxation does. To assess the solution behavior at the λ=100 timestep, the unsteady time history of the solution was processed to observe the extreme limits of the oscillations to measurable quantities such as the location of the shock structure, shown in Fig. 22(a) and the surface heat transfer profile shown in Fig. 22(b). It is clear from Fig. 22 that, although the separation point on the first cone remains constant, macroscopic motion of the shock system occurs in the reattachment region, resulting in significant changes in the reattachment heating profile of approximately 20% of the mean with the highest reattachment heating level predicted when the shock system swings most forward. The higher CFL solution using GMRES shows no measurable oscillation in the flowfield or on the surface; this heat transfer solution has been also plotted in Fig. 22(b) for reference, which resides midway between the limiting heat transfer solutions of the solution range obtained from the lower CFL of 100. No measured data for this case yet exists, so we cannot make any statements about the physical correctness of any of the solutions obtained. However, we can see from a purely numerical perspective that the GMRES method apparently offers an advantage in solving ill-conditioned linear systems that the line-relaxation method cannot since GMRES alone is capable of generating a stable solution at high CFL.



(a) computational Schlieren (density gradient magnitude) (b) surface heat transfer

Figure 22. Range of Oscillating Solutions Predicted Using Line Relaxation

C. Spherical Capsule at Angle of Attack

As a third test case, the three-dimensional flowfield around a spherical, Apollo-style capsule at angle of attack is considered, which is a test case of significant NASA interest. This test case is the Run 7 condition published by MacLean, et al.[40] in which forebody centerline heat transfer and pressure were measured at 28^{O} angle of attack in low enthalpy (non-reacting) air. Freestream conditions for this case are given as: ρ=0.009122 kg/m^3, U=1805 m/s, T=57 K, and with an isothermal wall temperature of T_W=300 K. The grid is a four block structured grid shown in Fig. 23 that consists of 552,960 cells with 96 cells in the wall normal direction and is solved on 44 processors. This case demonstrates two aspects of the PETSc and built-in GMRES implementations that the first two examples did not: (1) a three-dimensional solution with additional K-direction banding, and (2) the global matrix assembly across multiple physical blocks. This physical flowfield contains both a thin, viscous boundary layer that is presumed to be well suited for line-relaxation and a subsonic inviscid shock layer that has no obvious preferential direction. Cases using line relaxation are solved by preserving lines in the body-normal direction from the surface to the freestream.

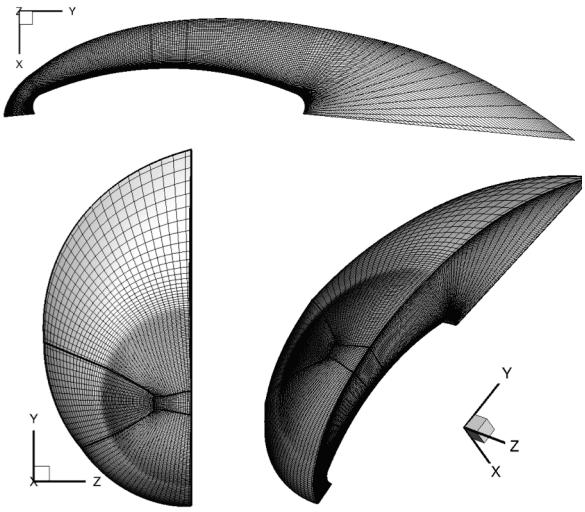


Figure 23. Run 7 Spherical Capsule Four Block Structured Grid

The DPLR CFD code allows the global CFL to be increased every 20 iterations, starting from a very small number up to a specified maximum, so the most aggressive possible ramp has been used for this case. This ramp is plotted in Fig. 24, which increases the CFL up to a value of 1x10^6 after 320 iterations, corresponding roughly to a physical timestep of approximately one millisecond per iteration. This was the largest CFL tested simply because, at such a large physical timestep, it was not possible to increase it further before the solution converged to machine zero. Both the built-in GMRES and the PETSc solver behaved properly at this level. The line relaxation technique using the default setting of four relaxation steps did not, and could only be run at a maximum CFL of 2x10^4. By increasing the number of relaxation steps, it was found that incrementally larger maximum CFL values could be used while maintaining good convergence behavior.

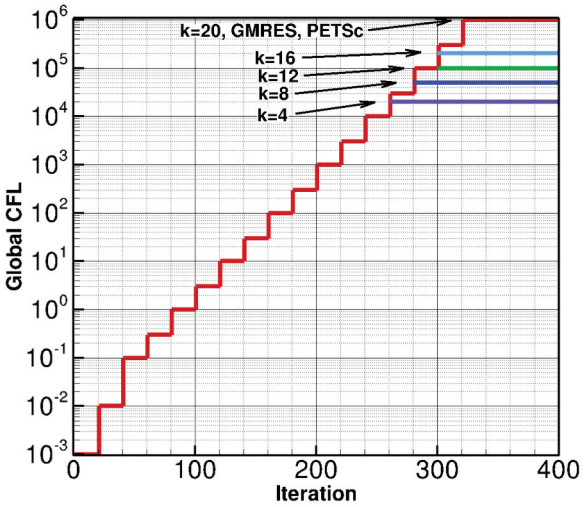
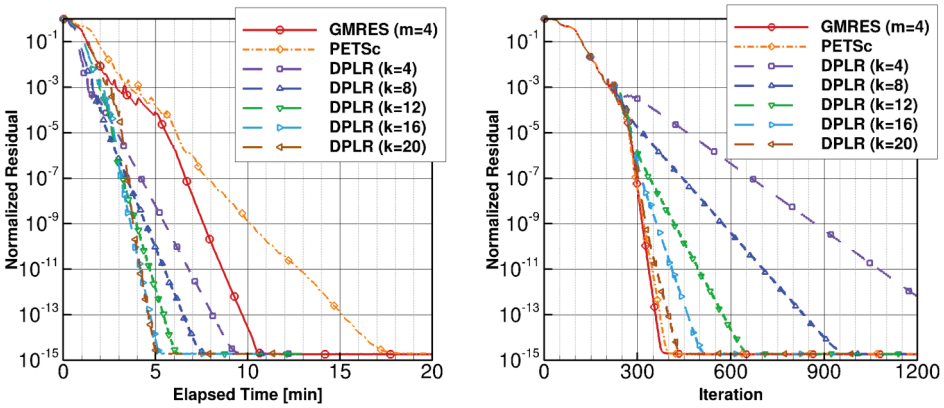


Figure 24. Global CFL Ramp Used for Spherical Capsule Cases

The normalized residual of the continuity equation is plotted against iteration number and against elapsed time in Fig. 25. Here, the PETSc method takes the most elapsed time of all the solver methods tested. The



(a) vs. elapsed time (b) vs. iteration number

Figure 25. Normalized Density Residual for Spherical Capsule Solutions

built-in GMRES method takes approximately 39% less elapsed time than the PETSc solver does for this case, showing again that, despite the use of the block preconditioner, the convergence criteria chosen play an important role in the overall speed of the method. This conclusion is supported by the fact that both the PETSc and the built-in GMRES solvers converge in about the same number of iterations, showing that PETSc is taking more time per iteration to solve the system. The convergence criteria of the PETSc solver were not changed from their default values for this test, but we may presume that a reduced elapsed time might be obtained by sufficiently modifying PETSc's options.

The results with the line relaxation show that the solution of each iteration is significantly faster at the expense of solving the system less accurately. The default setting of four relaxation steps converges in very close to the same elapsed time as the built-in GMRES method despite using about five times as many iterations. Increasing the number of line relaxation steps decreases the elapsed time and the number of iterations to achieve machine zero up to an optimal value of about 16 steps. Above 16 steps, the number of iterations required continues to decrease, but the elapsed time does not. At 20 relaxation steps per iteration, the number of iterations to machine zero is approaching the number of iterations required by the GMRES solvers, indicating that at least this number of relaxation steps is necessary to solve the system as accurately as GMRES solves the system with only four restarts. The summary of the maximum obtained CFL, elapsed time to machine zero convergence, and the number of iterations to machine zero convergence is summarized in Table 2 for all the GMRES and the line relaxation solutions.

Table 2. Maximum CFL, Elapsed Times (m), and Iterations to Converge to Machine Zero for Spherical Capsule Case

	Maximum CFL	Iterations	Elapsed Time, [min]
Built-in GMRES	1,000,000	400	10.7
PETSc GMRES	1,000,000	405	18.0
DPLR, k=4	20,000	1530	9.6
DPLR, k=8	50,000	980	7.6
DPLR, k=12	100,000	665	6.2
DPLR, k=16	200,000	520	5.3
DPLR, k=20	1,000,000	445	5.2

V. Conclusions

Two implementations of the GMRES Krylov subspace solution technique have been added to the chemically reacting DPLR CFD code. There are advantages and disadvantages to using an external linear solver package for applications such as this one. Therefore, the newest version of the DPLR CFD code (4.03.0 or "Big Bend") contains a built-in GMRES solver that uses the existing DPLR CFD code matrices and vectors for the most efficient use of memory. This solver also compiles with the code without requiring users to link to any additional shared libraries or external dependencies. Alternately, users can link to the PETSc library instead of the built-in GMRES solver. The PETSc library has significantly more invested development time and has a number of solver options available for difficult problems. This flexibility is at the expense of additional memory since PETSc uses its own data structures and results in storing the Jacobian terms twice during the solution process.

The results of testing the built-in GMRES solver demonstrated that preconditioning with the ILU(0) technique resulted in improved performance by as much as 1000 times over the non-preconditioned technique (the PETSc implementation also uses a similar ILU preconditioner). Parallelization of the ILU(0) preconditioner by neglecting off-processor terms did not result in a significant performance penalty for the cases tested. The use of Householder transformations to orthogonalize the monomial basis did not result in improved performance so modified Gram-Schmidt orthogonalization was selected as the technique used by the built-in GMRES solver.

Convergence criteria are a major factor in the overall performance of the GMRES methods, where four parameters are of prime importance. A relative residual tolerance parameter controls the fraction of the initial residual below which the system is considered solved. An absolute residual tolerance controls the absolute level of residual below which the system is considered solved to avoid issues with round-off error. The maximum number of restarts controls how many times GMRES is allowed to restart. The number of basis vectors per restart determines the size of the Krylov system and the memory footprint of the method. Although these parameters have been set to default values of $1x10^{-3}$, $1x10^{-8}$, 4, and 10 respectively, test cases have demonstrated that the optimum settings are problem-dependent. In addition, the DPLR line-relaxation technique requires specification of a number of relaxation steps, and it has been found that the default number of four is not optimal for two of the three flowfield studied in this work. A larger value equal to 16 was found to be optimal for these two cases, suggesting that optimization of the line-relaxation method is also heavily problem-dependent.

The performance of the GMRES solvers relative to the line-relaxation solver and the performance of the PETSc against the built-in GMRES solvers depend heavily on the convergence criteria selected for the methods for each particular problem. In general, the GMRES solvers have more computational overhead than line-relaxation and will be somewhat slower for well-tuned problems. The spherical capsule test case showed that the GMRES method can be competitive with the line-relaxation method if care is not taken to properly tune the line-relaxation steps for the problem. The cone test case showed that GMRES significantly outperforms point-relaxation techniques. However, for problems like the double cone flow or the cone with wall normal breaking in which the convergence rate of the line relaxation method is limited, GMRES is an attractive alternative. For any general-purpose CFD code such as this one, no single method will be best for all problems. It is necessary to have multiple options available to users for different problems types, and the implementation of two GMRES techniques in the DPLR CFD code adds a significant alternative capability to the tool.

Acknowledgments

This project was sponsored by grant #NNX10AP41G from NASA Johnson Space Center, technical monitor Brian P. Anderson.

References

[1]Wright, M.J.; Candler, G.V.; and Prampolini, M. "Data-Parallel Lower-Upper Relaxation Method for the Navier-Stokes Equarions," *AIAA Journal*. Vol 34, No 7, Pgs 1371 – 1377. July 1996.

[2]Wright, M.J.; Bose, D.; and Candler, G.V. "Data-Parallel Line Relaxation Method for the Navier-Stokes Equations". *AIAA Journal*. Vol 36, no 9. Pgs 1603 – 1609. Sept 1998.

[3]Candler, G. and Nompelis, I. "Computational Fluid Dynamics for Atmospheric Entry," *Proceedings of AVT-162 RTO AVT/VKI Lecture Series*. RTO-EN-AVT-162 AC/323(AVT-162)TP/279. Paper #15. NATO Research and Technology Organisation: September 2009.

[4]Saad, Youcef and Schultz, Martin. "GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems," *SIAM J. Sci. Stat. Comput*. Vol 7, No 3. Pgs 856 – 869. July 1986.

[5]Balay, S.; Brown, J.; Buschelman, K.; Gropp, W.D.; Kaushik, D.; Knepley, M.G.; Curfman McInnes, L.; Smith, B.F.; and Zhang, H. "PETSc Web page," <http://www.mcs.anl.gov/petsc> 3 April 2011.

[6]Heroux, M.; Bartlett, R.; Howle, V.; Hoekstra, R.; Hu, J.; Kolda, T.; Lehoucq, R.; Long, K.; Pawlowski, R.; Phipps, E.; Salinger, A.; Thornquist, H.; Tuminaro, R.; Willenbring, J.; and Williams, A. "An Overview of Trilinos," <http://trilinos.sandia.gov/index.html>. 4 April 2011.

[7]Kirk, B.S.; Bova, S.W.; and Bond, R.B. "A Streamline-Upwind Petrov-Galerkin Finite Element scheme for Non-Ionized Hypersonic Flows in Thermochemical Nonequilibrium," AIAA Paper 2011–0134, 49TH AIAA Aerospace Sciences Meeting, Orlando, FL: 4 – 7 January 2011.

[8]Nompelis, I.; Wan, T.; and Candler, G.V. "Performance Comparisons of Parallel Implicit Solvers for Hypersonic Flow Computations on Unstructured Meshes," AIAA Paper 2007–4334,18TH AIAA Computational Fluid Dynamics Conference, Miami, FL: 25 – 28 June 2007.

[9]Gropp, W.D.; Keyes, D.E.; McInnes, L.C.; and Tidriri, M.D. "Globalized Newton-Krylov-Schwarz Algorithms and Software for Parallel Implicit CFD," *Int. J. High Performance Computing Applications*, Vol 14(2), Pgs 102 – 136. Summer 2000.

[10]Blanco, M. and Zingg, D.W. "Newton-Krylov Algorithm with a Loosely Coupled Turbulence Model for Aerodynamic Flows," *AIAA Journal*, Vol 45, no 5. Pgs 980 – 987. May 2007.

[11]MacCormack, R.W. and Candler, G.V. "The Solution of the Navier-Stokes Equations Using Gauss-Seidel Line Relaxation". *Computers and Fluids*. Vol 17, No 1. Pgs 135 – 150. 1989.

[12]Candler, G.V. "Chemistry of External Flows". *Aerothermochemistry for Hypersonic Technology*: Von Karman Institute for Fluid Dynamics Lecture Series. VKI LS 1995-04.

[13]Landau, L. and Teller, E. "Theory of Sound Dispersion". *Physikalische Zeitschrift der Sowjetunion*. Vol 10, no 34. 1936.

[14]Millikan, R. and White, D. "Systematics of Vibrational Relaxation". *Journal of Chemical Physics*. Vol 39, no 12. Pgs 3209 – 3213. 1963.

[15]Camac, M. "CO_2 Relaxation Processes in Shock Waves". *Fundamental Phenomena in Hypersonic Flow*. J.G. Hall Ed. Cornell University Press. Pgs 195 – 215, 1964.

[16]Park, C.; Howe, J.T.; Jaffe, R.J.; and Candler, G.V. "Review of Chemical-Kinetic Problems of Future NASA Missions II: Mars Entries". *Journal of Thermophysics and Heat Transfer*. Vol 8, no 1. Pgs 9 – 23. 1994.

[17]Park, Chul. "Assessment of Two-temperature Kinetic Model for Ionizing Air". AIAA Paper 87-1574. AIAA 22ND Thermophysics Conference. Honolulu, HI: 8-10 June 1987.

[18]Marrone, P.V. and Treanor, C.E. "Chemical Relaxation with Preferential Dissociation from Excited Vibrational Levels". *The Physics of Fluids*, Vol 6, no 9. Pgs 1215 – 1221. September 1963.

[19]Palmer, G.E. and Wright, M.J. "A Comparison of Methods to Compute High Temperature Gas Viscosity". *Journal of Thermophysics and Heat Transfer*. Vol 17, no 2. Pgs 232 – 239. 2003.

[20]Palmer, G.E. and Wright, M.J. "A Comparison of Methods to Compute High Temperature Gas Thermal Conductivity". AIAA Paper 2003-3913. Jun 2003.

[21]Gupta, R.; Yos, J.; Thompson, R.; and Lee, K. "A Review of Reaction Rates and Thermodynamic and Transport Properties for an 11-Species Air Model for Chemical and Thermal Nonequilibrium Calculations to 30000 K". NASA RP-1232. August 1990.

[22]Ramshaw, J.D. "Self-consistent Effective Binary Diffusion in Multicomponent Gas Mixtures". *Journal of Non-Equilibrium Thermodynamics*. Vol 15, no 3. Pgs 295 – 300. 1990.

[23]Saad, Youcef. *Iterative Methods for Sparse Linear Systems*. 2nd Ed. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2003.

[24]"Gram-Schmidt Process." <http://en.wikipedia.org/wiki/Gram_schmidt>. 28 December 2010.

[25]Walker, Homer F. "Implementation of the GMRES Method Using Householder Transformations," *SIAM J. Sci. Stat. Comput*. Vol 9, No 1. Pgs 152 – 163. January 1988.

[26]Hoemmem, Mark. "Communication-Avoiding Krylov Subspace Methods," Ph.D. Diss., University of California: Berkeley, 2010.

[27]Rotella, F. and Zambettakis, I. "Block Householder Transformation for Parallel QR Factorization," *Applied Mathematical Letters*, Vol 12, Pgs 29–34. 1999.

[28]Greenbaum, A.; Rozložník, M.; and Strakoš, Z. "Numerical behavior of the modified Gram-Schmidt GMRES implementation," *BIT Numerical Mathematics*, Vol 37. 1997.

[29]Paige, C.C.; Rozložník, M.; and Strakoš, Z. "Modified Gram-Schmidt (MGS), Least-Squares, and Backward Stability of MGS-GMRES," *SIAM J. Matrix Anal. Appl.* Vol 28, no 1, Pgs 262–284. 2006.

[30]Hindmarsh, A.C. and Walker, H.F. "Note on a Householder implementation of the GMRES method," Tech. Rep. UCID-20899, Lawrence Livermore National Laboratory, 1986.

[31]Z. Bai, D. Hu, and L. Reichtel. "A Newton Basis GMRES Implementation." *IMA Journal of Numerical Analysis*, 14:563–581, 1994.

[32]Erhel, Jocelyne. "A Parallel GMRES Version for General Sparse Matrices," *Electronic Transactions on Numerical Analysis*. Vol 3. Pgs 160 – 176. December 1995.

[33]MacLean, M.; Wadhams, T.; Holden, M.; and Johnson, H. "Ground Test Studies of the HIFiRE-1 Transition Experiment Part 2: Computational Analysis," *Journal of Spacecraft and Rockets*. Volume 45, number 6. Pages 1149-1164. November – December 2008.

[34]Harvey, J.K.; Holden, M.S.; and Wadhams, T.P. "Code Validation Study of Laminar Shock/Boundary Layer and Shock/Shock Interactions in Hypersonic Flow, Part B: Comparison with Navier-Stokes and DSMC Solutions," AIAA Paper No. 2001–1031, 39TH Aerospace Sciences Meeting and Exhibit, Reno, NV: 8 – 11 January, 2001.

[35]Candler, G.; Nompelis, I.; Druguet, M.-C.; Holden, M.; Wadhams, T.; Boyd, I.; and Wang, W.-L. "CFD Validation for Hyperconic Flight: Hypersonic Double-Cone Flow Simulations," 40TH AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV: 14 – 17 January, 2002.

[36]Druguet, M.-C.; Candler, G.; Nompelis, I. "Effect of Numerics on Navier-Stokes Computations of Hypersonic Double-Cone Flows," *AIAA Journal*, Vol 43, no 3. Pgs 616 – 623. March 2005.

[37]Nompelis, I.; Candler, G.; and Holden, M. "Effect of Vibrational Nonequilibrium on Hypersonic Double-Cone Experiments," *AIAA Journal* Vol 41, no 11. Pgs 2162 – 2169. November 2003.

[38]Druguet, M.-C.; Candler, G.; Nompelis, I. "Comparison of Physical Models in Computations of High-Enthalpy Double-Cone Flows". AIAA Paper 2006–3419. 9TH Joint Thermophysics/Heat Transfer Conference. San Francisco, CA: 5 – 8 June 2006.

[39]Olejniczak, J. and Candler, G. "Study of Experiments Sensitive to Vibration-Dissociation Coupling Models". *Proceedings of the 20TH International Symposium on Shock Waves, Vol I*. Sturtevant, B.; Shepard, J.; and Hornung, H. Eds. Singapore: World Scientific Publishing Co, 1996.

[40]MacLean, M.; Mundy, E.; Wadhams, T.; Holden, M.; Parker, R. "Analysis and Ground Test of Aerothermal Effects on Spherical Capsule Geometries." AIAA Paper 2008-4273. 38TH AIAA Fluid Dynamics Conference & Exhibit, Seattle, WA: 22 – 26 June 2008.