IN-34
136516
P.13

NASA Technical Memorandum 105929
AIAA–93–0881
ICOMP–92–22

Preconditioned Conjugate-Gradient Methods for Low-Speed Flow Calculations

Kumud Ajmani
Institute for Computational Mechanics in Propulsion
Lewis Research Center
Cleveland, Ohio

Wing-Fai Ng
Department of Mechanical Engineering
Virginia Polytechnic Institute and State University
Blacksburg, Virginia

and

Meng-Sing Liou
National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio

Prepared for the
31st Aerospace Sciences Meeting and Exhibit
sponsored by the American Institute of Aeronautics and Astronautics
Reno, Nevada, January 11–14, 1993

PRECONDITIONED CONJUGATE GRADIENT METHODS FOR LOW SPEED FLOW CALCULATIONS

Kumud Ajmani*
Institute for Computational Mechanics in Propulsion
NASA Lewis Research Center, Cleveland, Ohio

and

Wing-Fai Ng†
Department of Mechanical Engineering
Virginia Polytechnic Institute and State University, Blacksburg, VA

and

Meng-Sing Liou‡
Internal Fluid Mechanics Division
NASA Lewis Research Center, Cleveland, Ohio

Abstract

An investigation is conducted into the viability of using a generalized Conjugate Gradient-like method as an iterative solver to obtain steady-state solutions of very low-speed fluid flow problems. Low-speed flow at Mach 0.1 over a backward-facing step is chosen as a representative test problem. The unsteady form of the two-dimensional, compressible Navier-Stokes equations is integrated in time using discrete time-steps. The Navier-Stokes equations are cast in an implicit, upwind finite-volume, flux split formulation. The new iterative solver is used to solve a linear systems of equations at each step of the time-integration. Preconditioning techniques are used with the new solver to enhance the stability and convergence rate of the solver, and are found to be critical to the overall success of the solver. A study of various preconditioners reveals that a preconditioner based on the Lower-Upper Successive Symmetric Over-Relaxation iterative scheme is more efficient than a preconditioner based on Incomplete L-U factorizations of the iteration matrix. The performance of the new preconditioned solver is compared with a conventional Line Gauss-Seidel Relaxation (LGSR) solver. Overall speed-up factors of 28 (in terms of global time-steps required to converge to a steady-state solution) and 20 (in terms of total CPU time on one processor of a CRAY-YMP) are found in favor of the new preconditioned solver, when compared with the LGSR solver.

Introduction

Conventional iterative solvers like Line Gauss-Seidel Relaxation[1] (LGSR) and the Approximate Factorization[2] (AF) scheme have enjoyed considerable popularity as tools for solving the large systems of simultaneous, linear equations that appear at each time-step of an implicit time-integration scheme. However, these conventional solvers exhibit several documented weaknesses[3], namely, (i) convergence of LGSR depends on the choice of the relaxation parameter (under-relaxation is often required for stability), (ii) LGSR and AF encounter a maximum time-step restriction (even in an implicit formulation) in complex flow problems, and, (iii) convergence of AF is sensitive to the time-step employed in the time-integration. These weaknesses limit the applicability of these solvers, and they are particularly inefficient when applied to 'stiff' problems (e. g. low-speed flows).

The past few years have seen a resurgence of interest in using other methods like the Preconditioned Conjugate-Gradient Method[4] (PCGM), and its generalizations, as iterative solvers. The earliest instances of the application of PCGMs to fluid flow problems are found in the works of Wong and Hafez[5]. The work in reference 5 used PCGMs to solve the potential flow equations for calculations of flow over transonic airfoils. Wigton et. al[6] used the GMRES[7] algorithm to improve the robustness and convergence of existing CFD codes. Reference 6 used GMRES to obtain efficient potential and Euler solutions for subsonic and transonic flow over airfoils. The work of Venkatakrishnan[8] has provided positive evidence of the viability of preconditioned GMRES for the compressible Navier-Stokes equations. Reference 8 used PCGMs to obtain viscous solutions for subsonic and transonic flow over airfoils.

Inspite of the efforts mentioned above, the family of PCGMs has not attracted the full attention of researchers working in the area of algorithm development for linear system solvers. (i) The power of generalized CGMs (like GMRES) has not been harnessed to develop a common computational tool for low-speed (or incompressible) and compressible flow — the regime of low-speed flows is virtually unexplored. (ii) The issues of how and when to terminate the preconditioned GMRES solver need to be examined. The termination criteria determines the num-

* Research Associate, AIAA Member

† Professor, Senior AIAA Member

‡ Senior Research Scientist, AIAA Member

ber of sub-iterates at each global iteration (or time-step), and directly affects the storage requirements and computational efficiency of the solver. (iii) The performance of preconditioned GMRES (and other related algorithms) needs to be documented against the performance of conventional solvers for upwind schemes, particularly for incompressible flow problems. None of these issues has been examined in detail in the above mentioned references 5–8, or elsewhere in the literature.

This paper has endeavored to examine the issues listed in the above paragraph. An attempt has thus been made to fill a gap in the existing literature by conducting an investigation of the applicability of preconditioned generalized Conjugate Gradient like methods, for solving flow problems in the low-speed regime. This paper also serves to complement the earlier work of the authors[9], where the regime of interest was transonic and hypersonic flow problems. The success of preconditioned CGMs was clearly demonstrated in reference 9 for compressible flow problems. This encouraged the authors to investigate the possibility of using preconditioned CGMs for solving incompressible flow problems, with the goal of developing a universal code for incompressible and compressible flows.

A full and more complete description of the theory will follow this Introduction in the next section. The governing equations of fluid flow, the Conjugate Gradient Methods, and the details of their implementation in this research will be presented. The importance of preconditioning, and some possible preconditioners will be discussed. The results of the application of the new solver to a low-speed flow problems will be presented and compared to results using a conventional line relaxation algorithm (i.e., LGSR) on the same test problem. The results demonstrate the effectiveness of using preconditioned, generalized CGMs as iterative solvers. The final section enumerates the conclusions drawn from this work.

Presentation of Theory

Navier-Stokes Equations

The governing equations of compressible fluid flow in 2-D are the Navier-Stokes equations written as

$$\frac{\partial Q}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} = \frac{\partial F_v}{\partial x} + \frac{\partial G_v}{\partial y} \qquad (1)$$

In this research, the thin-layer form of the above equations is used, i.e. all viscous terms which involve a gradient in the streamwise direction are neglected. The use of the thin-layer equations is justified because the coarseness of the computational mesh used in this research precludes the resolution of the viscous fluxes in the streamwise direction.

The thin-layer equations are transformed from cartesian coordinate form (x,y) to generalized (ξ,η) coordinates, which results in

$$\frac{1}{J}\frac{\partial Q}{\partial t} + \frac{\partial \hat{F}}{\partial \xi} + \frac{\partial \hat{G}}{\partial \eta} = \frac{\partial \hat{G}_v}{\partial \eta} \qquad (2)$$

where

$$Q = [\rho, \rho u, \rho v, \rho e_0]^T \; ; \quad J = \xi_x \eta_y - \xi_y \eta_x$$

$$\hat{F} = \frac{\xi_x F}{J} + \frac{\xi_y G}{J} \; ; \quad \hat{G} = \frac{\eta_x F}{J} + \frac{\eta_y G}{J}$$

$$\hat{G}_v = \left(\frac{\mu}{Re_L}\right)[\hat{g}_{v_1}, \hat{g}_{v_2}, \hat{g}_{v_3}, \hat{g}_{v_4}]^T$$

where ξ_x,ξ_y,η_x,η_y are the metrics of the transformation. In addition,

$$F = F(Q) = [\rho u, \rho u^2 + p, \rho uv, (\rho e_0 + p)u]^T$$

$$G = G(Q) = [\rho v, \rho uv, \rho v^2 + p, (\rho e_0 + p)v]^T$$

$$p = (\gamma - 1)\left[\rho e_0 - \rho\left(\frac{u^2 + v^2}{2}\right)\right]$$

$$\hat{g}_{v_1} = 0, \; \hat{g}_{v_2} = \alpha_1 u_\eta + \alpha_3 v_\eta, \; \hat{g}_{v_3} = \alpha_3 u_\eta + \alpha_2 v_\eta$$

$$\hat{g}_{v_4} = \frac{\alpha_1}{2}(u^2)_\eta + \frac{\alpha_2}{2}(v^2)_\eta + \alpha_2(uv)_\eta + \frac{(a^2)_\eta}{Pr(\gamma - 1)}$$

$$\alpha_1 = 1 + \frac{1}{3}\frac{\eta_x^2}{J}, \; \alpha_2 = 1 + \frac{1}{3}\frac{\eta_y^2}{J}, \; \alpha_3 = \frac{1}{3}\frac{\eta_x \eta_y}{J}$$

The specific heat ratio, γ, is taken to be 1.4. The molecular viscosity is given by μ, a is the speed of sound and Re_L is the Reynolds number per unit length. Nondimensionalization is with respect to the freestream density and velocity. The physical coordinates (x,y) and viscosity are nondimensionalized by a reference length L and the molecular viscosity of the freestream, respectively. It must be remarked that, in this research, no correction has been made to the compressible flow equations to account for the incompressible effects of low-speed flow. This is expected to affect the overall performance of the code (i. e. slower convergence rates for all solvers). However, the quality of the computed solution is not affected, and this is borne out by the excellent comparisons obtained with experimental data. In short, the compressible flow equations are used 'as is' for low-speed flow, and this is consistent with the goal of developing a common computational tool for all regimes of flow (incompressible and compressible).

The governing equations are solved computationally in their integral, conservation law form, using a cell-centered finite volume formulation. Inviscid flux terms are upwinded using Van Leer's[10] flux-splitting scheme. The thin-layer viscous fluxes are evaluated with second order accurate central differences.

Equation 2 can be rewritten in compact form as

$$\frac{1}{J}\frac{\partial Q}{\partial t} = -R \tag{3}$$

where the right-hand-side vector is written as

$$R = R(Q) = \frac{\partial \hat{F}}{\partial \xi} + \frac{\partial \hat{G}}{\partial \eta} - \frac{\partial \hat{G}_v}{\partial \eta}$$

R is called the residual, and equals to zero for a steady state solution. The Euler implicit discretization of equation 3 in time gives

$$\frac{\Delta Q^n}{J\Delta t} = -R^{n+1} \tag{4}$$

where ΔQ^n is the incremental change in the cell-centered values of the vector Q between the $n+1^{\text{th}}$ time level and the known n^{th} time level, i.e.

$$\Delta Q^n = Q^{n+1} - Q^n \tag{5}$$

R^{n+1} is linearized in time about the n^{th} time level which results in

$$\left(\frac{I}{J\Delta t} + \frac{\partial R}{\partial Q}\right)^n \Delta Q^n = -R^n \tag{6}$$

where $I/J\Delta t$ is a block-diagonal matrix and $\frac{\partial R}{\partial Q}$ is a large, sparse, block, banded matrix.

Equation 6 can be rewritten as

$$V^n \Delta Q^n = -R^n \tag{7}$$

Equation 7 represents the system of linear simultaneous algebraic equations that has to be solved at each global time-step in the computation. This system of equations can be solved by direct matrix inversion; however, this requires extensive computer memory and computation effort at each time step. Iterative schemes are attractive because of their computational efficiency and relatively meager memory requirements. In this paper, a comparison is made between the efficiency of two iterative solvers — a conventional Line Gauss-Seidel Relaxation (LGSR) solver and a preconditioned generalized conjugate-gradient type solver. Results of comparisons with the Approximate Factorization (AF) solver may be found in reference 3, but are not presented in this paper.

Generalized Conjugate Gradient Methods

Recall, that we are interested in solving the linear system of equations

$$V^n \Delta Q^n = -R^n \quad \Leftrightarrow \quad Ax = b \tag{8}$$

The classical Conjugate Gradient Method (CGM) was proposed by Hestenes and Steifel[4], to solve the system $Ax = b$, where A is a symmetric and positive definite (SPD) matrix. The idea of using CGMs as iterative methods was first discussed by Reid[11].

The condition that A has to be an SPD matrix is seemingly restrictive because it is difficult to guarantee an SPD matrix for general fluid flow problems which incorporate the Navier-Stokes equations. However, the method can be generalized for solving linear systems where the coefficient matrix is not symmetric and/or positive definite. Several generalizations have been proposed in the literature, particularly for non-symmetric systems, by Saad and Schultz[7], Young and Jea[12], and Axelsson[13], to name a few. The particular generalization used in this paper is the Generalized Minimal RESidual (GMRES) method of Saad and Schultz[7]. As the name of the method suggests, GMRES seeks to minimize the norm of the computed residual vector, r^n, ($r^n \equiv b - Ax^n$) at each iteration.

The GMRES method is directly applicable to solve linear systems with non-symmetric coefficient matrices. The generalization is effected by a two step procedure — the generation of a set of orthonormal vectors from a given initial guess using Arnoldi's Method[14], and the solution of a minimization problem. Arnoldi's Method[14] uses the well known Gram-Schmidt algorithm for computing an orthonormal basis of vectors for the Krylov subspace $K(A, v_1, k) \equiv \text{span}\{v_1, Av_1, \ldots, A^{k-1}v_1\}$.

It is possible to build linear system solvers for sparse matrices, with the help of the Arnoldi process. In order to solve the linear system $Ax = b$, we calculate an approximate solution x_k of the form $x_k = x_0 + z_k$, where x_0 is some arbitrary initial guess to the exact solution $\bar{x}$ ($= A^{-1}b$). The vector z_k lies in the Krylov subspace collectively defined by A, r_0 ($= b - Ax_0$) and k, i.e.,

$$\begin{aligned} z_k \in K(A, r_0, k) &= \text{span}\{r_0, Ar_0, \ldots, A^{k-1}r_0\} \\ &= \text{span}\{r_0, r_1, \ldots, r_k\} \end{aligned} \tag{9}$$

These methods are referred to as Krylov subspace methods.

The iterative scheme based on Krylov subspace methods will be most successful when the iterate x_k minimizes the corresponding residual norm, $\|r_k\|$. Mathematically, this is equivalent to a minimization of $\|r_k\|$ over $z_k \in K(A, r_0, k)$, i.e.,

$$\min_{z_k} \|r_k\| \quad \Leftrightarrow \quad \min_{z_k} \|(b - Ax_k)\| \tag{10}$$

The solution of this minimization problem forms the second part of the GMRES algorithm.

The complete GMRES algorithm can be summarized as follows:

1. For any starting vector x_0, form the initial vector $r_0 = b - Ax_0$; $\beta = \|r_0\|_2$; $v_1 = r_0/\beta$.
2. Generate the basis of orthonormal vectors defined in equation 9

3. Form the approximate solution:
   a) Find the vector z_k which solves the minimization problem of equation 10
   b) Compute $x_k = x_0 + z_k$

In summary, the GMRES method is a minimization process to solve linear matrix systems like $Ax = b$. The minimization proceeds as a sequence of k sub-iterations, and the minimizer is obtained by a simple upper-triangular solve in step 3 of the algorithm. One major practical difficulty with GMRES is that as k increases, both storage and operation cost increase as $O(k)$ and $O(k^2)$, respectively. Hence, the number of sub-iterations has to be restricted to minimize the cost of each global iteration. This issue, which has not received much attention in the literature, is addressed later in this paper.

Further mathematical details, implementation techniques and convergence analyses of the method are contained in references 3 and 8. Since this paper is also concerned with convergence acceleration, it may be remarked that the speed of convergence of the algorithm depends on the condition number of the matrix A ($\kappa_2(A)$), and the distribution of singular-values of A (i.e. the spectrum of A). $\kappa_2(A)$ is defined as the ratio of the maximum to minimum singular-value of the matrix A. If $\kappa_2(A)$ is large and/or the spectrum of singular-values of A is wide and scattered, A is said to be poorly conditioned, and convergence may be very slow. Preconditioning is employed to improve the conditioning of the coefficient matrix. The preconditioning technique chosen may hence be critical to the success of any generalized CGM used to solve linear systems of equations.

Preconditioning Techniques for CGMs

One of the most effective iterative methods for solving large, sparse linear systems of equations is a combination of a generalized Conjugate Gradient like procedure with some appropriate preconditioning technique. Assuming that a preconditioning matrix M is used on the left of the original unpreconditioned system, this involves solving the preconditioned linear system

$$M^{-1}Ax = M^{-1}b \quad \Leftrightarrow \quad \tilde{A}x = \tilde{b} \qquad (11)$$

instead of the original system $Ax = b$.

The motivation for preconditioning is twofold — to reduce the computational effort required to solve the linearized system of equations at each time-step, and, to reduce the total number of time-steps (or global iterations) required to obtain a steady-state solution. Preconditioning will be cost-effective only if the additional computational work incurred for each sub-iteration is compensated for by a reduction in the total number of iterations to convergence — so that the total cost of solving the overall non-linear system is reduced.

The costs associated with preconditioning can be enumerated as (i) Computing the preconditioning matrix M, (ii) Matrix-vector multiplies or equivalent linear system solves associated with M, and, (iii) Additional computer storage to store M — which may be of the order of storage requirements for the coefficient matrix A. The selection of an 'efficient' preconditioner is motivated by the minimization of the afore-mentioned costs, and is considered crucial to the success of the preconditioned GMRES algorithm.

The cheapest preconditioner is the identity matrix (of appropriate rank). However, for $M = I$, the original, unpreconditioned iterative scheme is recovered! The costliest preconditioner results when $M = A$. This is equivalent to performing a direct solver approach and inverting A to obtain $x = A^{-1}b$ — an approach which is unacceptable for the solution of large, sparse linear systems of equations. It is apparent that a practical preconditioner lies somewhere between the two extremal choices of $M = I$ and $M = A$. It is also evident that M should be in some sense "close" to A — so that $M^{-1}A$ is close to the identity matrix. For effective preconditioning, this means that the eigenstructure of $\tilde{A}$ should be close to that of the identity matrix I, i.e. $\tilde{A}$ should be well-conditioned.

For practical implementations, the 'explicit' computation of M^{-1} is not advisable because (i) Even though M may be sparse (corresponding to the sparsity of A), M^{-1} may be a dense matrix. Storage requirements for M^{-1} may thus far exceed those for M, and, (ii) If M is ill-conditioned (corresponding to ill-conditioning of A), computation of M^{-1} in computer arithmetic will be highly error-prone.

In practice, 'explicit' preconditioning is replaced by an equivalent but highly effective technique called 'implicit' preconditioning. Implicit preconditioning transforms the 'explicit' problem of forming matrix-vector products of the type $M^{-1}u$ ($= \tilde{u}$) to an equivalent 'implicit' problem of solving a linear system for $\tilde{u}$, with M as the coefficient matrix. This can be written as

$$M^{-1}u = \tilde{u} \qquad \Leftrightarrow \qquad M\tilde{u} = u \qquad (12)$$

This linear system has to be solved for each sub-iteration of the preconditioned GMRES algorithm. Thus, any matrix M which produces easy-to-solve (i.e., computationally efficient) linear systems of the type $M\tilde{u} = u$, is a potentially acceptable preconditioner.

In the light of the preceding discussions, a good preconditioning matrix M should

i) produce an iteration matrix $\tilde{A}$ which is better conditioned than A, and
ii) produce easy-to-solve linear systems of the form $M\tilde{u} = u$.

On the basis of the above criteria, several preconditioners that can be derived from the coefficient matrix A are diagonal, block-diagonal, incomplete L-U factorization (ILUF) and block-ILUF[15] — in increasing order of computational cost. Iterative methods used in existing CFD codes can also be used as effective preconditioners[6]. Some examples

are Line Gauss-Seidel Relaxation (LGSR), spatial Approximate Factorization (AF), and variants of the LUSSOR scheme of Yoon and Jameson[16].

The choice of an effective, stable preconditioner is extremely important to the success of GMRES. For this work, several preconditioners were investigated for their effectiveness and stability. An "effective" preconditioner is defined here as one that assists GMRES in obtaining rapid convergence, while requiring a minimal overhead cost for the preconditioning. A "stable" preconditioner is one that can be successfully computed from the matrix A (e. g. the computation of the ILUF of A may be unstable if A is severely ill-conditioned). Diagonal preconditioners — both scalar and block versions — were found to be 'unstable' when applied to the test case in this paper. LGS relaxation and spatial AF, representing the use of existing flow solvers in CFD codes as preconditioners, were found to be 'stable' preconditioners. They were however not as 'effective' in accelerating convergence as the LUSSOR (Lower-Upper Symmetric Successive Over-Relaxation) or the BILUF (Block Incomplete Lower-Upper Factorization with zero fill-in) preconditioners. LUSSOR and BILUF were thus selected as primary preconditioners for use with GMRES in this research. The original LUSSOR scheme of reference[16] was modified, and then incorporated in a block version, for use as a preconditioner in this research. Complete details of LUSSOR and BILUF preconditioning may be found in reference 3.

Implementation of Preconditioned GMRES

In this paper, the preconditioned GMRES algorithm has been implemented for solving the system of equations represented by equation 8. The coefficient matrix A is non-symmetric, banded and sparse. The knowledge of the sparse, banded structure of A is exploited by storing only the five (for a first-order left-hand-side implicit formulation) nonzero block diagonals of the matrix — each element in a diagonal being a 4*4 block matrix. All matrix-vector multiplications are performed using the algorithm of Madsen et. al[17], which uses only the diagonals of the sparse matrix to effect the multiplication. This algorithm is used since it is consistent with the storage scheme for the matrix A in this paper.

As described earlier, GMRES solves the system of linear equations at each global iteration by performing a number of sub-iterations within each global iteration. For the overall efficiency of the preconditioned GMRES solver, the number of sub-iterations, k, needs to be 'limited', at each time-step. One approach often adopted is to fix k[6,8], i.e., perform a predetermined (user-specified) number of sub-iterations at each time step. A different approach, which is used in this research, is to assign a 'stopping criteria' based on the reduction in the norm of the initial residual vector of the linear system, i.e., to terminate the sub-iterations when $||r_k||/||r_0|| \leq \epsilon$, where $\epsilon < 1$. In practice, this translates to truncation of the orthogonalization process after k steps, in step 2 of the GMRES algorithm. The use of a 'stopping criteria' provides a flexible, rather than a fixed k. Thus, the issue is to choose an ϵ (or accuracy value) for each global iteration which will provide a globally stable and efficient iterative scheme.

During this research, it was observed through numerical experimentation that ϵ can be related to the Courant number (λ) of the time-integration by the following relation:

$$\begin{aligned} \epsilon &= 0.5 && 0 < \lambda \leq 10 \\ &= \frac{1}{\log_{10}(\lambda^2)} && \lambda > 10 \end{aligned} \qquad (13)$$

The use of the above criteria for ϵ provides a stable GMRES scheme for the low-speed test case in this research. It is also valid for both the preconditioners (LUSSOR and Block ILUF) used with GMRES. It should be remarked that the use of equation 13 to specify ϵ as the stopping criterion successfully limits k to values below 10, for values of λ upto 1000 (as used in this research). This criteria was also successful when subsequently applied to the transonic and hypersonic flow test cases of reference 9.

Some issues related to storage requirements are now discussed. Recall that the global implicit matrix has a block, banded structure with a known sparsity pattern. All results presented in this work involve a first-order accurate discretization of the implicit operator, thus creating an implicit matrix with five well-defined diagonals. Hence, for a problem size of N (i.e., N grid points), storage corresponding to $5*N$ blocks (each of size 4*4) is required for the implicit coefficient matrix. This storage is required for both the solvers (GMRES and LGSR) tested in this research. The use of preconditioning in conjunction with the GMRES solver necessitates the storage of the corresponding preconditioning matrix, M. The additional storage depends on the particular preconditioner, and varies from N blocks (for block diagonal and LUSSOR preconditioning) to $5*N$ blocks (for block ILU preconditioning with no fill-in).

Recall, that each linear system solve with GMRES requires k sub-iterations, which translates to $k * N * 4$ additional storage locations for the sub-iterate vectors. As k becomes large (for high Courant numbers and stiff problems), the storage cost associated with GMRES may become significant. In this research, it has been demonstrated that the use of equation 13 establishes an upper limit of $k = 10$. The corresponding storage of size $40 * N$ locations has been extracted from existing 'temporary' storage in the code, i.e., storage is shared by GMRES and other subroutines in the code. Hence, the need for additional storage for the sub-iterate vectors has been successfully eliminated in the implementation of preconditioned GMRES in this research.

Test Results and Discussion

The problem of computing low-speed flow over a backward-facing step was selected to demonstrate the effectiveness of preconditioned GMRES over a conventional Line Gauss Seidel Relaxation (LGSR) solver. This flow problem illustrates the phenomena of flow separation and recirculation in internal flows. Extensive experimental and theoretical investigations have been performed for this flow by Armaly et. al[18].

All computations are done for laminar flow and have been performed using the optimum vector processing facilities on a single processor of a Cray-YMP/832. All flow variables are second-order accurate, fully-upwinded in the ξ direction, and third-order accurate, upwind-biased in the η direction. All boundary conditions are first-order accurate. Conserved variables are used for interpolation of cell-centered values to cell-face values. The implicit (left-hand-side), global operator is discretized in a first-order accurate manner. This is done to assure stability of the LGSR solver, and to save on storage and computational costs at each global iteration. All boundary conditions are linearized consistently, and are included in the implicit coefficient matrix.

The computation is started with freestream flow as the initial guess. The relaxation parameter (ω) for LGSR is set equal to one. It must be remarked that $\omega > 1$ (i.e., over-relaxation) is not a stable choice for the LGSR solver for the particular test case investigated here, and $\omega < 1$ (i.e., under-relaxation) can only serve to slow down the convergence rate. Alternate sweeps in both the horizontal and vertical directions are used for stability of the LGSR solver.

Excellent qualitative comparisons with experimental data have been obtained for the test problem examined. However, it is not the intent of this work to present detailed comparisons of computational and experimental data. Detailed code validation results have been presented for the code being used in this research in an earlier paper[19]. The results presented in this work stress the relative performance of the solvers and preconditioners being examined, which is consistent with the overall goals of this paper.

Performance Indices for Comparisons of Solvers

A standard and widely accepted approach to judge the performance of linear system solvers is to study the convergence rate provided by the particular solver. The idea is to examine the rate of convergence of the global non-linear problem as a function of the number of global iterations (or time steps or solution updates). In this research, the comparison of solvers is based on the number of iterations required to reduce the l_2 norm of the residual vector (normalized by the norm of the initial residual vector) of the non-linear problem by ten orders-of-magnitude.

It is often argued, and apparently correctly so, that a 2–3 order-of-magnitude reduction of the non-linear residual is sufficient to obtain solutions within the limits of engineering accuracy. Hence, there seems to be no practical justification in imposing a seemingly strict criteria of a ten orders-of-magnitude reduction for the residual. However, there are several instances when this strict criteria is required, and often necessary.

Firstly, if the initial guess to the solution is close to the true solution, then a greater (than 2–3 orders) reduction in the residual is necessary to provide the correct steady-state solution. Such instances arise when, say, a small perturbation (by changing the set of boundary conditions or the grid) is applied to a known solution (on a particular set of boundary conditions and grid). Secondly, very high residual reductions are often required when results from CFD codes are used as inputs to perform optimization of design parameters. In such cases, results to engineering accuracy are unacceptable since they may introduce large errors while evaluating sensitivities of critical design parameters. Thirdly, if the underlying non-linear problem is stiff, then a 2–3 orders residual reduction may not be sufficient, even to provide results to engineering accuracy. This has been clearly illustrated by the results of the backward-facing step test case in this research.

One of the issues that can determine the success of a solver is the amount of computational or Central Processing Unit (CPU) time required by the solver to perform each global iteration. The CPU time of an algorithm can be heavily influenced by the skills of the individual programmer. Efficient implementations often require investments in 'real' time and patience by the programmer. In addition, an implementation on one particular machine may run faster or slower on another machine, i.e., the CPU time may be machine dependent. Matters are further complicated by factors like compiler optimization, vectorization and parallelization of the algorithm.

All the test results presented in this research were obtained with vector implementations of algorithms executed on a computer with vector processing facilities. Hence, the CPU time comparisons of the preconditioned GMRES and LGSR solvers (and the comparisons of preconditioners) do account for the intrinsic differences in their levels of vectorization. Parallel and distributed computing can also afford rapid reductions in the 'turn-around' time of an algorithm. This research has focussed on vectorization issues in detail, and no efforts have been presently made to parallelize any of the solvers or preconditioners. It is however well accepted that Conjugate Gradient like algorithms can be successfully adapted for use with parallel architectures[20].

It may be remarked that the number of iterations to convergence is completely independent of all the factors that influence the CPU time, and can thus be said to reflect the true convergence characteristics of a solver. The CPU time comparisons are useful, and in conjunction with the number of iterations comparisons, provide an estimate of the

practical utility of a solver.

Details of Numerical Computations

Armaly et. al[18] have presented detailed measurements of velocity distribution and reattachment length for the incompressible flow of air downstream of a single backward facing step in a 2-D channel. The results show that the various flow regimes (in the Reynolds number range of $70 < Re < 8000$), are characterized by typical variations of separation length with Reynolds number. The Reynolds number is based on twice the height of the inlet channel, and two-thirds of the maximum inflow velocity at the step. The particular test cases chosen for this research corresponds to $Re < 400$, since the experimental data suggests that $Re > 400$ produces 3-D variations or turbulence in the flowfield.

The numerical computations are performed on an H-mesh with 61 and 51 points in the ξ (streamwise) and η (normal) directions, respectively. The grid is shown in Figure 1. Grid points are clustered, both in the normal and streamwise directions, to resolve the various viscous gradients and the reattachment point of the separated flow. A freestream Mach number of $M_\infty = 0.1$ is specified. Four different cases with Reynolds number of $Re_\infty = 100, 200, 300$ and 389 are computed. The thin-layer approximation to the Navier-Stokes equations is used. The full Navier-Stokes terms are not included during the computations because the coarseness of the grid in the streamwise direction will prevent resolution of these terms. Moreover, as will be shown later, extremely accurate physical results are obtained with the thin-layer Navier-Stokes terms.

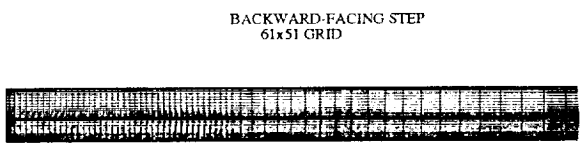

Figure 1. Computational Grid

Adiabatic, no slip boundary conditions are used on the top and bottom walls forming the boundaries of the channel, and on the lower portion (which defines the step) of the inflow boundary. For fully developed subsonic flow at the outflow boundary, three variables (ρ, u and v) are extrapolated and the pressure is determined by fixing the stagnation enthalpy. It must be remarked that this particular outflow boundary condition performed considerably better than a fixed static pressure condition at the outflow.

The computational code used in this research is designed to perform single-grid calculations only. Hence, it is not possible to simulate the entire experimental setup of reference[18], as this would require a minimum of two separate grids. It was thus decided to omit the inlet channel (before the step) in the computations, and simulate its effect by imposing a fully-developed profile for laminar flow, at the step. This, in turn, created a considerable challenge for the specification of the inflow boundary condition at the step.

According to characteristic wave theory, the proper specification of boundary conditions for subsonic inflow requires one physical boundary condition and three numerical boundary conditions to be satisfied. Note that only one physical quantity can be fixed, and the complete specification of a fixed inflow (parabolic in this case) velocity profile requires both components of velocity (u and v) to be fixed. Hence, it seems unlikely that a purely parabolic velocity profile, that will remain fixed as the time-integration proceeds, can be maintained at the inflow boundary.

One approach often used in external flow calculations is to specify a variation of stagnation enthalpy at the inflow, which in turn provides the desired velocity profile. This, however, did not prove to be a stable inflow boundary condition for this internal flow case. Several other variations of the inflow boundary condition were attempted, but none of them yielded satisfactory results. Some of these variations are listed below. Note, that the stagnation enthalpy and entropy are fixed, for all these variations. The subscripts $_b$ and $_i$ refer to boundary and interior values, respectively.

i) Specify profile of Mach number ; $v_b = v_i$
ii) Specify profile of u velocity ; $v_b = v_i$
iii) Specify profile of u velocity ; $p_b = p_i$
iv) Specify profile of total velocity ; $v_b = v_i$
v) $v_b = 0$; $u_b = u_i$
vi) $v_b = 0$; $p_b = p_i$

The problem is resolved by imposing a profile of Reimann invariants at the inflow boundary. The velocity profile obtained with this boundary condition does vary in time, but is sufficiently stable to simulate the incoming flow in an accurate manner. Since the Reimann invariants act as a non-reflecting boundary condition[21], their use helps in rapidly eliminating the initial transients in the solution, as will be shown later by the convergence history results.

Figure 2 shows Mach number contours obtained from the computations on the 61*51 grid for the $Re = 389$ case. The nature and size of the separation and recirculation behind the step closely matches the physical description of the flow as obtained in the experiments of Armaly et. al[18]. Similar results were also obtained for lower Reynolds numbers of $Re = 100$, 200 and 300.

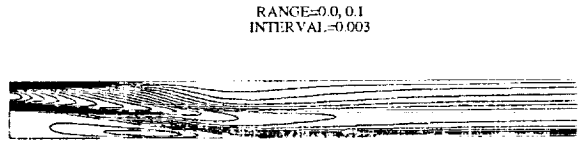

Figure 2. Mach Number Contours

It should be remarked that the reattachment point is the primary flow feature used to characterize the flow in the experimental data[18]. For

$Re = 389$, the experimental results suggest a downstream reattachment length (X_R) to step height (S) ratio of $X_R/S = 7.9$. The numerical computation predicts $X_R/S = 7.95$. The corresponding computed values for $Re = 100$, 200, and 300 are 2.97, 5.09 and 6.58, respectively. The experimental values are 3.1, 5.2 and 6.7, respectively. A comparison of experimental and numerical reattachment lengths is presented in figure 3. It can be seen that the experimental and numerical reattachment lengths are close to each other. This represents particularly good numerical solutions, in light of the fact that onset of reattachment is more difficult to predict than onset of separation. The reattachment point for each case is confirmed by inspecting the velocity profiles for each case. An example set of velocity profiles is shown in figure 4 (for $Re = 389$). Velocity profiles obtained from numerical solutions are compared with available experimental data in figure 5 ($Re = 100$) and figure 6 ($Re = 389$). The differences between the experimental and computed profiles can be attributed to the excessive numerical diffusion introduced by the Van-Leer flux-splitting scheme used in this research. Recent tests with the flux-splitting scheme of Liou and Steffen[22] show much better agreement with experimental data.

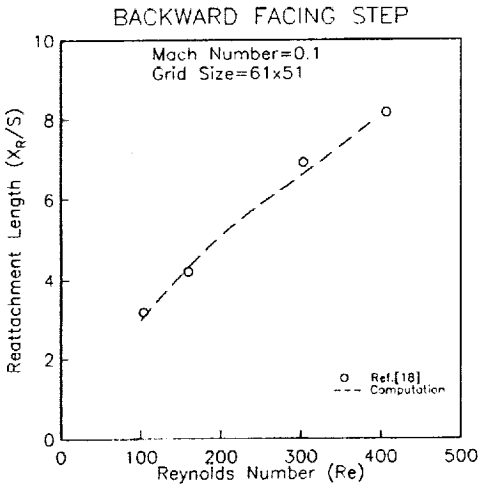

Figure 3. Reattachment Length vs. Reynolds No.

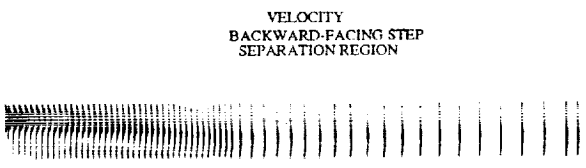

Figure 4. Velocity Vectors for $Re = 389$

Comparisons of Solvers and Preconditioners

Figure 7 presents the convergence history comparisons between GMRES with LUSSOR preconditioning (GMLUS), GMRES with BILUF preconditioning (GMILU) and the Line Gauss-Seidel Relaxation (LGSR) solver. The 'logarithm of the l_2 norm of the residual' has been plotted against the 'number of global iterations' (or time steps). It can be clearly seen that both GMLUS and GMILU converge at a much faster rate than the LGSR solver.

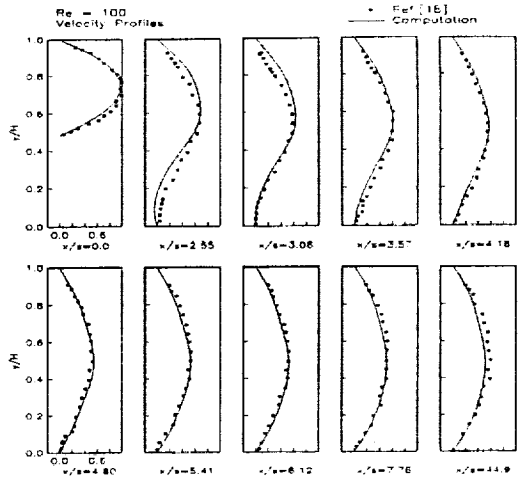

Figure 5. Velocity Profiles for $Re = 100$

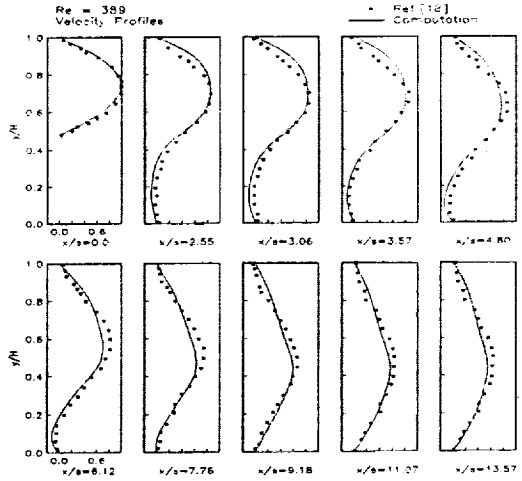

Figure 6. Velocity Profiles for $Re = 389$

The 'correct' physical solution (i.e., the proper reattachment point) is obtained after a six order reduction of the residual is achieved. The rapid reduction in the initial residual (from zero to three orders) is a result of using the Reimann invariants at the inflow boundary (as discussed earlier). This rapid reduction represents elimination of part of the initial transient. The maximum (or asymptotic) Courant numbers (λ) for GMLUS, GMILU and LGSR are 200, 200 and 10, respectively. The maximum λ for LGSR is the value of λ above which LGSR becomes unstable. The maximum λ for GMLUS and GMILU is the value of λ above which the number of subiterations exceeds 10. Curves A (for GMLUS) and B (for GMILU) are generated with an initial $\lambda = 100$, with an increase to $\lambda = 200$ after 100 global itera-

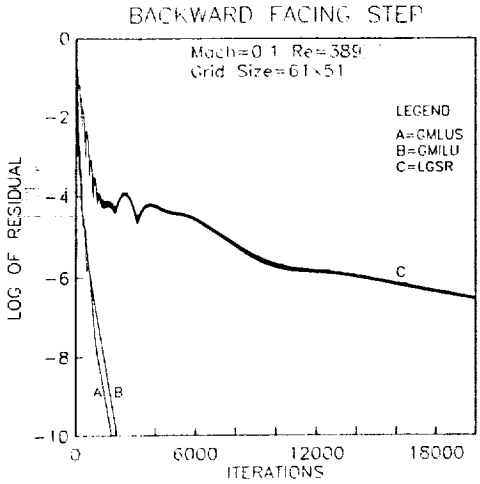

Figure 7. 'Number of Iterations' Comparison

tions. The LGSR solver (Curve C) 'permits' an initial Courant number of $\lambda = 10$, and does not permit any increase in the Courant number (i.e., the iterates diverge at higher values of λ), even after the correct physical solution has been obtained.

GMLUS and GMILU converge in approximately 2000 global iterations. LGSR, however, reaches a six order reduction (i.e., the correct physical solution) in 15,000 global iterations and is estimated to require about 56,000 iterations (according to convergence rate estimates) to attain a ten-order reduction (i.e., convergence). Hence, in terms of the number of overall global iterations to convergence, GMRES (with either preconditioner) is significantly (about 28 times) faster than the LGSR solver.

The use of different values of λ for the preconditioned GMRES and LGSR solvers may raise some question about the 'fairness' of the convergence rate comparisons, particularly since comparatively higher values of λ are used with preconditioned GMRES. The superior convergence rate of the preconditioned GMRES solver can be attributed to two factors — the higher "allowed" values of λ, and the use of preconditioning. The role of preconditioning in accelerating convergence can be determined by using preconditioned GMRES at the same λ as the LGSR solver — the result is that preconditioned GMRES continues to have a superior convergence rate than the LGSR solver, although the speed-up factor is reduced. In the absence of any preconditioning, it was observed that the GMRES solver requires considerably more sub-iterations (than the upper limit of ten sub-iterations with preconditioned GMRES) to provide an overall stable iterative scheme (even at low Courant numbers of order one). The unpreconditioned GMRES is thus impractical as a solver at low Courant numbers, and is unstable at high Courant numbers.

It must be remarked that the use of preconditioning enables the GMRES solver to accept higher values of λ than those "allowed" by the LGSR solver. It is hence only prudent to take advantage of the increased stability afforded by the use of preconditioning, and increase the convergence rate of the preconditioned GMRES solver by using large Courant numbers. The maximum speed-up is thus obtained by using the maximum allowable values of λ. The preconditioned GMRES solver can accept even higher values of the Courant number than those used in this research, but this requires an increase in the number of sub-iterations (and hence, increase in the storage and CPU time per time-step), without any appreciable increase in the overall convergence rate. The choice of λ is thus influenced by the need to obtain the best computational efficiency with the preconditioned GMRES solver, and by the need to limit the storage for the sub-iterates (as discussed earlier).

The CPU time comparison of the solvers is shown in figure 8. The superior efficiency of GMLUS over GMILU and LGSR is seen in this comparison. GMLUS and GMILU require 750 and 3750 seconds, respectively, to converge to a steady-state solution. The reason for this difference in CPU times is that the Block Incomplete L-U Factorization (BILUF) preconditioner used in the GMILU solver requires (partially vectorized) computation of the 'L' and 'U' factors at each global iteration (set-up cost), and subsequent forward and backward (scalar) solves with the 'L' and 'U' factors at each sub-iteration. The block-LUSSOR preconditioner used in the GMLUS solver requires virtually no set-up time, and involves two (partially vectorized) block-diagonal inversions across the domain at each sub-iteration[3]. This huge difference in computational overhead is the reason why GMILU (using BILUF) requires almost five times as much CPU time as GMLUS (using LUSSOR). The LGSR solver is again extremely slow in terms of total CPU time, as it was in terms of number of iterations. LGSR is estimated to require about 15,000 seconds of CPU time for convergence. Thus, LGSR requires about 20 times more CPU time than that required by GMLUS.

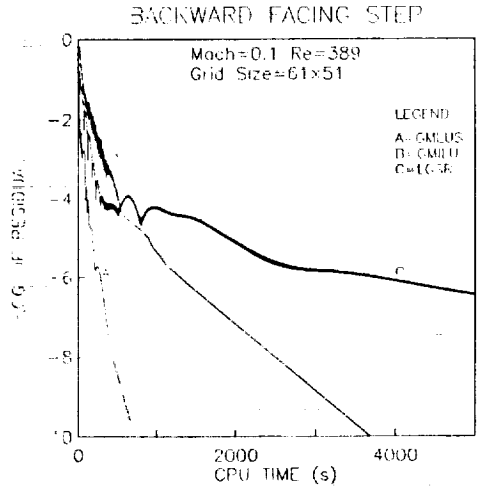

Figure 8. 'CPU Time' Comparison

One of the goals of this research is to identify stable and efficient preconditioners for the GMRES

algorithm, for use with the Navier-Stokes equations. Note, that the 'number of iterations' comparison shows that both the LUSSOR and BILUF preconditioners are stable, and are equally effective in converging to the steady state. The CPU time comparison shows that the use of LUSSOR preconditioning can afford considerable gains in CPU time over the use of the BILUF preconditioner, while maintaining a competitive convergence rate.

The CPU time comparisons reveal that the speed-up factors with the LUSSOR preconditioner are much higher than those for the BILUF preconditioner. The major reason for this difference may be the lack of vectorizability of the BILUF preconditioner. The CPU time for the BILUF preconditioner can be reduced by reusing the 'L' and 'U' factors over a (user fixed) number of global iterations. This translates to performing the incomplete LU decomposition only every i^{th} iteration, and then reusing the incomplete LU for the next $(i-1)$ global iterations. This greatly reduces the overhead cost of computing the BILUF preconditioner at each timestep, and makes BILUF more competitive in terms of CPU time.

A thorough mathematical analysis of the competing solvers in this paper could provide an in-depth explanation of why one solver or preconditioner is better than another. Such an analysis would require a detailed eigenvalue analysis of each iterative scheme. Unfortunately, it is practical to do such an analysis only for small matrices, as may be encountered in 1-D problems or in 2-D problems with very coarse meshes. A coarse mesh for the test problems of this paper could be adopted for this purpose, but then there is no theory to provide a one-on-one correspondence between eigenvalues for a coarse and fine mesh.

It is possible to obtain estimates of extremal (maximum and minimum) eigenvalues for the GMRES solver[23]. However, the usefulness of such estimates is limited and questionable since a) the estimates may not be accurate, b) the estimates are not available when GMRES is preconditioned, and, c) extremal eigenvalues may provide an accurate condition number estimate, but more information (about the distribution of interior eigenvalues) is required to judge the effectiveness of any particular preconditioner. Thus, even though a complete eigenvalue analysis is desirable to fully explain the differences in convergence rates of the solvers, it is impractical for the problem being tested in this research. The development of cheap and accurate methods to perform such eigenvalue analyses could serve as a good source of future work.

Conclusions

The applicability of the preconditioned GMRES solver to low-speed (or incompressible) flows has been demonstrated by computing the incompressible flow over a backward facing step. The new solver performs with considerable success when compared to a conventional Line Gauss-Seidel Relaxation solver. Remarkable speed-ups in convergence rate are afforded by the new solver. When the 'number of iterations to convergence' is used as a comparison criterion, both the preconditioned GMRES solvers (using LUSSOR and BILUF preconditioners, respectively) are approximately 28 times faster than the LGSR solver, for the backward-facing step test case. For 'CPU time to convergence' as a comparison criterion, GMRES with LUSSOR preconditioning is 20 times faster than the LGSR solver.

This research establishes that the LUSSOR (Lower Upper Symmetric Successive Over Relaxation) scheme of Jameson and Yoon[16] can be successfully used as a preconditioner for the GMRES solver, for incompressible flow applications. The modified, block LUSSOR scheme used in this research outperforms the more popular preconditioners based on incomplete factorization(s) of the iteration matrix e.g. BILUF (Block Incomplete Lower-Upper Factorization with zero fill-in). This reinforces the findings of reference[9], where the superiority of the LUSSOR preconditioner was established for compressible (transonic and hypersonic) flows.

The ability of the LUSSOR preconditioner to be as effective in accelerating convergence as the BILUF preconditioner is a significant finding related to the development of preconditioners. BILUF has been one of the most popular preconditioning techniques used by researchers working with conjugate-gradient methods. The present work has shown that the LUSSOR technique can match BILUF in stability and effectiveness, and is considerably more efficient in terms of the overhead costs associated with the preconditioning effort.

One of the major issues raised in context with methods which require sub-iterations to solve linear systems is the stopping criteria used to terminate the sub-iterations. The development of a uniformly applicable stopping criteria for a variety of flow phenomena represents an important step in using preconditioned GMRES as a universal solver. A method for selection of a stopping criteria is obtained in this research. The criteria is based on the Courant number of the global iteration, and can be easily automated for different problems. The criteria is based on empirical evidence, but works well for the incompressible flow test case in this research (the same criteria also worked equally well for compressible flow problems of reference 9).

The success of the preconditioned GMRES solver in accelerating convergence for low-speed flows has been clearly demonstrated. The LUSSOR scheme has been identified as a computationally cheap, yet effective preconditioner. A uniform stopping criteria for the GMRES solver has been developed. All the major results of this research, when combined with the results of reference 9, serve as a significant step towards the development of a universal flow solver for compressible and incompressible flow problems.

Acknowledgements

This work was sponsored as part of the Graduate Research Program in Aeronautics at NASA Lewis Research Center. The authors are indebted to Drs. M. Goldstein and L. A. Povinelli for their support of this research.

References

1. Thomas, J. L., Van Leer, B., and Walters, R. W., "Implicit Flux-Split Schemes of the Euler Equations," AIAA Paper 85-1680, July 1985.

2. Beam, R. M., and Warming, R. F., "An Implicit Factored Scheme for the Compressible Navier-Stokes Equations," *AIAA Journal*, Vol. 16, No. 4, April 1978, pp. 393–402.

3. Ajmani, K., "Preconditioned Conjugate Gradient Methods for the Navier-Stokes Equations," Ph.D Dissertation, Virginia Polytechnic Institute and State University, Blacksburg, VA, 1991.

4. Hestenes, M. R., and Steifel, E., "Methods of Conjugate Gradients for Solving Linear Systems," *J. Res. Nat. Bur. Stand.*, Vol. 49, 1952, pp. 409–436.

5. Wong, Y. S., and Hafez, M. M., "A Minimal Residual Method for Transonic Potential Flow," ICASE Report 81-15, June 1982.

6. Wigton, L. B., Yu, N. J., and Young, D. P., "GMRES Acceleration of Computational Fluid Dynamics Codes," AIAA Paper 85-1494, January 1985 (also in *AIAA Journal*, Vol. 29, No. 7, July 1991, pp. 1092–1100).

7. Saad, Y., and Schultz, M. H., "GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems," *SIAM Journal of Scientific and Statistical Computing*, Vol. 7, 1986, pp. 856–869.

8. Venkatakrishnan, V., "Preconditioned Conjugate Gradient Methods for the Compressible Navier-Stokes Equations," AIAA Paper 90-0586, January 1990.

9. Ajmani, K., Ng, W. F., and Liou, M. S., "Generalized Conjugate Gradient Methods for the Navier-Stokes Equations," AIAA Paper 91-1556, June 1991.

10. Van Leer, B., "Flux Vector Splitting for the Euler Equations," *Lecture Notes in Physics*, Vol. 170, 1982, pp. 507–512 (also ICASE Report 82-30, September 1982).

11. Reid, J. K., "On the Method of Conjugate Gradients for the Solution of Large Sparse Linear Equations," in *Large Sparse Sets of Linear Equations*, Academic Press, New York, 1971, pp. 231–254.

12. Young, D., and Jea, K. C., "On the Simplification of Generalized Conjugate Gradient Methods for Nonsymmetrizable Linear Systems," *Linear Algebra and its Applications*, Vol. 52, 1983, pp. 399–471.

13. Axelsson, O., "Conjugate Gradient Type Methods for Unsymmetric and Inconsistent Systems of Linear Equations," *Linear Algebra and its Applications*, Vol. 29, 1980, pp. 1–16.

14. Arnoldi, W. E., "The Principle of Minimized Iteration in the Solution of the Matrix Eigenvalue Problem," *Quart. Appl. Math.*, Vol. 9, 1951, pp. 17–29.

15. Meijerink, J. A., and Van der Vorst, H. A., "Guidelines for Usage of Incomplete Decompositions in Solving Sets of Linear Equations as they occur in Practical Problems," *Journal of Computational Physics*, Vol. 44, 1981, pp. 134–155.

16. Yoon, S., and Jameson, A., "An LU-SSOR Scheme for the Euler and Navier-Stokes Equations," AIAA Paper 87-0600, January 1987.

17. Madsen, N. K., Rodrigue, G. H., and Karush, J. I., "Matrix Multiplication by Diagonals on a Vector/Parallel Processor," *Info. Proc. Letters*, Vol. 5, 1976, pp. 41–55.

18. Armaly, B. F., Durst, F., Pereira, J. C. F., and Schonung, B., "Experimental and Theoretical Investigation of Backward Facing Step Flow," *Journal of Fluid Mechanics*, Vol. 127, 1983, pp. 473–496.

19. Ng, W. F., Mitchell, C. R., Ajmani, K., Taylor, A. C., and Brock, J. S., "Viscous Analysis of High Speed Flows Using an Upwind Finite Volume Technique," AIAA Paper 89-0001, January 1989.

20. Meurant, G., "Practical Use of the Conjugate Gradient Method on Parallel Supercomputers," *Computer Physics Communications*, Vol. 53, 1989, pp. 467–477.

21. Giles, M. B., "Non-reflecting Boundary Conditions for Euler Equation Calculations," AIAA Paper 89-1942, June 1989.

22. Liou, M-S, and Steffen, C. J., Jr., "A New Flux Splitting Scheme," *Journal of Computational Physics*, to appear.

23. Saad, Y., "Variations on Arnoldi's Method for Computing Eigenelements of Large Unsymmetric Matrices," *Linear Algebra and its Applications*, Vol. 34, 1980, pp. 269–295.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	January 1993	Technical Memorandum

4. TITLE AND SUBTITLE
Preconditioned Conjugate-Gradient Methods for Low-Speed Flow Calculations

5. FUNDING NUMBERS
WU-505-62-21

6. AUTHOR(S)
Kumud Ajmani, Wing-Fai Ng, and Meng-Sing Liou

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135-3191

8. PERFORMING ORGANIZATION REPORT NUMBER
E-7438

9. SPONSORING/MONITORING AGENCY NAMES(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Washington, D.C. 20546-0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
NASA TM-105929

11. SUPPLEMENTARY NOTES
Prepared for the 31st Aerospace Sciences Meeting and Exhibit sponsored by the American Institute of Aeronautics and Astronautics, Reno, Nevada, January 11-14, 1993. Kumud Ajmani, Institute for Computational Mechanics in Propulsion, NASA Lewis Research Center (work funded under Space Act Agreement C99066G); Wing-Fai Ng, Department of Mechanical Engineering, Virginia Polytechnic Institute and State University, Blacksburg, Virginia 24061; Meng-Sing Liou, NASA Lewis Research Center. Space Act Monitor, Louis A. Povinelli.

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified - Unlimited
Subject Category 34

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 words)***

An investigation is conducted into the viability of using a generalized Conjugate Gradient-like method as an iterative solver to obtain steady-state solutions of very low-speed fluid flow problems. Low-speed flow at Mach 0.1 over a backward-facing step is chosen as a representative test problem. The unsteady form of the two-dimensional, compressible Navier-Stokes equations is integrated in time using discrete time-steps. The Navier-Stokes equations are cast in an implicit, upwind finite-volume, flux split formulation. The new iterative solver is used to solve a linear systems of equations at each step of the time-integration. Preconditioning techniques are used with the new solver to enhance the stability and convergence rate of the solver, and are found to be critical to the overall success of the solver. A study of various preconditioners reveals that a preconditioner based on the Lower-Upper Successive Symmetric Over-Relaxation iterative scheme is more efficient than a preconditioner based on Incomplete L-U factorizations of the iteration matrix. The performance of the new preconditioned solver is compared with a conventional Line Gauss-Seidel Relaxation (LGSR) solver. Overall speed-up factors of 28 (in terms of global time-steps required to converge to a steady-state solution) and 20 (in terms of total CPU time on one processor of a CRAY-YMP) are found in favor of the new preconditioned solver, when compared with the LGSR solver.

14. SUBJECT TERMS
Navier-Stokes equations; Linear system solvers; Preconditioning; Conjugate gradients; Convergence acceleration

15. NUMBER OF PAGES
12

16. PRICE CODE
A03

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	