Rend. Sem. Mat. Univ. Pol. Torino
Fascicolo Speciale 1991
Numerical Methods

Z. Johan - T.J.R. Hughes - F. Shakib

A MATRIX-FREE IMPLICIT ITERATIVE SOLVER FOR COMPRESSIBLE FLOW PROBLEMS

Abstract. *A procedure for solving nonlinear time-marching problems is presented. The nonsymmetric systems of equations arising from Newton-type linearizations are solved using an iterative strategy based on the Generalized Minimal RESidual (GMRES) algorithm. Matrix-free techniques leading to reduction in storage are presented. Incorporation of a linesearch algorithm in the Newton-GMRES scheme is discussed. An automatic time-increment control strategy is developed to increase the stability of the time-marching process. High-speed flow computations demonstrate the effectiveness of these algorithms.*

1. Introduction

Over the years, several solvers have been introduced for obtaining steady solutions to the time-dependent Euler and Navier-Stokes equations. Until recently, explicit time-marching solvers have dominated in the CFD community because of their simplicity and their low storage requirements. Unfortunately, the time increment limitation associated with these solvers leads to very costly computations. The role of implicit time-marching solvers has become increasingly important and has gained popularity by the introduction of efficient iterative techniques. Our objective is to develop a robust solver within the framework of a space-time finite element methodology requiring little storage.

An outline of the paper follows: In Section 2, we present a generic implicit time-marching solution algorithm. Some matrix-free solution techniques leading to a substantial reduction in storage are introduced in Section 3.

Linesearch and automatic time-increment control algorithms which increase the robustness of the time-marching scheme are described in Sections 4 and 5, respectively. Numerical experiments in Section 6 illustrate the proposed algorithms. Finally, conclusions are drawn in Section 7.

2. Implicit Time-Marching Solution Algorithm

At each discrete time t_n, space-time finite element discretization of the compressible Navier-Stokes equations leads to the following nonlinear problem [8, 9]:

Given the solution vector $\tilde{\mathbf{v}}_{(n-1)}$ *at time* t_{n-1}, *and a time increment* Δt, *find the solution vector* $\tilde{\mathbf{v}}$ *at time* t_n, *which satisfies the nonlinear system of equations*

$$\tilde{\mathbf{G}}(\tilde{\mathbf{v}};\, \tilde{\mathbf{v}}_{(n-1)},\, \Delta t) = \mathbf{0} \tag{1}$$

$\tilde{\mathbf{G}}$ is a system of nonlinear functionals of $\tilde{\mathbf{v}}$ and of parameters $\tilde{\mathbf{v}}_{(n-1)}$ and Δt. This system is solved for $\tilde{\mathbf{v}}$ by performing a succession of linearizations, leading to Newton's algorithm. This is done via a truncated Taylor series expansion of $\tilde{\mathbf{G}}$:

$$\tilde{\mathbf{G}}(\tilde{\mathbf{v}}^{(i)};\, \tilde{\mathbf{v}}_{(n-1)},\, \Delta t) + \frac{\partial \tilde{\mathbf{G}}}{\partial \tilde{\mathbf{v}}}(\tilde{\mathbf{v}}^{(i)};\, \tilde{\mathbf{v}}_{(n-1)},\, \Delta t)\, \tilde{\mathbf{p}}^{(i)} = \mathbf{0} \tag{2}$$

where the solution increment is given by

$$\tilde{\mathbf{p}}^{(i)} \overset{\text{def}}{=} \tilde{\mathbf{v}}^{(i+1)} - \tilde{\mathbf{v}}^{(i)} \tag{3}$$

$\tilde{\mathbf{v}}^{(i)}$ and $\tilde{\mathbf{v}}^{(i+1)}$ are the approximations of $\tilde{\mathbf{v}}$ at iterations i and $i+1$, respectively.

Denote

$$\tilde{\mathbf{R}}^{(i)} = \tilde{\mathbf{G}}(\tilde{\mathbf{v}}^{(i)};\, \tilde{\mathbf{v}}_{(n-1)},\, \Delta t) \tag{4}$$

$$\tilde{\mathbf{J}}^{(i)} = \frac{\partial \tilde{\mathbf{G}}}{\partial \tilde{\mathbf{v}}}(\tilde{\mathbf{v}}^{(i)};\, \tilde{\mathbf{v}}_{(n-1)},\, \Delta t) \tag{5}$$

$\tilde{\mathbf{R}}$ is the residual of the nonlinear problem and $\tilde{\mathbf{J}}$ is the consistent Jacobian associated with $\tilde{\mathbf{R}}$. The linear systems of equation (2) can be rewritten as

$$\tilde{\mathbf{J}}^{(i)}\, \tilde{\mathbf{p}}^{(i)} = -\tilde{\mathbf{R}}^{(i)} \tag{6}$$

In general, the consistent Jacobian in (6) is replaced by a Jacobian-like matrix $\tilde{\mathcal{J}}$ (see Section 3.1 for the choice of $\tilde{\mathcal{J}}$). Therefore, the linear systems of equations that need to be solved read

$$\tilde{\mathcal{J}}^{(i)} \tilde{\boldsymbol{p}}^{(i)} = -\tilde{\boldsymbol{R}}^{(i)} \tag{7}$$

Hence, the time-marching procedure is performed by looping over the discrete times t_n. At each of these times, the time increment Δt is given by a time-increment control strategy presented in Section 5, followed by the solution of the nonlinear system of equations (1) as described above. The complete algorithm is summarized in Box 1. In this algorithm, $N_{\max}$ is the maximum number of time steps; $i_{\max}$ is the maximum number of Newton iterations; and $\tilde{\boldsymbol{v}}_{(0)}$ is the initial guess of the solution. A detailed study of time-marching schemes for space-time finite element methods is given in Shakib [8].

Box 1 – Implicit Time-marching Solution Algorithm.

Given $N_{\max}$, $i_{\max}$ and $\tilde{\boldsymbol{v}}_{(0)}$, proceed as follows:

(Loop over time)
For $n = 1, 2, ..., N_{\max}$
  (Time-increment control)
  Choose the time increment Δt
  (Newton-type algorithm)
  $\tilde{\boldsymbol{v}}^{(0)} \leftarrow \tilde{\boldsymbol{v}}_{(n-1)}$
  For $i = 0, 1, ..., i_{\max} - 1$
    Solve $\tilde{\mathcal{J}}^{(i)} \tilde{\boldsymbol{p}}^{(i)} = -\tilde{\boldsymbol{R}}^{(i)}$ for $\tilde{\boldsymbol{p}}^{(i)}$
    Compute the linesearch coefficient λ
    $\tilde{\boldsymbol{v}}^{(i+1)} \leftarrow \tilde{\boldsymbol{v}}^{(i)} + \lambda \tilde{\boldsymbol{p}}^{(i)}$
  $\tilde{\boldsymbol{v}}_{(n)} \leftarrow \tilde{\boldsymbol{v}}^{(i_{\max})}$

3. Matrix-free Solution Techniques

The system of equations presented (7) is solved using the Generalized Minimal RESidual (GMRES) algorithm; see Saad and Schultz [7] and Shakib *et al.* [9] for a detailed description and some applications. One advantage of

this iterative algorithm is that the matrix $\tilde{J}$ need not be explicitly formed and stored, but instead a procedure for computing the matrix-vector product $\tilde{J}\,\tilde{u}$ can be used. The coupling of Newton's algorithm and the GMRES algorithm will be referred to as the Newton-GMRES algorithm in the remainder of this paper.

3.1 - *Choice of the Left-hand-side Matrix*

Use of the consistent Jacobian $\tilde{J}$ as the left-hand-side matrix yields in general the fastest convergence to the steady-state solution, provided the initial guess is within the radius of convergence of Newton's algorithm. Experience, however, has shown that for most fluid dynamics problems the use of this matrix leads to very poor stability in the initial phase of the time-marching convergence. One possible way to circumvent this instability is to choose $\tilde{J}$ to be the frozen-coefficient Jacobian for the first few iterations of the time-marching problem. The frozen-coefficient Jacobian is derived by assuming a frozen field, that is the advective and diffusive matrices in the Navier-Stokes equations are assumed constant in the process of differentiating the residual. We have noticed that the frozen-coefficient Jacobian results in a more stable time-marching algorithm, even for an initial guess which is far from the steady-state solution. Once the solution is sufficiently close to the steady-state solution, one can switch to the consistent Jacobian for rapid convergence.

3.2 - *Matrix-vector Products*

The classical way of performing the matrix-vector products in an iterative procedure, such as in the GMRES algorithm, is to initially compute the entries of the left-hand-side matrix $\tilde{J}$ and store them in core memory. Then, the matrix-vector products can be performed explicitly. The use of this technique in the GMRES algorithm is often referred to as the *"linear" GMRES algorithm.* Unfortunately, the nonlinearities in the Navier-Stokes equations make the explicit computation of the consistent Jacobian very difficult and/or CPU-intensive. Only the components of the frozen-coefficient Jacobian can be evaluated at a reasonable cost. The storage requirement for large problems is a drawback of this technique. Nevertheless, the linear GMRES algorithm used with the frozen-coefficient Jacobian has been used successfully in the past and will serve as a reference for the numerical experiments.

Since the matrix $\tilde{\mathcal{J}}$ is a Jacobian-like matrix, it is possible to define a residual-like vector $\tilde{\mathcal{R}}$ such that

$$\tilde{\mathcal{J}}(\tilde{\mathbf{v}})\,\tilde{\mathbf{u}} = \lim_{\varepsilon \to 0} \frac{\tilde{\mathcal{R}}(\tilde{\mathbf{v}} + \varepsilon\,\tilde{\mathbf{u}}) - \tilde{\mathcal{R}}(\tilde{\mathbf{v}})}{\varepsilon} \tag{8}$$

for a given vector $\tilde{\mathbf{u}}$. $\tilde{\mathcal{R}}$ can be either the residual $\tilde{\mathbf{R}}$, in which case $\tilde{\mathcal{J}}$ is the consistent Jacobian, or a modified residual, in which case the left-hand-side matrix is the frozen-coefficient Jacobian. In the latter case, $\tilde{\mathcal{R}}$ is evaluated by using a frozen field. As for the frozen-coefficient Jacobian, the advective and diffusive matrices are always computed at $\tilde{\mathbf{v}}$, even for the vector $\tilde{\mathcal{R}}(\tilde{\mathbf{v}} + \varepsilon\,\tilde{\mathbf{u}})$.

A formula for evaluating the matrix-vector product $\tilde{\mathcal{J}}(\tilde{\mathbf{v}})\,\tilde{\mathbf{u}}$ is derived from (8) by choosing a small ε and dropping the limit, viz.,

$$\tilde{\mathcal{J}}(\tilde{\mathbf{v}})\,\tilde{\mathbf{u}} \approx \frac{\tilde{\mathcal{R}}(\tilde{\mathbf{v}} + \varepsilon\,\tilde{\mathbf{u}}) - \tilde{\mathcal{R}}(\tilde{\mathbf{v}})}{\varepsilon} \tag{9}$$

The choice of ε is critical since sufficient accuracy is needed in evaluating the matrix-vector products using (9). A criterion for determining ε_{opt} is to minimize the error introduced when computing the forward difference approximation (8). Gill *et al.* [5] have derived ε_{opt} in the case of a scalar univariate function and their work can be easily extended to the case of a multivariate vector function. This technique does not require any knowledge of the components of the left-hand-side matrix and it saves a substantial amount of storage. Its implementation into GMRES will be referred to as the *"matrix-free" GMRES algorithm.*

3.3 - *Scaling*

A scaling transformation is required to nondimensionalize and at the same time improve the conditioning of the linear system $\tilde{\mathcal{J}}\,\tilde{\mathbf{p}} = -\tilde{\mathbf{R}}$. Shakib *et al.* [9] have studied different scaling transformations for the problems under consideration and have shown that a nodal block-diagonal scaling is very efficient. Unfortunately, computing the nodal blocks of $\tilde{\mathcal{J}}$ using a finite difference stencil of the form (9) is too expensive. It is, however, possible to get an analytical expression for the nodal blocks of the frozen-coefficient Jacobian. It is also possible to evaluate them at the same time as the residual $\tilde{\mathbf{R}}$ in order to minimize the cost. Although these nodal blocks are only an approximation to the nodal blocks of the consistent Jacobian, experience has shown that they perform almost as well.

Define $\widetilde{W}$ to be the nodal block-diagonal matrix of the frozen-coefficient Jacobian. Since $\widetilde{W}$ is symmetric positive-definite for the problems considered here, it accommodates a Cholesky factorization, denoted as

$$\widetilde{W} = \widetilde{U}^T \widetilde{U} \tag{10}$$

The scaled system of equations can be written as

$$\mathcal{J}\, p = -R \tag{11}$$

with

$$\mathcal{J} = \widetilde{U}^{-T} \widetilde{J}\, \widetilde{U}^{-1}\,, \quad p = \widetilde{U}\tilde{p}\,, \quad R = \widetilde{U}^{-T} \widetilde{R} \tag{12}$$

Moreover, define the scaled residual-like vector $\mathcal{R}(v)$ as

$$\mathcal{R}(v) = \widetilde{U}^{-T} \widetilde{\mathcal{R}}(\widetilde{U}^{-1} v) = \widetilde{U}^{-T} \widetilde{\mathcal{R}}(\tilde{v}) \tag{13}$$

where

$$v = \widetilde{U}\tilde{v} \tag{14}$$

The finite difference approximation (9) reads

$$\mathcal{J}\, p \approx \frac{\mathcal{R}(v + \varepsilon p) - \mathcal{R}(v)}{\varepsilon} = \frac{\widetilde{U}^{-T} \widetilde{\mathcal{R}}(\widetilde{U}^{-1}(v + \varepsilon p)) - \widetilde{U}^{-T} \widetilde{\mathcal{R}}(\widetilde{U}^{-1}(v))}{\varepsilon} \tag{15}$$

To take full advantage of the scaling in finite precision arithmetic, $\mathcal{J}\, p$ should be evaluated by the second equality in (15) as it is presented, i.e., with no simplification. In particular the term $\widetilde{U}^{-1}(v + \varepsilon p)$ should not be replaced by $\tilde{v} + \varepsilon \tilde{p}$. Numerical experiments have shown a substantial degradation in convergence when $\mathcal{J}\, p$ was not implemented as in (15).

4. Linesearch Algorithm

In order to increase the robustness of the Newton-GMRES algorithm, a control over the step length is required. Several techniques for unconstrained optimization have been studied in recent years; see Dennis and Schnabel [2] and Brown and Saad [1] for overviews of these techniques. Following the ideas suggested in [2], we consider a linesearch backtracking algorithm. The idea is to determine the step length by performing the minimization of a univariate function representing a measure of the residual. We give here an

algorithmic description of the method and we refer to the above references for the theoretical details.

Define the functions

$$f(\tilde{v}) = \frac{1}{2}\,\tilde{R}(\tilde{v})^T \lceil \tilde{A}_0^{-1} \rfloor\, \tilde{R}(\tilde{v}) \tag{16}$$

and

$$\hat{f}(\lambda) = f(\tilde{v} + \lambda \tilde{p}) \tag{17}$$

where $\tilde{v}$ is the current solution; $\tilde{p}$ is the step given by the Newton-GMRES algorithm; and λ is the linesearch parameter. The matrix $\lceil \tilde{A}_0^{-1} \rfloor \stackrel{\text{def}}{=} \text{block-diag}(\tilde{A}_0^{-1}, \ldots, \tilde{A}_0^{-1})$ can be viewed as a metric tensor for computing the norm of $\tilde{R}$ as in (16). For the problems under consideration, (16) is similar to energy norms used in structural analysis. It is important to note that computation of the Euclidian norm of $\tilde{R}$ leads to dimensional inconsistency for general definitions of $\tilde{R}$.

We have trivially

$$\hat{f}(0) = f(\tilde{v}) \tag{18}$$

$$\hat{f}(1) = f(\tilde{v} + \tilde{p}) \tag{19}$$

Moreover, the first derivative of $\hat{f}$ at $\lambda = 0$ is

$$\hat{f}'(0) = \nabla f(\tilde{v})^T \tilde{p} = \tilde{R}(\tilde{v})^T \lceil \tilde{A}_0^{-1} \rfloor\, \tilde{J}(\tilde{v})\, \tilde{p} \tag{20}$$

Recall that $\tilde{J}(\tilde{v})$ is the Jacobian of $\tilde{R}(\tilde{v})$. The matrix-vector product $\tilde{J}(\tilde{v})\,\tilde{p}$ is computed via the central finite difference approximation

$$\tilde{J}(\tilde{v})\,\tilde{p} \approx \frac{\tilde{R}(\tilde{v} + \varepsilon\,\tilde{p}) - \tilde{R}(\tilde{v} - \varepsilon\,\tilde{p})}{2\varepsilon} \tag{21}$$

with $\varepsilon = (\varepsilon_M)^{1/3}/\|\tilde{p}\|$. The approximation (21) yields higher accuracy for the evaluation of the matrix-vector product than the forward difference stencil of (9). The choice of ε is also greatly simplified. However, this approximation is twice as expensive as the forward difference scheme.

The objective of the linesearch method is to find $\lambda \in]0, 1]$ such that the Goldstein-Armijo condition

$$\hat{f}(\lambda) \leq \hat{f}(0) + \alpha\,\lambda\,\hat{f}'(0) \tag{22}$$

is satisfied. Here $\alpha \in]0, 1/2]$. It is always possible to find such a λ when $\tilde{p}$ is a descent direction, i.e., $\nabla f(\tilde{v})^T \tilde{p} = \hat{f}'(0) < 0$. Brown and Saad [1]

showed that $\tilde{p}$ is always a descent direction when it is the solution of the Newton-GMRES iteration with the consistent Jacobian. However, use of the frozen-coefficient Jacobian does not always yield a descent direction. If $\tilde{p}$ is found not to be a descent direction, a wise strategy is to repeat the time step with a smaller time increment Δt.

The inequality (22) can be replaced by the approximate condition

$$\hat{f}(1) \leq (1 - 2\alpha)\hat{f}(0) \tag{23}$$

in order to check if the Newton step ($\lambda = 1$) is acceptable, without evaluating $\hat{f}'(0)$. If (23) is not satisfied, the function $\hat{f}$ is approximated by a quadratic polynomial using the data $\hat{f}(0)$, $\hat{f}(1)$ and $\hat{f}'(0)$, whose minimization yields

$$\lambda = \frac{-\hat{f}'(0)}{2(\hat{f}(1) - \hat{f}(0) - \hat{f}'(0))} \tag{24}$$

If (24) does not satisfy the inequality (22), or if λ is too small, a cubic polynomial fitting $\hat{f}(0)$, $\hat{f}(\lambda)$, $\hat{f}(1)$ and $\hat{f}'(0)$ is used to evaluate a new λ. The minimization of the cubic model using the latest data $\hat{f}(0)$, $\hat{f}(\lambda)$, $\hat{f}(\lambda_p)$ and $\hat{f}'(0)$ is repeated until (22) is satisfied; here λ_p is the parameter obtained from the previous polynomial fit. The linesearch parameter may be too large to achieve a reasonable reduction of the step length. It may also be too small, resulting in a slow Newton convergence. Therefore, relative upper and lower bounds η_u and η_ℓ on λ with respect to the previous parameter λ_p are imposed. The complete algorithm is presented in Box 2.

Remark. The study presented in [2] and our own experience have shown that the values $\alpha = 10^{-4}$, $\eta_\ell = 0.1$ and $\eta_u = 0.5$ are appropriate for the problems tested.

Box 2 – Backtracking Linesearch Algorithm.

Given α, η_ℓ, η_u and ε_M, proceed as follows:

$\lambda \leftarrow 1$ and $\lambda_p \leftarrow 1$
Compute $\hat{f}(0)$ and $\hat{f}(1)$
If $\hat{f}(1) \leq (1-2\alpha)\hat{f}(0)$, **Return**
Compute $\hat{f}'(0)$
If $\hat{f}'(0) > 0$, **Return** (reduce Δt)
(Quadratic Approximation)

$$\lambda \leftarrow \frac{-\hat{f}'(0)}{2(\hat{f}(1) - \hat{f}(0) - \hat{f}'(0))}$$

If $\lambda < \eta_\ell \lambda_p$, $\lambda \leftarrow \eta_\ell \lambda_p$
Compute $\hat{f}(\lambda)$
If $\lambda > \eta_\ell \lambda_p$ **and** $\hat{f}(\lambda) \leq \hat{f}(0) + \alpha \lambda \hat{f}'(0)$, **Return**
Repeat
(Cubic Approximation)

$$\begin{Bmatrix} a \\ b \end{Bmatrix} \leftarrow \frac{1}{\lambda - \lambda_p} \begin{bmatrix} \frac{1}{\lambda^2} & \frac{-1}{\lambda_p^2} \\ \frac{-\lambda_p}{\lambda^2} & \frac{\lambda}{\lambda_p^2} \end{bmatrix} \begin{Bmatrix} \hat{f}(\lambda) - \hat{f}(0) - \lambda \hat{f}'(0) \\ \hat{f}(\lambda_p) - \hat{f}(0) - \lambda_p \hat{f}'(0) \end{Bmatrix}$$

$\lambda_p \leftarrow \lambda$
If $a < \sqrt{\varepsilon_M}$,

$$\lambda \leftarrow \frac{-\hat{f}'(0)}{2b}$$

Else

$$\lambda \leftarrow \frac{-b + (b^2 - 3a\hat{f}'(0))^{1/2}}{3a}$$

If $\lambda < \eta_\ell \lambda_p$, $\lambda \leftarrow \eta_\ell \lambda_p$
If $\lambda > \eta_u \lambda_p$, $\lambda \leftarrow \eta_u \lambda_p$
Compute $\hat{f}(\lambda)$
Until $\hat{f}(\lambda) \leq \hat{f}(0) + \alpha \lambda \hat{f}'(0)$

5. Automatic Time-increment Control Algorithm

The linesearch technique presented in the previous section increases the stability of the Newton-GMRES algorithm. However, it does not provide the necessary robustness during the time-marching process. A control over the time increment overcomes this difficulty. Time-increment selection strategies have been applied with success to transient heat conduction analysis by Winget and Hughes [10]. Ericksson and Johnson [3, 4] have developed time-increment control algorithms based on a theoretical approach for linear and nonlinear parabolic problems. The complexity of the Navier-Stokes equations currently prevents us from developing a complete theory. We present here a heuristic automatic time-increment control algorithm based on simple considerations.

Let the correction factor at step n be defined as

$$\gamma(t_n) = \frac{\Delta t}{\Delta t_{\max}} \tag{25}$$

where Δt is the current time increment and $\Delta t_{\max}$ is the user-specified maximum time increment. We want to modify γ at each step so that the rate of change in the solution $\tilde{\mathbf{v}}$ is approximatively constant, i.e.,

$$\gamma(t_n) = \gamma(t_{n-1}) \min\left(\frac{\delta}{\|\Delta\tilde{\mathbf{v}}\|_{\tilde{\mathbf{A}}_0}}, \nu \right) \tag{26}$$

with $\Delta\tilde{\mathbf{v}} \overset{\text{def}}{=} \tilde{\mathbf{v}}(t_{n-1}) - \tilde{\mathbf{v}}(t_{n-2})$; $\|\Delta\tilde{\mathbf{v}}\|_{\tilde{\mathbf{A}}_0} \overset{\text{def}}{=} (\Delta\tilde{\mathbf{v}}^T \lceil \tilde{\mathbf{A}}_0 \rfloor \Delta\tilde{\mathbf{v}})^{1/2}$; δ is a tolerance to be described later; and $\nu > 1$ is a factor that limits the growth of γ. The factor ν eliminates sudden instability that could occur if γ changes too fast from one time step to another. Note that $\gamma \in]0, 1]$ by definition. Consequently, (26) has to be modified as

$$\boxed{\gamma(t_n) = \min\left\{ \gamma(t_{n-1}) \min\left(\frac{\delta}{\|\Delta\tilde{\mathbf{v}}\|_{\tilde{\mathbf{A}}_0}}, \nu \right), 1 \right\}} \tag{27}$$

Equation (27) reduces the correction factor when important changes in the solution occur, for example when shocks appear in the flow field. On the other hand, the correction factor increases to 1 as the solution converges to steady-state. The tolerance δ is defined as

$$\delta = \|\tilde{\boldsymbol{\delta}}\|_{\tilde{\mathbf{A}}_0} \tag{28}$$

where $\tilde{\boldsymbol{\delta}}$ is the tolerance vector. We use a stable and constant Δt for the first N time steps. At the N^{th} time step, $\tilde{\boldsymbol{\delta}}$ is chosen to be the change in the solution, $\tilde{\mathbf{v}}(t_{N-1}) - \tilde{\mathbf{v}}(t_{N-2})$. For most parts, this definition is adequate. However, it does not guarantee that (27) generates stable Δt's for all time steps. A good indicator of possible instabilities is a linesearch parameter $\lambda < 1$, meaning that the Newton step is too large since Δt is too large. The procedure in such a case is to reduce the tolerance vector $\tilde{\boldsymbol{\delta}}$ and the correction factor γ by a factor $\mu < 1$ and proceed to the next step. Finally, a lower bound $\gamma_{\min}$ is imposed on γ to obtain a sufficiently large time increment. The complete algorithm is presented in Box 3.

Remarks:

1. If a local time-increment strategy is used (see Shakib *et al.* [9]), the definition of γ can be changed to

$$\gamma(t_n) = \frac{\text{CFL}}{\text{CFL}_{\max}} \tag{29}$$

   where $\text{CFL}_{\max}$ is the maximum CFL number specified by the user.
2. Numerical experiments have shown that the values $N = 10$, $\mu = 0.5$ and $\nu = 1.25$ perform well.

Box 3 – Automatic Time-increment Control Algorithm.

Given $N, \mu, \nu, \Delta t_{\max}, \gamma_{\min}$ and the current values of $n, \tilde{\delta}, \lambda$ and γ, proceed as follows:

(Tolerance)

If $n = N$, $\tilde{\boldsymbol{\delta}} \leftarrow \tilde{\mathbf{v}}(t_{N-1}) - \tilde{\mathbf{v}}(t_{N-2})$

If $\lambda \neq 1$, $\tilde{\boldsymbol{\delta}} \leftarrow \mu \tilde{\boldsymbol{\delta}}$

(Correction Factor)

If $\lambda \neq 1$ $\gamma \leftarrow \mu \gamma$

If $n > N$ and $\lambda = 1$,

$\Delta \tilde{\mathbf{v}} \leftarrow \tilde{\mathbf{v}}(t_{n-1}) - \tilde{\mathbf{v}}(t_{n-2})$

Compute $\|\tilde{\boldsymbol{\delta}}\|_{\tilde{\mathbf{A}}_0}$ and $\|\Delta \tilde{\mathbf{v}}\|_{\tilde{\mathbf{A}}_0}$

$$\gamma \leftarrow \gamma \min \left(\frac{\|\tilde{\boldsymbol{\delta}}\|_{\tilde{\mathbf{A}}_0}}{\|\Delta \tilde{\mathbf{v}}\|_{\tilde{\mathbf{A}}_0}}, \nu \right)$$

$\gamma \leftarrow \min(\gamma, 1)$

$\gamma \leftarrow \max(\gamma_{\min}, \gamma)$

(Time Increment)

$\Delta t \leftarrow \gamma \Delta t_{\max}$

Return

6. Numerical Results

In this section, we present three numerical experiments to illustrate the proposed procedures.

A local time-increment strategy was used for all problems. Only one pass was performed in the Newton loop, i.e., $i_{\max} = 1$ in Box 1. The tolerance for the GMRES algorithm (ε_{tol} as defined in [9]) was set to 0.1. All computations were done on a CONVEX-C1 in double precision (64 bits per floating point word), using 3-point Gaussian quadrature for linear triangular elements.

6.1 - *NACA-0012 Airfoil*

A NACA-0012 airfoil is placed in a Mach 0.85 two-dimensional inviscid flow at 1 degree of angle of attack. The computational domain and the boundary conditions are depicted in Figure 1, where ρ is the density; u_1 and u_2

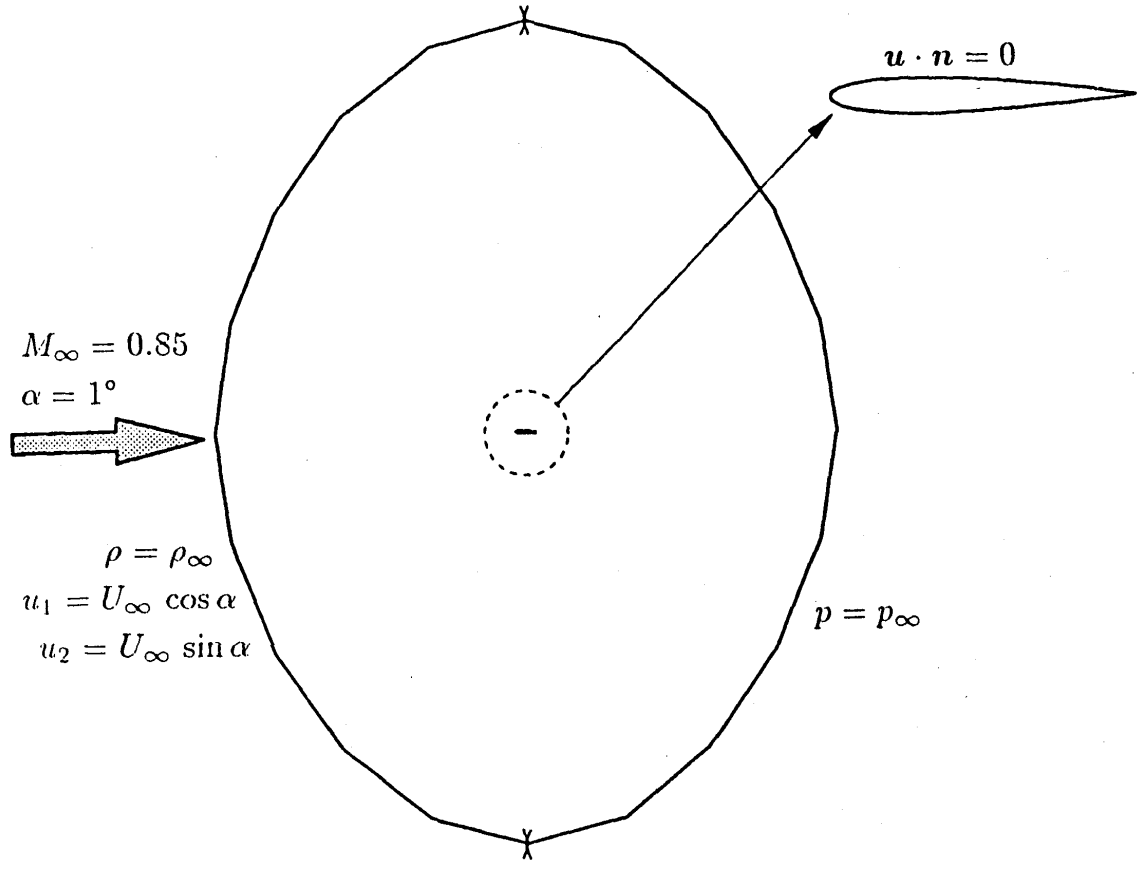

Fig. 1 – NACA-0012 airfoil. Problem description.

are the components of the velocity vector $\boldsymbol{u}$ (not to be confused with the $\boldsymbol{u}$ defined in the previous sections); p is the pressure, and $\boldsymbol{n}$ is the vector normal to the boundary. The finite element mesh consists of 1,393 nodes and 2,602 linear triangles (see Figure 2). The pressure contours near the airfoil are shown in Figure 3. One can note two low pressure regions followed by normal shocks, as is classically seen in transonic flow calculations.

This problem is converged to steady-state starting from a uniform flow. A constant CFL number of 15 is used. The maximum dimension of the Krylov space for the GMRES algorithm is set to 20. Both linear and matrix-free GMRES algorithms were used to solve this test case. For the matrix-free GMRES algorithm, the frozen-coefficient Jacobian was used for the first 100 steps. Then, it was replaced by the consistent Jacobian.

The convergence for both methods as a function of time step is shown in Figure 4. The labels "MF-GMRES" and "Lin-GMRES" refer to the matrix-free and linear GMRES algorithms. The convergence acceleration using the matrix-free GMRES algorithm with the consistent Jacobian is particularly striking. A thorough study of the convergence has revealed that the non-convergence of the linear GMRES algorithm is due to the nonlinearities of the outflow boundary condition. An important remark is that matrix-free GMRES is 2 to 3 times more expensive that linear GMRES to march the same number of time steps. This is observed to be a general rule for two-dimensional computations. The explanation can be found by analyzing the CPU-time cost of each technique as a function of the number of GMRES iterations. For the linear GMRES algorithm, there is an initial overhead associated with the formation of the frozen-coefficient Jacobian matrix. The cost of computing the matrix-vector products is however low ($\sim$ 0.3 times the cost of evaluating the right-hand-side vector). On the other hand, the initial overhead of the matrix-free GMRES algorithm is lower but the cost of a matrix-vector product is equivalent to one evaluation of the right-hand-side vector. The break-even point for these two techniques is about three GMRES iterations. For both techniques the number of GMRES iterations per time step for a typical high speed flow calculation is $\sim 10 - 15$.

Another improvement generated through the use of the matrix-free GMRES algorithm is the reduction in storage as can be seen in Table 1. The storage requirements for explicit and column height direct solvers are given for reference, as well as the storage requested by all the modules of the program. The ratio of memory usage between linear and matrix-free GMRES algorithms is about 3 for this two-dimensional computation. We have calculated this ratio to be of the order of 10 for three-dimensional problems with linear tetrahedra.

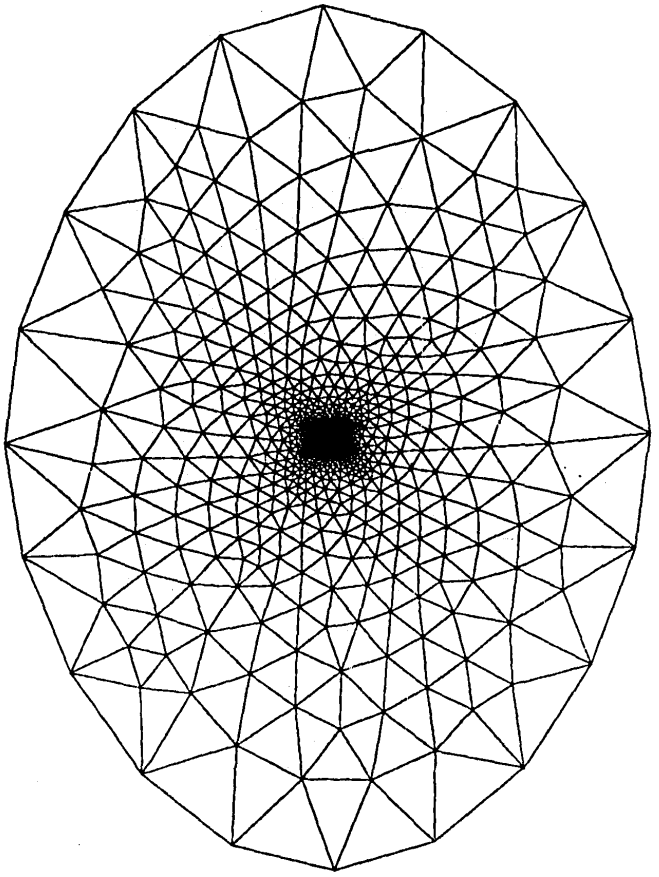

Fig. 2 – NACA-0012 airfoil. Finite element mesh (1,393 nodes; 2,602 elements).

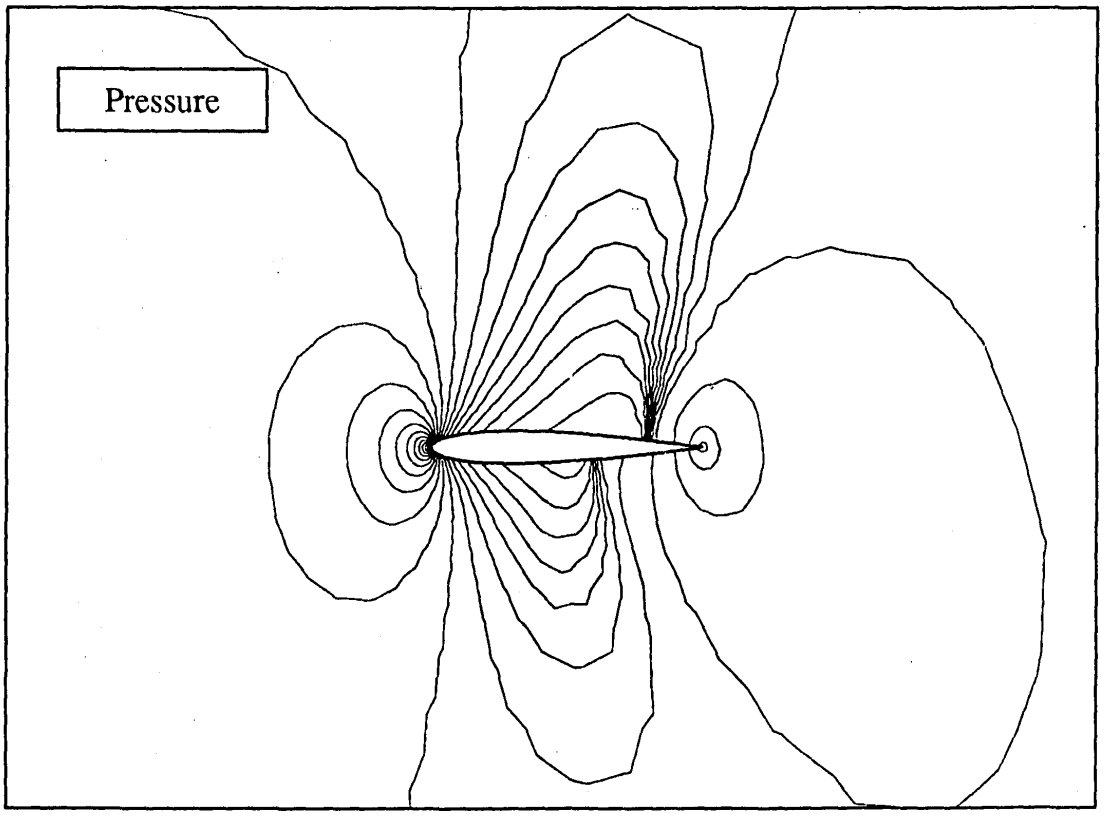

Fig. 3 – NACA-0012 airfoil. Pressure contours.

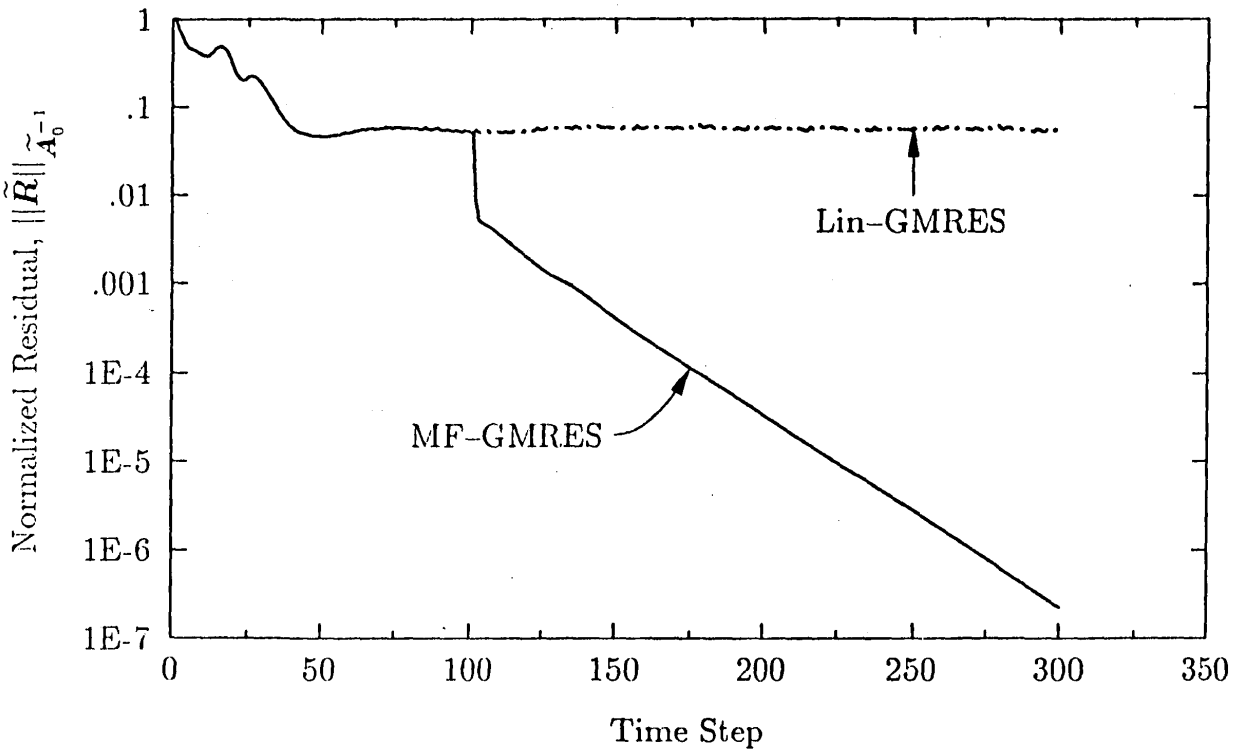

Fig. 4 – NACA-0012 airfoil. Convergence as a function of time step.

Table 1 - *NACA-0012 airfoil. Storage requirement for the solution algorithms and the program, in MBytes.*

	Explicit	MF-GMRES	Lin-GMRES	Direct
Solution Algorithm	0.0	1.3	4.2	21.5
Total	0.7	2.0	4.9	22.2

6.2 - *Half Cylinder*

This test case consists of a Mach 3 inviscid flow over the front part of a cylinder. See Figure 5 for the prescribed boundary conditions, with T being the temperature. The computation was done on a 520-node mesh (see Figure 6) starting from a uniform flow. The maximum dimension of the Krylov space was 20. The linear GMRES algorithm was coupled with the linesearch and automatic time-increment control strategies in order to achieve a stable convergence. The CFL number was allowed to vary between 0.5 and 5, starting with 5. The CFL number as a function of time step is presented

in Figure 7. Note that the CFL number was reduced to 0.5 at the first time step because of possible instabilities at higher CFL numbers detected by the linesearch algorithm (see mark *a* on Figure 7). After the first 10 time steps were completed, the tolerance δ was computed and the auto-Δt algorithm was activated by the program driver (mark *b*). Unfortunately, δ was too large and the CFL had to be reduced twice before an acceptable tolerance was found (marks *c* and *d*). Then, the CFL number progressively increased to 5 while the bow shock was forming in the flow field.

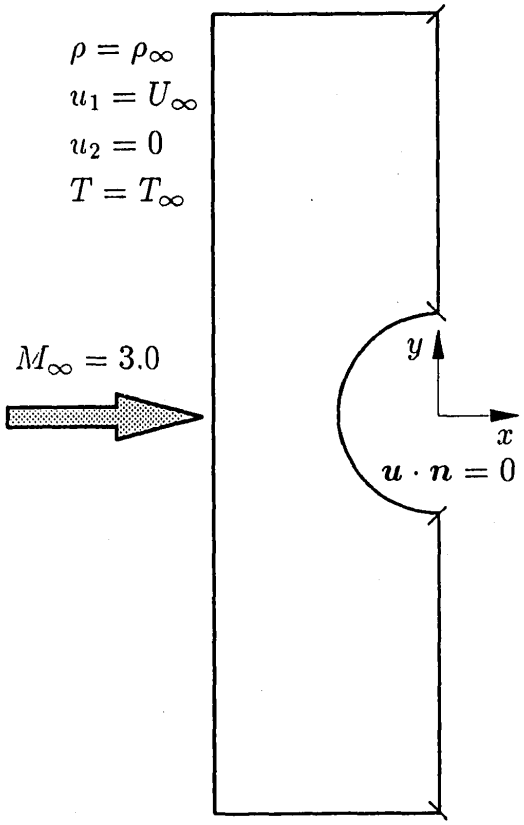

Fig. 5 – Half cylinder.
Problem description.

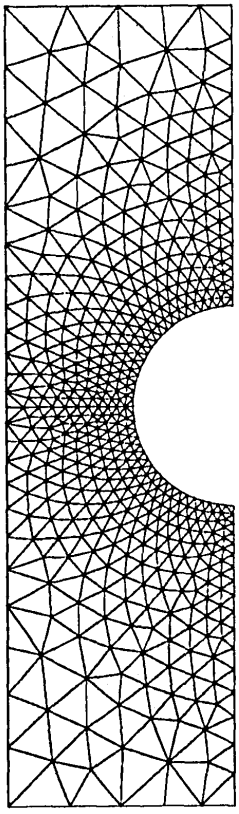

Fig. 6 – Finite elements mesh
(520 nodes; 930 elements).

Convergence difficulties that resulted from the use of the frozen-coefficient Jacobian can be seen in Figure 8. In turn, they generate oscillations on the CFL number in Figure 7 (mark *e*). These difficulties would not exist if the consistent Jacobian was used in place of the frozen-coefficient Jacobian.

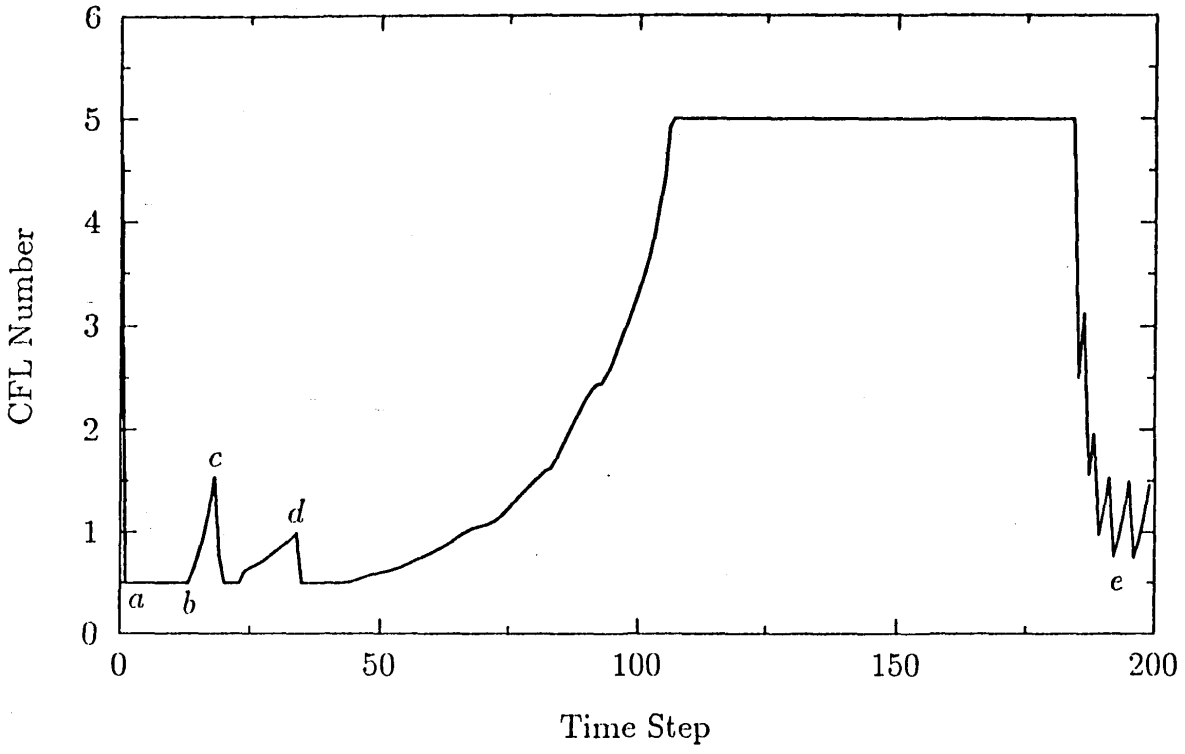

Fig. 7 – Half cylinder. CFL number.

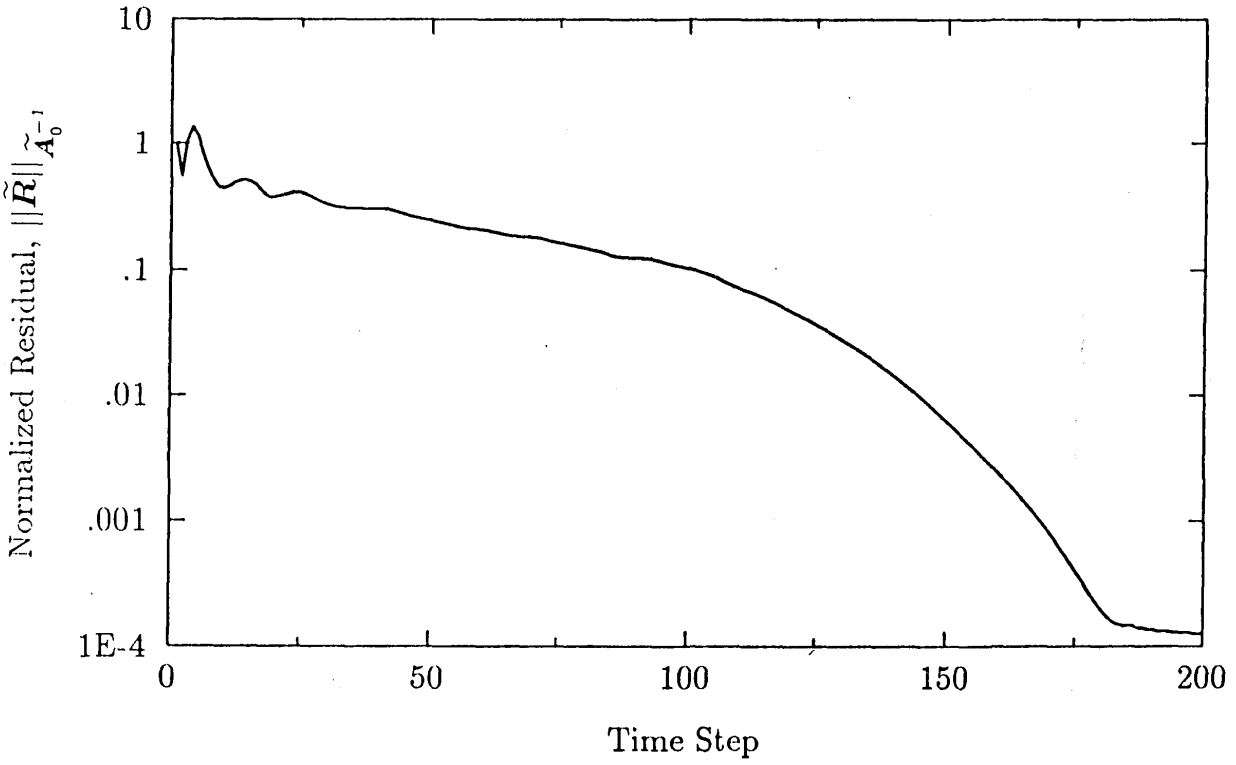

Fig. 8 – Half cylinder. Convergence.

6.3 - *Double Ellipse*

This two-dimensional hypersonic problem consists of a shuttle-like geometry defined by two ellipses placed in a Mach 25 inviscid flow at 30 degrees of angle of attack. The description of this problem is depicted in Figure 9. A first solution was computed on a 624-node mesh (not shown) and was projected onto a 2,350-node mesh (see Figure 10). The temperature contours are

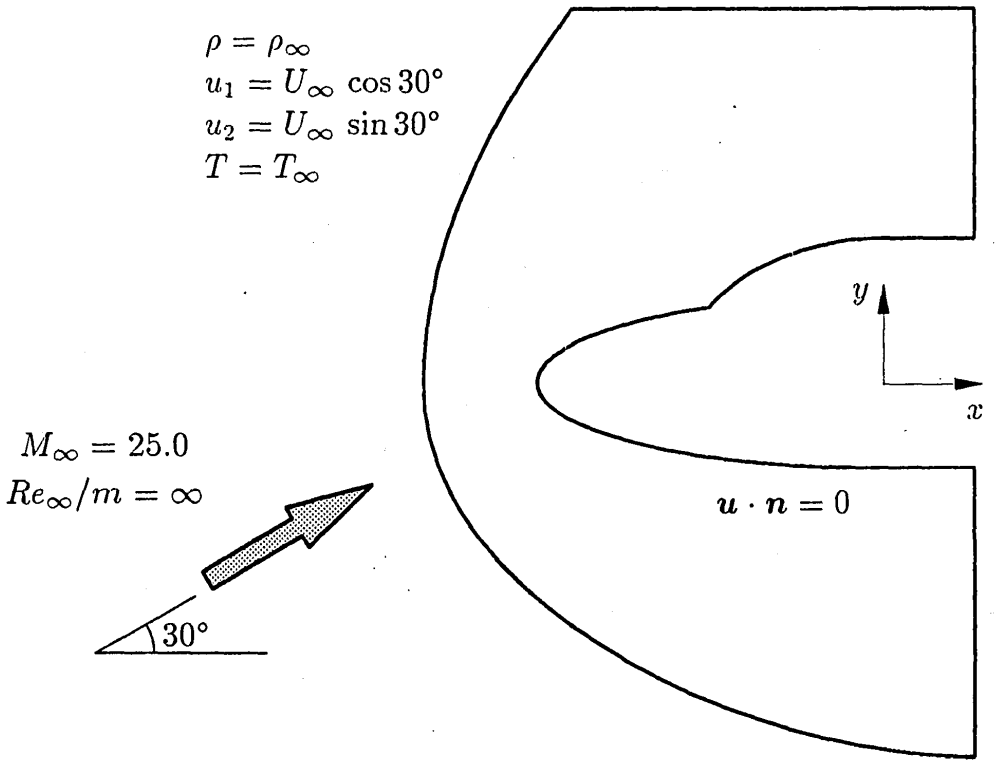

Fig. 9 – Double ellipse. Problem description.

presented in Figure 11. Note that the strong bow shock and the canopy shock could have been better captured with an adaptive mesh procedure. The convergence presented in Figure 12 was obtained by using a CFL number of 2 for the linear GMRES algorithm. Numerical tests have shown that this is the maximum CFL number that could be used with this algorithm. On the other hand, the consistent Jacobian was used for the matrix-free GMRES algorithm and a CFL number of 2 was set for the first 50 steps, followed by a CFL number of 10. This example demonstrates that a better stability, and therefore a faster convergence of the time-marching scheme, can be achieved for high speed flows when the consistent Jacobian is used. This is with the condition that the initial guess is sufficiently close to the steady-state solution. A factor of two

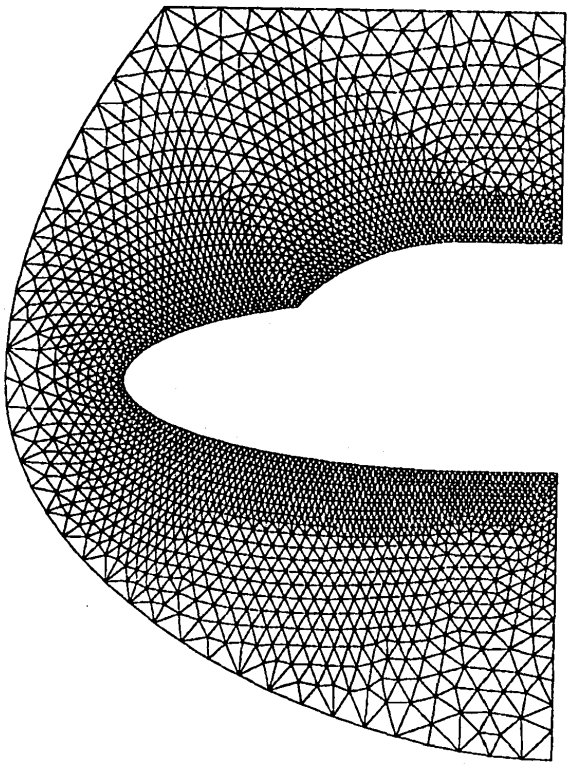

Fig. 10 – Double ellipse. Mesh (2,350 nodes; 4,448 elements).

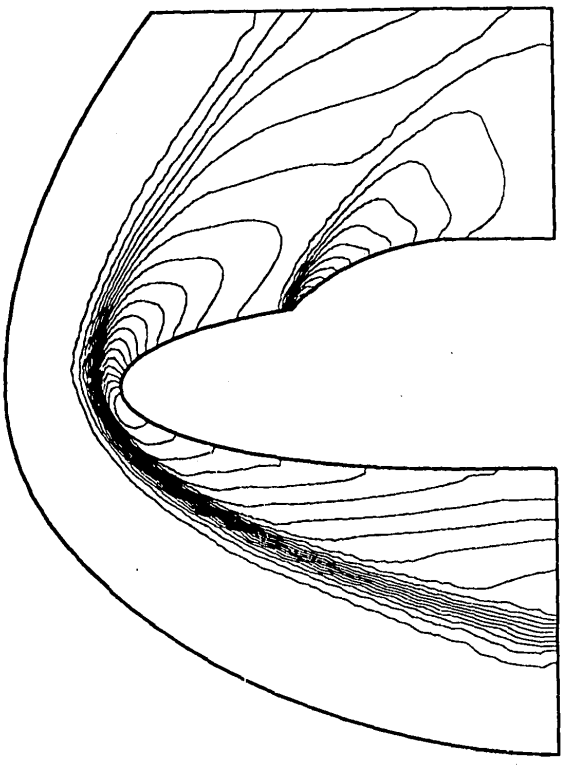

Fig. 11 – Double ellipse. Temperature contours.

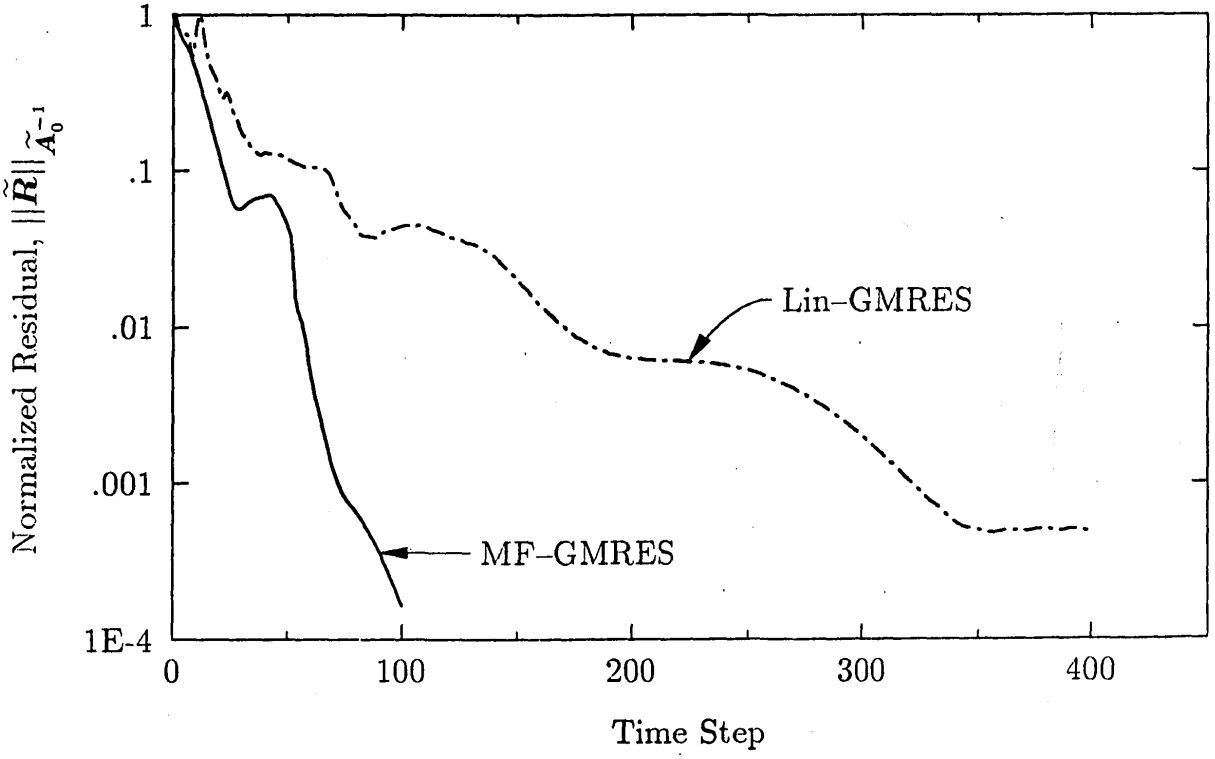

Fig. 12 – Double ellipse. Convergence.

speed-up in CPU-time is achieved with the matrix-free GMRES algorithm at a normalized residual of 10^{-3}.

7. Conclusions

We have presented a solution algorithm for implicit time-marching schemes. Its robustness and efficiency have been enhanced through the use of a linesearch procedure and an automatic time-increment control algorithm. Matrix-free techniques have eliminated the storage of the Jacobian matrix. Two-dimensional computations of inviscid and viscous compressible flows have demonstrated the potential of these strategies. However, three-dimensional industrial problems will provide a better basis for evaluating efficiency. Future work will focus on producing a higher level of automation in the choice of the Jacobian matrix and the different parameters involved in the strategy.

Acknowledgments

The authors would like to express their appreciation to Michel Mallet, Walter Murray and Youcef Saad for helpful comments. This research was supported by the NASA Langley Research Center under Grant NASA-NAG-1-361 and Dassault Aviation, St. Cloud, France.

REFERENCES

[1] P.N. Brown and Y. Saad, "Hybrid Krylov Methods for Nonlinear Systems of Equations", *SIAM Journal of Scientific and Statistical Computing*, **11** (1990) 450-481.

[2] J.E. Dennis and R.B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice-Hall, Englewood Cliffs, NJ, 1983.

[3] K. Eriksson and C. Johnson, "Adaptive Finite Element Methods for Parabolic Problems. I: A Linear Model Problem", Preprint no. 1988: 31/ISSN 0347-2809, Department of Mathematics, Chalmers University of Technology, The University of Göteborg, 1988.

[4] K. Eriksson and C. Johnson, "Error Estimates and Automatic Time-Step Control for Nonlinear Parabolic Problems, I", *SIAM Journal of Numerical Analysis*, **24** (1987) 12-23.

[5] P.E. Gill, W. Murray and M.H. Wright, *Practical Optimization*, Academic Press, London, 1981.

[6] T.J.R. Hughes, *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis*, Prentice-Hall, Englewoods Cliffs, NJ, 1987.

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

[8] F. Shakib, "Finite Element Analysis of the Compressible Euler and Navier-Stokes Equations", *Ph.D. Thesis*, Stanford University, 1989.

[9] F. Shakib, T.J.R. Hughes and Z. Johan, "A Multi-Element Group Preconditioned GMRES Algorithm for Nonsymmetric Systems Arising in Finite Element Analysis", *Computer Methods in Applied Mechanics and Engineering*, **75** (1989) 415-456.

[10] J.M. Winget and T.J.R. Hughes, "Solution Algorithms for Nonlinear Transient Heat Conduction Analysis Employing Element-by-Element Iterative Strategies", *Computer Methods in Applied Mechanics and Engineering*, **52** (1985) 711-815.

Zdeněk JOHAN - Thomas J.R. HUGHES
Division of Applied Mechanics, Stanford University, Stanford, CA 94305, U.S.A.

Farzin SHAKIB
CENTRIC Engineering Systems, Inc., 3801 East Bayshore Road, Palo Alto,
CA 94303, U.S.A.