INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS, VOL. 21, 1171–1200 (1995)

ASSESSMENT OF SOME ITERATIVE METHODS FOR NON-SYMMETRIC LINEAR SYSTEMS ARISING IN COMPUTATIONAL FLUID DYNAMICS

ANTONIO SORIA AND FRANÇOIS RUEL

European Commission, Safety Technology Institute, Joint Research Centre, I-21020 Ispra, Italy

SUMMARY

Various tests have been carried out in order to compare the performances of several methods used to solve the non-symmetric linear systems of equations arising from implicit discretizations of CFD problems, namely the scalar advection–diffusion equation and the compressible Euler equations. The iterative schemes under consideration belong to three families of algorithms: relaxation (Jacobi and Gauss–Seidel), gradient and Newton methods. Two gradient methods have been selected: a Krylov subspace iteration method (GMRES) and a non-symmetric extension of the conjugate gradient method (CGS). Finally, a quasi-Newton method has also been considered (Broyden). The aim of this paper is to provide indications of which appears to be the most adequate method according to the particular circumstances as well as to discuss the implementation aspects of each scheme.

KEY WORDS: implicit schemes; non-symmetric matrices; iterative procedures; relaxation methods; gradient methods; quasi-newton methods; convection–diffusion equation; Euler equation

1. INTRODUCTION

Implicit time discretization of computational fluid dynamics (CFD) problems usually yield non-symmetric systems of equations. A large variety of methods are currently in use to solve this class of problems. The scope of this paper is to compare and evaluate several iterative techniques which have been used in the past to treat problems within this framework. *A priori* discarding any claim of exhaustiveness, the study has been focused on the consideration of classically used families of methods, identifying among each of these families the most conventional algorithm according to the recent literature.

The first group of methods considered comprises the well-known classical *relaxation methods* (Jacobi and Gauss–Seidel). They do not require accurate preconditioning to provide satisfactory convergence and are easy to implement.

The performance of preconditioned conjugate gradient techniques applied to symmetric, positive definite systems has been well assessed. Several extensions of these methods to the non-symmetric case have been proposed during the last few years. The reader is referred to References 1 and 2 for surveys of these methods using various preconditioning schemes (incomplete lower–upper (ILU) decompositions in the former and element-by-element (EBE) techniques in the latter). Following the results published by Joly and Eymard[3] and van der Vorst,[4] we retained the conjugate gradient squared method (CGS) proposed by Sonneveld[5] from this algorithm family. Although it is not clear whether CGS is the best technique of its type (recently proposed algorithms such as van der Vorst's stabilized biconjugate gradient method (Bi-CGSTAB)[6] seem to yield smoother and faster convergence paths), we included CGS in the test set since it is a widely used method for the kind of applications under consideration.

CCC 0271–2091/95/121171–30

Received August 1994
Revised April 1995

Starting with Arnoldi,[7] the Krylov subspace iteration techniques have also been widely used in past years to deal with non-symmetric matrices. They are based upon a minimization process which takes place in a series of Krylov subspaces whose dimension increases as the iteration process advances. These methods are closely linked to the *gradient methods*, both of them being *exact* from the theoretical viewpoint, since it has been proven that all of them converge to the exact solution in a *finite* number of numerical operations. The generalized minimum residual method (GMRES) due to Saad and Schulz[8] appears to be one of the best-performing schemes of this class and has become quite popular in recent years within the CFD community. For an extensive survey of the performance of GMRES with various EBE preconditioning schemes the reader is referred to Reference 9. Further details on other Krylov subspace methods are given in References 5 and 8 and references cited therein.

A third class of iterative methods will be considered based on *Newton schemes*. The underlying idea consists of treating the linear system to solve as a non-linear vector problem and then using the common techniques for non-linear equations for this particular problem, where the system matrix plays the role of the Jacobian of the non-linear equation. As outlined in Reference 10, the Broyden method[11] is a quasi-Newton method particularly suited to dealing with non-symmetric matrices that arise in CFD problems, so it has also been retained for comparison with the other schemes.

The interest in this comparison exercise was raised during some discussions held with Messrs. Guillard and Nkonga from the French Institut National de Recherche en Informatique et Automatique (INRIA), with whom a fruitful, informal exchange of information in this field has been set-up. In this frame, some of the test problems were agreed with them.

Concerning the model problems, we have considered a single scalar equation (the advection–diffusion equation) discretized in time by a fully implicit scheme and a system of equations (the compressible Euler equations) using a semi-implicit approach. In both cases 2D problems have been considered.

The paper is organized as follows. The basic tested algorithms are presented in Section 2. A brief outline of the preconditioning techniques used is given in Section 2.1. The relaxation methods are recalled in Section 2.2. The gradient methods GMRES and CGS are described in Sections 2.3 and 2.4. Finally, the Broyden method is presented in Section 2.5. The considered numerical tests are described, with an outline of the discretization numerical scheme used for each model problem, in Section 3. Section 3.1 is devoted to the tests concerning the advection–diffusion equation, while Section 3.2 deals with the compressible Euler equations. To end with, the conclusions of the study are summarized in Section 4.

2. NUMERICAL METHODS

The problem under consideration is the solution of a linear system of equations

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

coming from the implicit (or semi-implicit) time discretization of a hyperbolic partial differential equation or a system of partial differential equations. The matrix A depends on the spatial discretization selected as well as on the time integration scheme. Hyperbolic equations (or systems) typically lead to non-symmetric matrices. Several numerical methods to solve linear systems of this kind will be presented in this section and then compared in the following one.

2.1. Preconditioning techniques

An efficient way to facilitate the solution of problem (1) consists of replacing the original equation by

$$B^{-1}Ax = B^{-1}b. \tag{2}$$

This procedure aims at improving the conditioning of the resulting matrix $B^{-1}A$, i.e. to concentrate the eigenvalues of the resulting matrix around unity. The matrix B^{-1} is usually referred to as the *preconditioning matrix*. This modification of the spectral characteristics of the matrix helps in the numerical solution of the system (1).

Needless to say, the better the matrix B approximates A, the more efficient is the preconditioning. However, the preconditioning process should require a significantly lower computational cost when compared with the direct solution of (1) in order to justify the procedure. The simplest choice consists of taking the *diagonal preconditioning*

$$B_{\text{diag}}^{-1} = \text{diag}\{a_{11}^{-1}, a_{22}^{-1}, \ldots, a_{nn}^{-1}\} \tag{3}$$

for scalar problems and the corresponding *block-diagonal preconditioning*

$$B_{\text{diag}}^{-1} = \text{diag}\{A_{11}^{-1}, A_{22}^{-1}, \ldots, A_{nn}^{-1}\} \tag{4}$$

for vectorial problems. Another problem choice for the preconditioning matrix is the so-called *incomplete lower–upper* (ILU) decomposition, which is the non-symmetric counterpart of the *incomplete Choleski* decomposition.[12]

The *lower–upper factorization* of a matrix is the most common way to numerically solve linear systems by direct methods. It is based on a result that states that any non-singular matrix A admits a *unique* factorization $A = LU$, where L is lower triangular *with unit main diagonal* and U is upper triangular. Once the factorization has been obtained, the solution is computed by successive back-substitutions

$$y = L^{-1}b, \qquad x = U^{-1}y. \tag{5}$$

This process of factorization can be seen as a kind of Gauss elimination on the matrix A. The LU decomposition is the most expensive part of the procedure from the computational point of view and becomes inpracticable for systems of moderately large size.

The basic idea of the ILU method is to perform the factorization rejecting all entries in the product LU except for those belonging to a *given sparsity pattern P*.

If we define the sparsity pattern P as the set of double subscripts (i, j) for which $a_{ij} \neq 0$, the basic ILU factors verify

$$u_{ij}^{ILU} \begin{cases} \neq 0 & \text{if } (i,j) \in P \text{ and } i < j, \\ = 0 & \text{otherwise,} \end{cases} \qquad l_{ij}^{ILU} = \begin{cases} \neq 0 & \text{if } (i,j) \in P \text{ and } i > j, \\ = 0 & \text{otherwise.} \end{cases} \tag{6}$$

In particular, we note that given a matrix A which has the same fill-in pattern of its triangular factors (e.g. banded matrices), the corresponding ILU and LU decompositions are equal.

There are other ways to select the sparsity pattern P. For instance, one could select only those entries where the absolute value of the system matrix entry a_{ij} exceeds a prescribed value or one could choose a banded sparsity pattern. However, the aim of this study is to compare iterative strategies rather than preconditioning techniques, so only the original ILU method will be retained.

Both full LU and ILU factorizations can be performed blockwise for problems coming from the discretization of systems of partial differential equations. Division by a scalar entry in the pointwise version is then just replaced by LU back-substitution of the corresponding block, whose factorization has to be kept in memory.

The algorithms used to carry out the ILU decomposition as well as the ILU back-substitution depend on the stockage strategy adopted for the matrix, which turns out to be also dependent on the expected size of the system. For very large systems the in-core storage of the matrix is no longer possible and one has simply to recompute each entry a_{ij} when needed. For a detailed description of a class of such algorithms the reader is referred to Reference 1, where a study on the performances of several incomplete decompositions according to different sparsity patterns has also been made.

2.2. Standard relaxation methods: Jacobi and Gauss–Seidel

The Jacobi and Gauss–Seidel relaxation methods are based on so-called *additive splittings* of the system matrix A, i.e.

$$A = S - P, \tag{7}$$

where S is a matrix whose inverse is easy to obtain. The original system (1) is then equivalent to

$$Sx = Px + b, \tag{8}$$

which leads to the iterative strategy

$$x^{k+1} = S^{-1}Px^k + S^{-1}b. \tag{9}$$

The resulting iterative method will converge to the solution independently of the initial guess x_0 if the spectral radius ρ of the matrix $S^{-1}P$ verifies $\rho(S^{-1}P) < 1$ (see e.g. Reference 13 for a formal proof).

The *Jacobi method* consists of taking

$$S = \text{diag}(A) = D, \tag{10}$$

which implies a matrix P given by

$$p_{ij} = \begin{cases} -a_{ij} & \text{if } i \neq j, \\ 0 & \text{if } i = j. \end{cases} \tag{11}$$

With this splitting, equation (9) yields the iterative procedure for the Jacobi method, which, denoting the iteration counter by a superscript and the vector component as a subscript, reads

$$\begin{aligned} a_{11}x_1^{k+1} &= -a_{12}x_2^k - a_{13}x_3^k \cdots - a_{1n}x_n^k + b_1, \\ a_{22}x_2^{k+1} &= -a_{21}x_1^k - a_{23}x_3^k \cdots - a_{2n}x_n^k + b_2, \\ &\vdots \\ a_{nn}x_n^{k+1} &= -a_{n1}x_1^k - a_{n2}x_2^k - a_{n3}x_2^k \cdots + b_n. \end{aligned} \tag{12}$$

The *Gauss–Seidel* method comes from another selection of the matrices S and P. Defining the triangular matrices U (upper) and L (lower) as

$$u_{ij} = \begin{cases} a_{ij}, & \text{if } i < j, \\ 0, & \text{otherwise,} \end{cases} \qquad l_{ij} = \begin{cases} a_{ij}, & \text{if } i > j, \\ 0, & \text{otherwise,} \end{cases} \tag{13}$$

one can take $S = D + L$ and $P = -U$, thus yielding the iteration scheme

$$\begin{aligned} a_{11}x_1^{k+1} &= -a_{12}x_2^k - a_{13}x_3^k \cdots - a_{1n}x_n^k + b_n, \\ a_{22}x_2^{k+1} &= -a_{21}x_1^{k+1} - a_{23}x_3^k \cdots - a_{2n}x_n^k + b_2, \\ &\vdots \\ a_{nn}x_n^{k+1} &= -a_{n1}x_1^{k+1} - a_{n2}x_2^{k+1} - a_{n3}x_3^{k+1} \cdots + b_n. \end{aligned} \tag{14}$$

Equations (12) and (14) express the pointwise Jacobi and Gauss–Seidel methods respectively. It can be shown[13,14] that the convergence of one of them ensures the convergence of the other, since the spectral radii of the associated matrices verify the relation

$$\rho([S^{-1}P]_{G-S}) = (\rho([S^{-1}P]_J))^2, \tag{15}$$

which indicates also that the convergence rate of the Gauss–Seidel method is always better than that of

the Jacobi method. The field of application of the Jacobi method will thus be limited to those applications in which the update of the vector components is not made sequentially (i.e. parallel computations).

By decomposing the solution vector x and the right-hand-side b into subvectors of length n_b, and correspondingly the matrix A into squared blocks of arbitrary size n_b, blockwise versions of these algorithms are easily derived. The matrix D is now block-diagonal, i.e.

$$D = \text{diag}\{A_{11}, A_{22}, \ldots, A_{nn}\}, \tag{16}$$

and the block-Jacobi and block-Guass–Seidel schemes read respectively

$$\begin{aligned}
X_1^{k+1} &= A_{11}^{-1}(\quad -A_{12}x_2^k - A_{13}x_3^k \cdots - A_{1n}x_n^k + B_1),\\
X_2^{k+1} &= A_{22}^{-1}(-A_{21}x_1^k \quad - A_{23}x_3^k \cdots - A_{2n}x_n^k + B_2),\\
&\vdots\\
X_n^{k+1} &= A_{nn}^{-1}(-A_{n1}x_1^k - A_{n2}x_2^k - A_{n3}x_3^k \cdots \quad + B_n),
\end{aligned} \tag{17}$$

$$\begin{aligned}
X_1^{k+1} &= A_{11}^{-1}(\quad -A_{12}x_2^k \quad - A_{13}x_3^k \quad \cdots - A_{1n}x_n^k + B_1),\\
X_2^{k+1} &= A_{11}^{-1}(-A_{21}x_1^{k+1} \quad - A_{23}x_3^k \quad \cdots - A_{2n}x_n^k + B_2),\\
&\vdots\\
X_n^{k+1} &= A_{11}^{-1}(-A_{n1}x_1^{k+1} - A_{n2}x_2^{k+1} - A_{n3}x_3^{k+1} \cdots \quad + B_n),
\end{aligned} \tag{18}$$

both of them requiring the storage of the factored diagonal blocks, which can be viewed as a block-diagonal preconditioning technique whose quality increases with the block size n_b.

2.3. *Gradient methods for non-symmetric systems: GMRES*

The generalized minimum residual method (GMRES) is at present one of the most popular iterative methods used for non-symmetric ssytems and belongs to the family of gradient methods. It is based on the *Krylov subspace iteration method*, where the solution is found, starting from an *initial guess* x_0, as follows. The original (N-dimensional) system (1) is replaced by

$$Az = r_0, \tag{19}$$

where $z = x - x_0$ and $r_0 = b - Ax_0$ is the initial residual. The mth approximation to the solution, z_m, is then searched in the Krylov subspace $K^m = \text{span}\{r_0, Ar_0, A^2r_0, A^3r_0, \ldots, A^mr_0\}$.

The original Krylov subspace iteration method of Arnoldi[7] was improved by Saad and Schulz,[8] who formulated the GMRES method to be presented here. The general update formula is

$$x_m = x_0 + z_m, \tag{20}$$

where z_m belongs to the Krylov subspace K^m. In other words,

$$x_m = x_0 + \sum_{k=1}^{m} y_k^m s_k, \tag{21}$$

where the scalars y_k are the components of the correction z_m in the base formed by the vectors s_k, which are constructed as the iterative process advances. To that approximation corresponds a residual $r_m = b - Ax_m$. The value of y_k is chosen in such a way as to minimize the norm $\| b - Ax \|$. Let us assume that at the mth iteration the vectors s_k, $k = 1, \ldots, m$, are known. Let us define, for each

iteration counter m, the $(m+1) \times m$ Hessenburg matrix

$$H_m = \begin{bmatrix} s_1^T A s_1 & s_1^T A s_2 & \cdots & s_1^T A s_m \\ \|A s_1\| & s_2^T A s_2 & \cdots & s_2^T A s_m \\ 0 & \|A s_2\| & \cdots & \cdots \\ \cdots & \cdots & \cdots & s_m^T A s_m \\ 0 & 0 & 0 & \|A s_m\| \end{bmatrix}. \tag{22}$$

We will also denote by S_m the $N \times m$ matrix made up of the m-vector base of the Krylov subspace s_k, $k=1, \ldots, m$, which is orthonormal by construction. The following important property is then verified:

$$AS_m = S_{m+1} H_m. \tag{23}$$

Let us represent by $e^m = \{ \| r_0 \|, 0, , \ldots, 0\}$ the m-vector whose first entry is the Euclidean norm of the initial residual and whose last $m-1$ entries are zero. Since s_0 and r_0 are collinear, it follows that $r_0 = S_{m+1} e^{m+1}$.

The evaluation (by minimization) of the global residual norm in the N-dimensional space is made by means of the following property, which allows for a dimension reduction in the space where the norm is calculated:

$$\|r_m\|_N = \left\| r_0 - A \sum_{k=1}^{m} y_k^m s_k \right\|_N = \|r_0 - AS_m y\|_N = \|S_{m+1}(e^{m+1} - H_m y)\|_N = \|e^{m+1} - H_m y\|_{m+1}, \tag{24}$$

where the subscripts to the norms indicate the dimension of the space in which they are calculated. Using this property, the minimization problem takes place for iteration m in a subspace of dimension $m+1$:

$$\min \|b - A(x_0 + z)\|_N = \min \|e^{m+1} - H_m y\|_{m+1}. \tag{25}$$

At the beginning of an iteration an orthonormal base of the $(m+1)$-dimensional Krylov subspace K^m has to be found. At this point we have a set of m orthonormal vectors s_k and the additional vector $A^m r_0$, which is not orthonormal to the subspace generated by the others. A modified Gram–Schmidt orthonormalization process is then required to obtain s_{m+1}.

Modified Gram–Schmidt orthonormalization algorithm

First vector of the base: $s_1 \leftarrow \| r_0 \|^{-1} r_0$
For $m = 1, \ldots, m_{\max}$ repeat:
  $s_{m+1} \leftarrow A s_m$
  For $j = 1, \ldots, m$ repeat:
    Calculate and store $s_j^T A s_m$, i.e. the mth column of H_m
    Update: $s_{m+1} \leftarrow s_{m+1} - (s_j^T A s_m) s_j$
  Normalize: $s_{m+1} \leftarrow \| s_{m+1} \|^{-1} s_{m+1}$
  Next j

At this point the base of the $(m+1)$-dimensional Krylov subspace is known, i.e. the matrix S_m, as well as the Hessenberg matrix H_m. The problem now is to find the co-ordinates y_k^{m+1} that minimize the expression given in (25). This can be done taking advantage of the quasi-triangular structure of the matrix H_m, which allows for an easy conversion to a triangular one. A series of orthogonal Givens rotations R are applied to the matrix in such a way that the resulting $(k+1) \times k$ matrix $\tilde{H}_m = R H_m$ has a

zero last row:

$$\|e^{m+1} - H_m y\|_{m+1} = \|Re^{m+1} - RH_m y\|_{m+1} = \|\tilde{e}^{m+1} - \tilde{H}_m y\|_{m+1}. \tag{26}$$

If the first m components of the vector are forced to be zero, we have the triangular system

$$\begin{bmatrix} \tilde{H}_{11} & \tilde{H}_{12} & \cdots & \tilde{H}_{1m} \\ \tilde{H}_{21} & \tilde{H}_{22} & \cdots & \tilde{H}_{2m} \\ \cdots & \cdots & \cdots & \cdots \\ \tilde{H}_{m1} & \tilde{H}_{m2} & \cdots & \tilde{H}_{mm} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \cdots \\ y_m \end{bmatrix} = \begin{bmatrix} \tilde{e}_1 \\ \tilde{e}_2 \\ \cdots \\ \tilde{e}_m \end{bmatrix} \tag{27}$$

and it necessarily follows that

$$\min\|e^{m+1} - H_m y\|_{m+1} = |\tilde{e}_{m+1}|. \tag{28}$$

The construction of the matrix RH_m can be viewed as a QR factorization in which the matrix R is the composition of m plane Givens rotations, i.e.

$$R = R_m R_{m-1} R_{m-2} \ldots R_1, \tag{29}$$

where each Givens rotation is defined as

$$R_j = \begin{bmatrix} I_{j-1} & \cdots & 0 \\ \cdots & \begin{bmatrix} c_j & -s_j \\ -s_j & c_j \end{bmatrix} & \cdots \\ 0 & \cdots & I_{m-j} \end{bmatrix}. \tag{30}$$

In addition, owing to (25), the exact value of the residual $r_m = b - Ax_m$ is obtained by (28) even without the explicit calculation of the corrected solution x_m. This permits us to construct the solution by means of (21) only when convergence has been reached.

For the sake of completeness the full GMRES algorithm (with left preconditioning matrix L) is given below.

GMRES algorithm

Compute $r_0 \leftarrow L^{-1}(b - Ax_0)$
$e_1 \leftarrow \| r_0 \|$
First vector of the base: $s_1 \leftarrow \| r_0 \|^{-1} r_0$
For $m = 1, \ldots, m_{\max}$ repeat:
  $s_{m+1} \leftarrow L^{-1} A s_m$
Modified Gram–Schmidt algorithm:
  For $j = 1, \ldots, m$ repeat:
    Calculate and store $s_j^{\mathrm{T}} L^{-1} A s_m$, i.e. the mth column of H_m
    Update: $s_{m+1} \leftarrow s_{m+1} - (s_j^{\mathrm{T}} L^{-1} A s_m) s_j$
  Next j
  Normalize: $s_{m+1} \leftarrow \| s_{m+1} \|^{-1} s_{m+1}$
  End of modified Gram–Schmidt algorithm
  Complete the subdiagonal term $H_{m+1,m} \leftarrow \| s_{m+1} \|^{-1}$
  QR algorithm:
  For $j = 1, \ldots, m - 1$ repeat:
    Elementary Givens rotations: $h_1 \leftarrow H_{j,m+1}$, $h_2 \leftarrow H_{j+1,m+1}$
    Elementary Givens rotations: $H_{j,m+1} \leftarrow c_j h_1 + s_j h_2$, $H_{j+1,m+1} \leftarrow -s_j h_1 + c_j h_2$
  Next j

Compute s_m and c_m: $r \leftarrow \sqrt{(H^2_{m,m+1} + H^2_{m+1,m+1})}$
$c_m \leftarrow r^{-1} H_{m,m+1}$, $s_m \leftarrow r^{-1} H_{m+1,m+1}$
$H_{m,m+1} \leftarrow r$, $H_{m+1,m+1} \leftarrow 0$
Update e_m and e_{m+1}: $e_{m+1} \leftarrow -s_m e_m$, $e_m \leftarrow c_m e_m$
End of QR algorithm
Convergence test: if $|e_{m+1}| < \varepsilon$, exit m-loop
Solve the $m \times m$ system $Hy = e$
Update the solution: $x \leftarrow x_0 \sum_{k=1}^{m} y_k^m s_k$

It is evident from (21) that the selection of an appropriate initial guess x_0 is extremely important for an efficient performance of the algorithm. In addition, the size of the system of equations to be solved at each iteration level increases with the iteration counter. In practice, when difficulties in convergence are detected and/or when the computational cost of solving $Hy = e$ begins to be significant, it is advisable to stop the cycle, form a new tentative solution and restart the procedure from this point on. Typical values for maximum inner iterations are in the range 10–50, whereas the outer iterations are restarted until convergence is reached.

2.4. Gradient methods for non-symmetric systems: CGS

The conjugate gradient method is, when combined with an adequate preconditioning, one of the most powerful iterative methods for solving symmetric, positive definite systems of equations. The starting point of this method consists of associating the solution of the linear system

$$Ax = b \tag{31}$$

with the minimization of the function $\| Ax - b \|^2$. The idea is to produce a series of corrections s_i, mutually A-conjugate, that produce a series of approximate solutions $x_{i+1} = x_i + s_i$ in such a way that the residual vectors $r_i = b - Ax_i$ are mutually orthogonal. The hypothesis of A being positive definite is basic to ensure that it possesses an associated norm. Under these conditions it can be proved (see e.g. Reference 15) that the method converges to the exact solution in at most N iterations, N being the system size. The overall algorithm is summarized as follows, assuming a left preconditioning matrix B.

Conjugate gradient algorithm

First residual vector: $r_0 = b - Ax_0$ and first correction direction $p_0 = s_0 = B^{-1} r_0$
For $m = 0, \ldots, m_{\max}$ repeat:
Compute $\alpha_m \leftarrow r_m^{\mathrm{T}} s_m / p_m^{\mathrm{T}} A p_m$
Update $x_{m+1} \leftarrow x_m + \alpha_m p_m$
Update $r_{m+1} \leftarrow r_m - \alpha_m A p_m$
Convergence test: if $\| r_{m+1} \| < \varepsilon$, stop the calculation
Update $s_{m+1} \leftarrow B^{-1} r_{m+1}$
Compute $\beta_{m+1} \leftarrow r_{m+1}^{\mathrm{T}} s_{m+1} / r_m^{\mathrm{T}} s_m$
Update $p_{m+1} \leftarrow s_{m+1} + \beta_{m+1} p_m$
Next m

When dealing with *non-symmetric* systems of equations, there is no norm associated with the matrix and the optimal properties of the algorithm are lost. Several attempts to remedy this inconvenience have been proposed. For instance, premultiplication by A^{T} in (31) would yield the symmetric system $A^{\mathrm{T}} A x = A^{\mathrm{T}} b$, to which the original scheme could be applied. This choice constitutes the so-called *normal equation method*. Another possibility, usually referred to as the *biconjugate gradient method*,[16]

consists of applying the conjugate gradient method to the system

$$\begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} b \\ b \end{bmatrix}. \tag{32}$$

The iterative process is carried out in this case by means of two series of vectors corresponding to the two subsystems associated with the scheme. The (unpreconditioned) iterative algorithm is constructed according to the following rule.

Biconjugate gradient algorithm

First residual vector: $r_0 = b - Ax_0$ and first correction direction $p_0 = s_0 = q_0 = r_0$
For $m = 0, \ldots, m_{\max}$ repeat:
  Compute $\alpha_m \leftarrow r_m^{\mathrm{T}} s_m / q_m^{\mathrm{T}} A p_m$
  Update $x_{m+1} \leftarrow x_m + \alpha_m p_m$
  Update $s_{m+1} \leftarrow s_m - \alpha_m A^{\mathrm{T}} q_m$
  Update $r_{m+1} \leftarrow r_m - \alpha_m A p_m$
  Convergence test: if $\| r_{m+1} \| < \varepsilon$, stop the calculation
  Compute $\beta_m \leftarrow r_{m+1}^{\mathrm{T}} s_{m+1} / r_m^{\mathrm{T}} s_m$
  Update $p_{m+1} \leftarrow r_{m+1} + \beta_m p_m$
  Update $q_{m+1} \leftarrow s_{m+1} + \beta_m q_m$
Next m

The above algorithm can be viewed as a form to obtain the vectors r_m, s_m, p_m and q_m in a recursive manner, expressed as polynomials of degree m, of the matrix A applied to the initial residual vector r_0. The polynomial schemes are

$$r_m = \Phi_m[A] r_0, \qquad s_m = \Phi_m[A^{\mathrm{T}}] r_0, \qquad p_m = \Psi_m[A] r_0, \qquad r_m = \Psi_m[A^{\mathrm{T}}] r_0, \tag{33}$$

where the polynomials Φ_m and Ψ_m are defined recursively according to

$$\Phi_{m+1}[A] = \Phi_m[A] - \alpha_m A \Psi_m[A], \qquad \Psi_{m+1}[A] = \Psi_m[A] + \beta_m A \Phi_m[A]. \tag{34}$$

The basic idea of the *conjugate gradient squared* method due to Sonneveld[5] consists of squaring the biconjugate gradient recursive relations given by (34). By doing so, the residual after m iterations is contracted to $\Phi_m^2[A] r_0$ rather than $\Phi_m[A] r_0$, thus providing a better convergence behaviour.

The resulting CGS algorithm is summarized in the following, assuming a left preconditioning matrix B^{-1}.

Conjugate gradient squared algorithm

First residual vector: $r_0 = B^{-1}(b - Ax_0)$ and initialization $v_0 = q_0 = \mathbf{0}$ and $\beta_0 = 1$
For $m = 1, \ldots, m_{\max}$ repeat:
  Compute $\beta_m \leftarrow r_{m-1}^{\mathrm{T}} r_0 / \beta_{m-1}$
  Update $p_m \leftarrow r_{m-1} + \beta_m v_{m+1}$
  Update $q_m \leftarrow p_m + \beta_m (v_{m-1} + \beta_m q_{m-1})$
  Compute $\alpha_m \leftarrow r_{m-1}^{\mathrm{T}} r_0 / p_n^{\mathrm{T}} r_0$
  Update $v_m \leftarrow p_m - \alpha_m B^{-1} A p_m$
  Update $x_m \leftarrow x_{m-1} + \alpha_m (p_m + v_m)$
  Update $r_m \leftarrow r_m - B^{-1} A (p_m + v_m)$
  Convergence test: if $\| r_m \| < \varepsilon$, stop the calculation
Next m

When comparing the biconjugated gradient algorithm with the CGS algorithm, one remarks that the matrix–vector multiplications in the former are of the type Ar as well as $A^{\mathrm{T}}r$, whereas the latter contains only products of the form Ar. This is an interesting advantage of the CGS method with respect to the biconjugated gradient method, because the practical implementation of products $A^{\mathrm{T}}r$ is usually complicated and requires an *ad hoc* subprogramme which depends on the matrix storage scheme, which normally has been devised to perform operations of the type Ar efficiently. Note also that both methods in their preconditioned versions require two back-substitutions of the type $B^{-1}r$ *per iteration*.

The result that ensures the exact convergence of the conjugate gradient method applied to symmetric, positive definite matrices in at most N iterations, N being the size of the system,[15] has a counterpart for non-symmetric systems, under a slightly more restrictive hypothesis, for the biconjugate gradient method or the CGS method.

According to preliminary computations, which agree with the results found by other researchers (see e.g. Reference 3), the advantages of CGS with respect to the biconjugate gradient algorithm are considerable. Another recent method based on the biconjugated gradient technique has shown even better performance in terms of smoothness of the convergence curves as well as in terms of the number of iterations, namely the *stabilized biconjugate gradient* method (Bi-CGSTAB).[6] This method has also been used and compared with CGS for some fluid dynamics applications, showing a comparable behaviour.[17] However, for the sake of conciseness, only the CGS method was retained from this family for comparison with the other schemes, since it has been more widely used up to date.

2.5. *Non-linear iteration-based methods: quasi-Newton method*

During the last few decades, significant advances have been made in the numerical solution of non-linear vectorial equations. In particular, in applications involving the numerical discretization of non-linear partial differential equations, the discrete solution of the problem (either stationary or time-dependent) is frequently found as the solution of a non-linear vectorial equation expressed in the *quasi-linear* form

$$A(x)x = b(x). \tag{35}$$

Here $A(x)$ is a matrix whose dependence on the unknown vector x is in conformity with the nature of the problem under consideration. Equation (35) has to be solved iteratively according to a strategy that should take into account, among many other factors, the size of the system, the number of times that a similar system has to be solved (boundary value problem or initial value problem) and the degree of simplicity of the functional dependence of $A(x)$ on x.

As is well-known, the most powerful method of solving (35) iteratively is the *Newton–Raphson* algorithm. According to this scheme, an improved update for the solution is found as

$$x_{m+1} = x_m + J^{-1}(x_m)r_m, \tag{36}$$

where the residual vector is given by $r_m = b(x_m) - A(x_m)x_m$ and the Jacobian matrix J is calculated as

$$J(x_m) = \frac{\partial}{\partial x}(A(x_m)x_m - b(x_m)). \tag{37}$$

The Newton–Raphson algorithm provides a quadratic convergence rate, i.e. there exists a positive constant c_{NR} that verifies

$$\|x_m - x\| \leqslant c_{\mathrm{NR}}\|x_{m-1} - x\|^2, \quad m = 1, 2, \ldots, \tag{38}$$

where x is the solution of (35).

This algorithm is computationally expensive since it involves the computation of the Jacobian matrix as well as the solution of a different linear system at each iteration level. There are two possible ways to reduce the numerical cost of the iteration scheme: in both cases one can replace the Newton–

Raphson correction $J^{-1}(x_m)r_m$ by a simpler approximation. The first approach is based on the computation of the exact Jacobian and a subsequent approximation of the solution of the system for the correction $s_m = J^{-1}(x_m)r_m$ by means of an iterative process interrupted before complete convergence.[18,19] The second approach consists of replacing the Jacobian matrix by another simpler matrix to invert G_m and then approximating $J^{-1}(x_m)r_m \approx G_m^{-1}r_m$. Some algorithms from this second approach, initially devised for the non-linear equation (35), can be applied to the particular case of the linear system (1). A popular choice in this approach is to take

$$G_m = J(x_0), \quad m = 1, 2, \ldots, m_{\max}, \tag{39}$$

which is usually referred to as the *modified Newton–Raphson* method. This method has no application for our purposes since it implies the exact computation of the Jacobian at the first iteration, which is precisely our original linear problem. Other methods from this second approach come from the quasi-Newton algorithm family (see Reference 20 for an extensive survey), which is based upon the *secant* condition

$$x_m - x_{m-1} = H_m(r_m - r_{m-1}), \tag{40}$$

where H_m is an approximation of the matrix $J^{-1}(x_m)$. The matrix H_m has to be determined according to a recursive scheme by imposing suitable additional properties. Once an initial approximation to the inverted Jacobian has been defined, the method should be able to compute the successive updates for the solution x_m and for the approximate inverted Jacobian H_m according to these rules. For instance, if the first approximation to the inverted Jacobian, H_1, is symmetric, positive definite, one could pretend this character to be inherited by the successive updates H_m. This approach gives the so-called BFGS (Broyden–Fletcher–Goldfarb–Shanno) method, which has been used in many fields but whose application in the frame of non-symmetric matrices does not seem feasible.

If, in addition to the secant condition, we claim that the matrices H_m and H_{m-1} only differ in the monodimensional subspace given by the vector $x_m - x_{m-1}$, then the matrix H_m is completely determined as a function of the preceding H_{m-1} by the so-called quasi-Newton rank-one update or *Broyden method*, which is presented here according to the format proposed in Reference 10:

$$H_m = \left(I + \frac{s_{m-1} - H_{m-1}y_{m-1}}{s_{m-1}^{\mathrm{T}}(H_{m-1}y_{m-1})} s_{m-1}^{\mathrm{T}}\right) H_{m-1}, \tag{41}$$

where

$$s_{m-1} = x_m - x_{m-1}, \qquad y_{m-1} = r_{m-1} - r_m. \tag{42}$$

Using the auxiliary vector

$$w_{m-1} = \frac{s_{m-1} - H_{m-1}y_{m-1}}{s_{m-1}^{\mathrm{T}}(H_{m-1}y_{m-1})}, \tag{43}$$

the mth correction is given by

$$s_m = (I + w_{m-1}s_{i-1}^{\mathrm{T}})H_{m-1}r_m, \tag{44}$$

where the product $H_{m-1}r_m$ is found as

$$H_{m-1}r_m = \prod_{j=m-2}^{1} (I + w_j s_j^{\mathrm{T}}) H_1 r_m. \tag{45}$$

At each iteration one first obtains the initial correction $H_{m-1}r_m$ by means of (45). Note that the successive premultiplications are easily performed by taking into account the relationship

$$(I + w_j s_j^{\mathrm{T}})v_j = v_j + (s_j^{\mathrm{T}} v_j)w_j, \tag{46}$$

so that step (45) requires the storage of the successive vectors s_k and w_k, $k=1, \ldots, m-1$. Then $H_{m-1}r_m$ is substituted in (43) to obtain (and store) the vector w_{m-1}. Finally, the solution update s_m is computed by means of (44).

The iterative scheme has been presented in a way that allows for easy inclusion of a scalar relaxation factor at each iteration level. A *line search* parameter ρ_m can be introduced in the update formula according to

$$x_m = x_{m-1} + s_{m-1} = x_{m-1} + \rho_m H_{m-1} r_{m-1}. \tag{47}$$

Several techniques have been developed to estimate the optimal step length. However, for the type of problems under consideration, this approach has shown little advantage and so it has been preferred to select $\rho_m = 1$, $m = 1, \ldots, m_{\max}$. In this case the above algorithm admits some simplification. Defining the working vector $p = H_{m-1}r_m$, equations (43) and (44) yield

$$w_{m-1} = \frac{1}{\|s_{m-1}\|^2 - s_{m-1}^{\mathrm{T}} p} p, \qquad s_m = \frac{\|s_{m-1}\|^2}{\|s_{m-1}\|^2 - s_{m-1}^{\mathrm{T}} p} p, \tag{48}$$

which gives $w_{m-1} = s_m / \| s_{m-1} \|^2$, thus avoiding the need for storing the whole series of vectors w_k.

Broyden algorithm

First iteration: $s_0 \leftarrow B^{-1}(b - Ax_0)$
Update $x_1 \leftarrow x_0 + s_0$
Second iteration: $p \leftarrow B^{-1}(b - Ax_1)$
Compute and store $f(0) \leftarrow \| s_0 \|^2$
Compute $s_1 \leftarrow [f(0)/(f(0) - p^{\mathrm{T}} s_0)]p$
Update $x_2 \leftarrow x_1 + s_1$
For $m = 2, \ldots, m_{\max}$ repeat:
  Compute $p \leftarrow B^{-1}(b - Ax_m)$
  For $k = 0, \ldots, m-1$ repeat:
    Compute $p \leftarrow p + (p^{\mathrm{T}} s_k / f(k)) s_{k+1}$
  Next k
  Compute and store $f(m-1) \leftarrow \| s_{m-1} \|^2$
  Compute $s_m \leftarrow [f(m-1)/(f(m-1) - p^{\mathrm{T}} s_{m-1})]p$
  Update $x_{m+1} \leftarrow x_m + s_m$
  Convergence test: if $\| s_m \| < \varepsilon$, stop the calculation
Next m

3. NUMERICAL TESTS

Two hyperbolic model formulations have been considered for the following benchmark exercises. The first concerns the simple scalar convection–diffusion equation. For the second a semi-implicit version of the multicomponent inviscid Euler equations has been considered. An unstructured finite volume formulation has been used in both cases to carry out the spatial discretization. A different degree of implicitness in the time difference scheme has been adopted (fully implicit scheme for the former and semi-implicit scheme for the latter). The resulting matrices exhibit the typical non-symmetric character associated with implicit discretizations of non self-adjoint operators (as the advection operator). Seven iteration procedures have been tested for each problem: three of them can be classified as *strongly preconditioned* schemes (GMRES–ILU, Broyden–ILU and CGS–ILU); the remaining four schemes are *weakly preconditioned or non-preconditioned* (GMRES–diagonal, Broyden–diagonal, Gauss–

Seidel and Jacobi). The convergence curves (error measure versus iteration counter) for each case will be represented according to the following symbol notation.

1. Strongly preconditioned methods
   (a) GMRES method (with restart every five iterations) with ILU preconditioning ($\triangle$).
   (b) Broyden method ($\square$).
   (c) CGS method (with restart every five iterations) with ILU preconditioning ($\diamond$).
2. Weakly preconditioned or non-preconditioned methods
   (a) GMRES method (with restart every five iterations) with diagonal preconditioning (∇).
   (b) Broyden method (with restart every 10 iterations) with diagonal preconditioning ($\times$).
   (c) Gauss–Seidel method ($*$).
   (d) Jacobi method ($+$).

According to this test programme, the number of iterations to reach convergence can be compared only between schemes with the same level of preconditioning. For these the CPU time, the number of iterations and the smoothness and regularity of the convergence curve are factors that could be taken into consideration in selecting the optimum method. In contrast, for the overall comparison the basic performance parameter is only the CPU time.

The CGS–diagonal combination has been rejected because preliminary tests have shown that the CGS iterative scheme does not behave satisfactory unless it is combined with a 'good' preconditioning technique. The selection of the number of iterations allowed before a restart (with the computation of a new first guess for the solution and the corresponding initial residual vector) has also been made according to preliminary computations. The approximate optimum of five iterations per cycle for the GMRES method has also been reported by Nkonga.[21] Although they appear to be not far from the optimum for the presented cases, they certainly depend on the problem characteristics and could vary from case to case.

The error measure for iteration i selected for all the tests is $E(i) = \log(\| r_i \| / \| r_0 \|)$, where $\| a \|$ represents the Euclidean norm of vector a.

A typical time station of each problem has been selected to analyse the convergence curves and CPU times. All computations were made on an HP-835 computer using single-precision arithmetic.

3.1. *Scalar advection–diffusion equation*

Given a stationary velocity field $\mathbf{u}$ and a diffusion parameter κ, a passive scalar W is transported according to

$$\frac{\partial W}{\partial t} + \mathbf{u}\cdot\nabla W = \nabla\cdot(\kappa\nabla W). \tag{49}$$

Three different problems have been analysed. They have been borrowed from Nkonga,[21] who compared for these problems the performance of the GMRES and Jacobi methods combined with ILU and diagonal preconditionings.

A node-centred finite volume technique has been used to solve these problems. Let Ω_i represent the control volume around node i and $|\Omega_i|$ its area. Let W_i^n and W_i^{n+1} represent the control-volume-averaged values of the transported quantity at $t = t_n$ and $t = t_n + \Delta t = t_{n+1}$ respectively and define the nodal increment of the conserved quantity as $\delta W_i^n = W_i^{n+1} - W_i^n$. The advection–diffusion equation can be approximated by

$$|\Omega_i| \frac{\delta W_i^n}{\Delta t} = \int_{\Omega_i} [\nabla\cdot(\kappa\nabla W)]^{n+1} d\Omega - \int_{\Omega_i} [\nabla\cdot\mathbf{u}W]^{n+1} d\Omega = \sum_{j\in T(i)} \Psi_{ij}^{n+1} - \sum_{j\in T(i)} \phi_{ij}^{n+1}, \tag{50}$$

where ϕ_{ij}^{n+1} represents the *numerical convective flux* from volume Ω_i to volume Ω_j, Ψ_{ij}^{n+1} represents the *numerical diffusive flux* between the same control volumes and the sum is extended to the set of nodes $T(i)$ surrounding node i.

The computation of the *numerical convective flux* has been made as follows. Denoting by η_{ij} the surface normal of the interface between control volumes Ω_i and Ω_j, the average normal velocity across this interface is given by $u_{nij} = \eta_{ij} \cdot \bar{\mathbf{u}}_{ij}$, where $\bar{\mathbf{u}}_{ij} = 0{\cdot}5(\mathbf{u}_i + \mathbf{u}_j)$. The fully upwinded numerical flux of the conserved quantity across this interface can be computed according to

$$\phi_{ij} = u_{nij}(W_i + W_j) - |u_{nij}|(W_i - W_j). \tag{51}$$

If the velocity field is constant in time, the following linearization in time is exact:

$$\phi_{ij}^{n+1} = \phi_{ij}^{n} + \left[\frac{\partial \phi}{\partial W_i}\right]_n \delta W_i + \left[\frac{\partial \phi}{\partial W_j}\right]_n \delta W_j = \phi_{ij}^{n} + (u_{nij} - |u_{nij}|)\delta W_i + (u_{nij} + |u_{nij}|)\delta W_j. \tag{52}$$

The computation of the *numerical diffusive flux* has been made via a standard finite element discretization of the Laplace operator using linear triangles. The reader is referred to Reference 21 and 22 for further details on the numerical scheme. The assembly of the expressions for the implicit numerical fluxes yields the discrete system of equations that has to be solved iteratively at each time level, i.e.

$$A\delta W_i = \phi_{ij}^{n} + \Psi_{ij}^{n}, \tag{53}$$

where the finite-element-discretized explicit diffusive flux is second-order-accurate in space. A quasi-second-order scheme in space can be achieved also for the convective flux by putting in the right-hand side a second-order evaluation of the convective numerical fluxes entering the control volumes around each node, obtained with a MUSCL-like technique.[23]

3.1.1. Test case 1: concentration hill convection. The first problem consists of the translation of a concentration cone over a square domain. The problem description is

$$W_0(x, y) = 1 + \max\{0, 0{\cdot}1 - 5\sqrt{[(x - 0{\cdot}25)^2 + (y - 0{\cdot}25)^2]}\},$$

$$W_{\text{inlet}} = 1, \qquad \mathbf{u} = (1, 0{\cdot}5), \qquad \Omega = [0, 1] \times [0, 1].$$

The velocity field and initial conditions are shown in Figure 1. The inviscid solution for $\kappa = 0$ computed with a Courant–Friedrichs–Lewy (CFL) number of 0·4 at $t = 0{\cdot}4216$ and 0·75 is presented in Figure 2.

A series of tests varying the diffusion parameter for the same initial conditions and velocity field has been conducted for this problem. The aim is to analyse the convergence behaviour of the different methods applied to matrices representing several combinations between a 'pure' advective (hyperbolic)

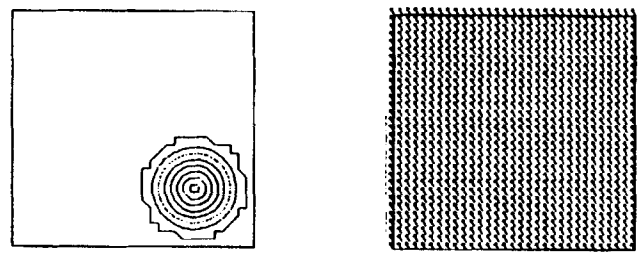

Figure 1. Initial conditions (left) and velocity field (right) for problem 3.1.1

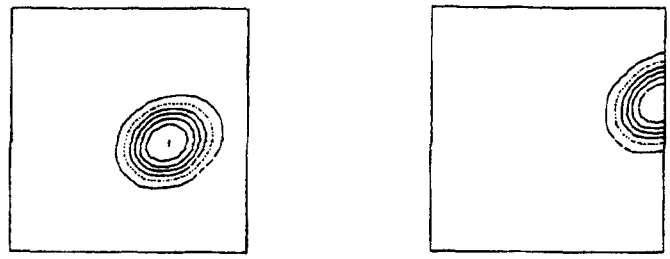

Figure 2. Solution to inviscid problems 3.1.1 for CFL = 0·4 at $t = 0{\cdot}4216$ (left) and 0·75 (right)

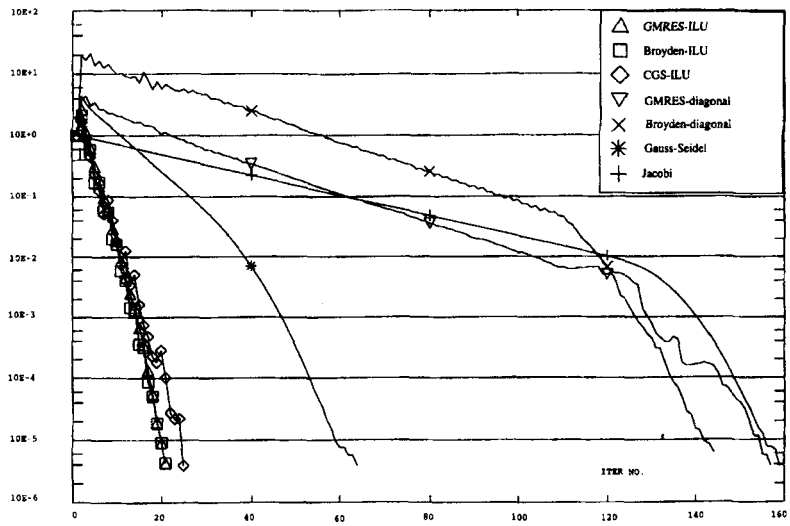

Figure 3. Convergence curves for problem 3.1.1 ($Pe = \infty$)

problem and a 'pure' diffusive (parabolic) problem. The appropriate parameter to measure this is the (grid) Peclet number $Pe = (|\mathbf{u}|\Delta x)/\kappa$, where Δx is the characteristic mesh size. A pure convective problem corresponds to $Pe = \infty$, while a pure diffusive problem implies $Pe = 0$. The selected values for this parameter were $Pe = \infty$, 3 and 0·15. The corresponding values of κ were $\kappa = 0$, 0·0025 and 0·05 respectively.

The convergence curves for the different methods tested for this problem are shown in Figures 3–5. The results are summarized in Table I.

Table I. Summary of test 3.1.1

	GMRES–ILU	Broyden–ILU	CGS–ILU	GMRES–diagonal	Broyden–diagonal	Gauss–Seidel	Jacobi
$Pe = \infty$	21 iter.	21 iter.	25 iter.	157 iter	144 iter.	64 iter.	159 iter.
	31 s	26 s	41 s	148 s	104 s	27 s	75 s
$Pe = 3$	27 iter.	26 iter.	23 iter.	157 iter.	147 iter.	102 iter.	239 iter.
	37 s	31 s	53 s	141 s	106 s	42 s	111 s
$Pe = 0{\cdot}15$	62 iter.	66 iter.	108 iter.	270-NC	400-NC	400-NC	400-NC
	82 s	72 s	163 s	250 s	302 s	162 s	189 s

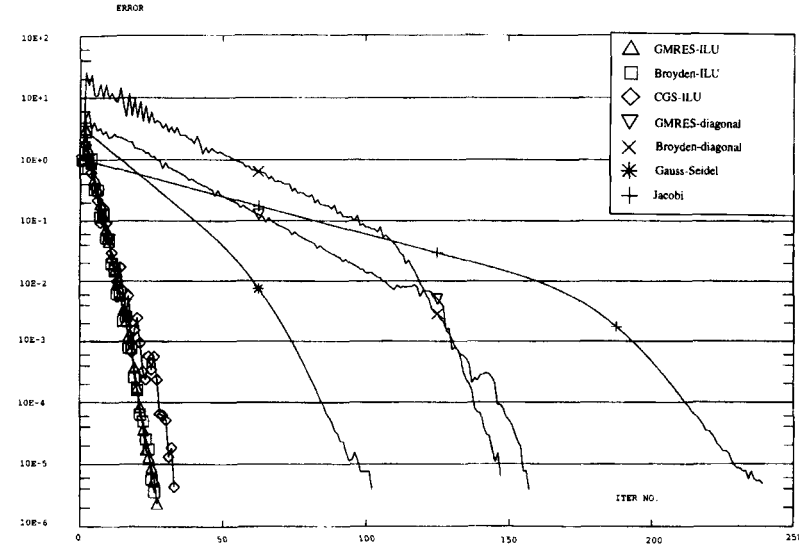

Figure 4. Convergence curves for problem 3.1.1 ($Pe = 3$)

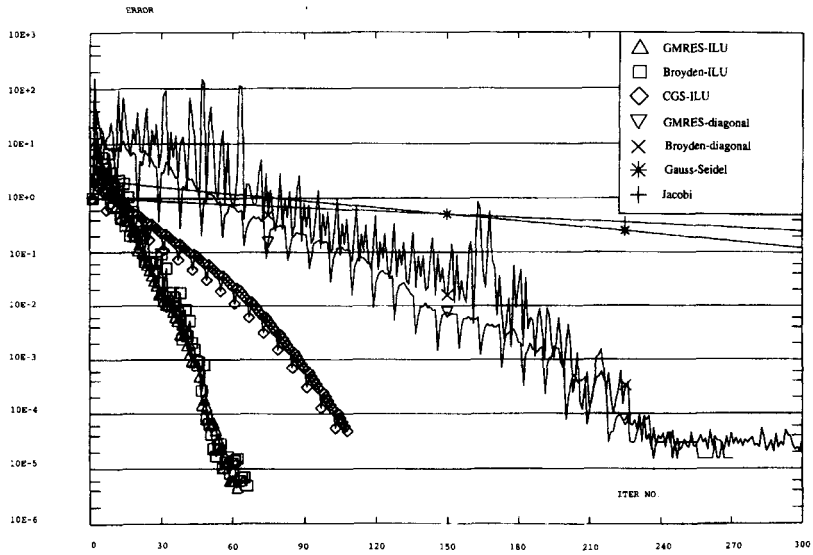

Figure 5. Convergence curves for problem 3.1.1 ($Pe = 0{\cdot}15$)

For all Peclet numbers, as expected, the number of iterations required when an ILU preconditioning technique is used is less than the corresponding number of iterations required for the same method with a weak preconditioning. However, in terms of computational cost the classical non-preconditioned relaxation methods seem to be competitive with the fully preconditioned gradient and Newton methods and the worst results are obtained with the gradient and Newton methods combined with a weak preconditioning. Another appealing feature of these results is the degeneration of the convergence properties of *all methods* as the Peclet number decreases, due to the spreading of the matrix

eigenvalues as the fast-moving phenomena associated with the diffusion term become more and more important in the model. For $Pe = 0{\cdot}15$ the slope of the convergence curve for the Jacobi and Gauss–Seidel methods becomes almost horizontal. The GMRES–diagonal and Broyden–diagonal methods exhibit a very irregular convergence curve. They reach an error level of about 5×10^{-4} in approximately 240 iterations, but the error remains constant around this value from that point on. The gradient and Newton methods with ILU preconditioning reach convergence satisfactorily, but a significant advantage is found for the Broyden and GMRES methods with respect to the CGS method when considering both the number of iterations and the CPU time. The reason for the high cost per iteration of the CGS method is the double back-substitution required at each iteration level. This tends to penalize this method, even for the same slope of the convergence curves (error versus iteration).

3.1.2. Test case 2: Gaussian hill rotation on a disc. The second test case for the advection–diffusion equation consists of the rotation of a Gaussian hill of density on a circular domain:

$$\begin{aligned} W_0(x, y) &= \exp[-10(x - 0{\cdot}15)^2 - 10(y - 0{\cdot}5)^2], \\ W_{\text{inlet}} &= 0, \qquad \Omega = B(0{\cdot}5, 0{\cdot}5, \sqrt{2}), \end{aligned} \tag{54}$$

where $B(0{\cdot}5, 0{\cdot}5, \sqrt{2})$ denotes the disc whose centre is at $(0{\cdot}5, 0{\cdot}5)$ and whose radius is $\sqrt{2}$. The computational grid (involving 10,201 nodes) is shown in Figure 6.

This case has been also used to study the influence of the Peclet number on the convergence patterns of the iterative schemes under consideration. Two runs have been made with $\kappa = 0{\cdot}0025$ ($Pe = 8{\cdot}8$) and $\kappa = 0{\cdot}017$ ($Pe = 1{\cdot}3$). The convergence curves for the former are plotted in Figure 7 and those corresponding to the latter are plotted in Figure 8. The results are summarized in Table II.

The general trends observed in test case 3.1.1 are repeated here. Among the relaxation methods, only the Gauss–Seidel method can be competitive with the ILU-preconditioned methods, although the degree of deterioration of the convergence (in terms of CPU time) when the Peclet number decreases from 8·8 to 1·3 is higher for the former (136 s/61 s = 2·29) than for the latter (46 s/28 s = 1·64 for GMRES–ILU, 38 s/22 s = 1·72 for Broyden–ILU and 59 s/33 s = 1·78 for CGS–ILU). Among the ILU-preconditioned methods, Broyden–ILU exhibits the best CPU times. On the other hand, the CPU cost of the diagonally preconditioned GMRES and Broyden methods seems to indicate that such a weak preconditioning is not adequate for these methods.

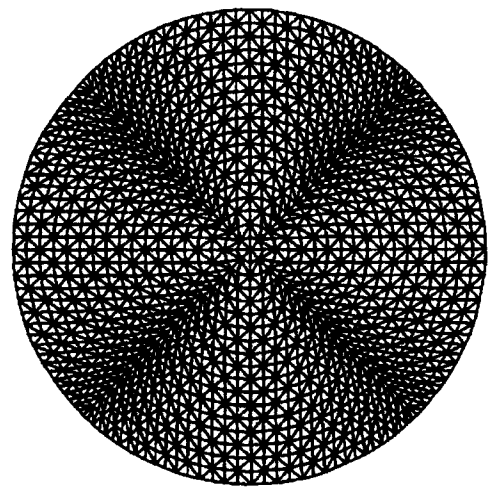

Figure 6. Computational grid for problem 3.1.2

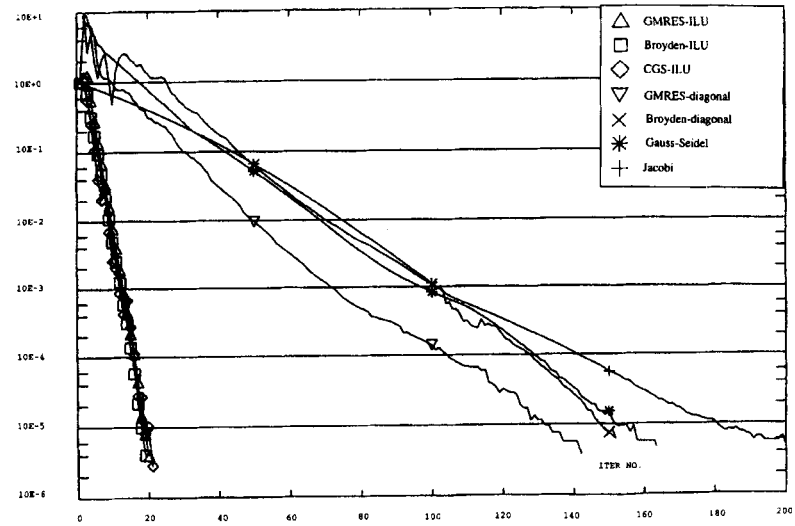

Figure 7. Convergence curves for problem 3.1.2 ($Pe = 8{\cdot}8$)

3.2. *Semi-implicit inviscid Euler equations*

The second model equation which will be considered for testing the set of algorithms under analysis is the system of inviscid, multicomponent Euler equations describing the flow of a mixture of Γ gases in two space dimensions, i.e.

$$\frac{\partial W}{\partial t} + \nabla \cdot \mathbf{F}(W) = 0, \tag{55}$$

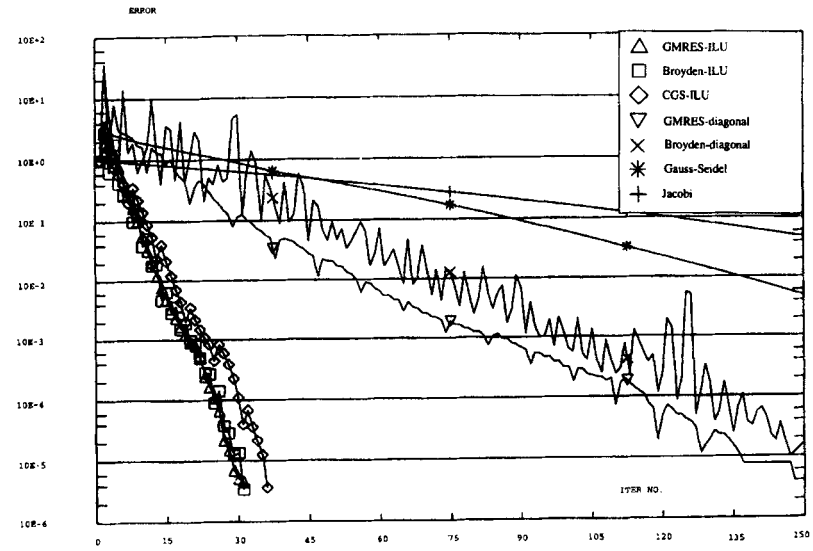

Figure 8. Convergence curves for problem 3.1.2 ($Pe = 1{\cdot}3$)

Table II. Summary of test 3.1.2

	GMRES–ILU	Broyden–ILU	CGS–ILU	GMRES–diagonal	Broyden–diagonal	Gauss–Seidel	Jacobi
$Pe = 8{\cdot}8$	20 iter.	19 iter.	21 iter.	142 iter.	152 iter.	164 iter.	200-NC
	28 s	22 s	33 s	124 s	103 s	61 s	89 s
$Pe = 1{\cdot}3$	31 inter.	31 inter.	36 inter.	148 inter.	161 inter.	352 inter.	400-NC
	46 s	38 s	59 s	141 s	120 s	136 s	118 s

where the conserved variables W and the components of the flux vector $\mathbf{F}$ are given by

$$W = \begin{bmatrix} \rho_1 \\ \rho_2 \\ \vdots \\ \rho_\Gamma \\ \rho u \\ \rho v \\ \rho E \end{bmatrix}, \qquad F_x = \begin{bmatrix} \rho_1 u_x \\ \rho_2 u_x \\ \vdots \\ \rho_\Gamma u_x \\ \rho u_x^2 + p \\ \rho u_x u_y \\ (\rho E + p) u_x \end{bmatrix}, \qquad F_y = \begin{bmatrix} \rho_1 u_y \\ \rho_2 u_y \\ \vdots \\ \rho_\Gamma u_y \\ \rho u_x u_y \\ \rho u_y^2 + p \\ (\rho E + p) u_y \end{bmatrix}. \tag{56}$$

Here ρ_k represents the partial density of component k, u_x and u_y are the components of the velocity vector $\mathbf{u}$, E is the total energy per unit mass and ρ is the total density given by $\sum_k \rho_k$. The total energy E is made up of the internal energy per unit mass, e (which is assumed to be only a function of temperature), and the kinetic energy per unit mass, $\| \mathbf{u} \|^2/2$. The closure equation linking the pressure p and the conserved quantities W_i is provided by the state equation

$$p = RT \sum_{k=1}^{\Gamma} \frac{\rho_k}{w_k} = (\gamma - 1)\rho e, \tag{57}$$

where R is the universal gas constant, T is the temperature, w_k is the molar mass of component k and γ is the ratio of specific heats at constant pressure and volume. The temperature is given by

$$T = \frac{e}{C_v}, \tag{58}$$

where the mixture heat capacity at constant volume is calculated according to

$$C_v = \sum_{k=1}^{\Gamma} C_{vk} \frac{\rho_k}{\rho}, \tag{59}$$

in which C_{vk} stands for the specific heat at constant volume for component k.

The spatial discretization scheme is a non-structured, node-centred finite volume formulation. Let η_{ij} be the normal to the interface between control volumes Ω_i and Ω_j. A projection of the flux vector $\mathbf{F}$ in the direction η_{ij} can be made, yielding $F_{ij} = \mathbf{F} \cdot \eta_{ij}$. The conservation system (55) is known to be hyperbolic, so the Jacobian matrix associated with F_{ij}, i.e.

$$A_{ij} = \frac{\partial F_{ij}}{\partial W}, \tag{60}$$

is diagonalizable and has real eigenvalues for all physically compatible states. In addition, it can be shown that $F_{ij} = A_{ij} W$. These properties imply that there exists a set of characteristic variables $U = f(W)$ for which the conservation equation (55) decouples into a system of $\Gamma + 3$ pure advection equations

(linear or not), whose advection speeds are the eigenvalues of the matrix $\underline{A}_{ij}$. These values are

$$\begin{aligned} \lambda_1 &= \mathbf{u}\cdot\eta_{ij} \quad (\Gamma + 1 \text{ times}), \\ \lambda_2 &= \mathbf{u}\cdot\eta_{ij} + \|\eta_{ij}\|c \quad (\text{one time}), \\ \lambda_3 &= \mathbf{u}\cdot\eta_{ij} - \|\eta_{ij}\|c \quad (\text{one time}), \end{aligned} \tag{61}$$

where c is the sound speed associated with the state W.

The first $\Gamma + 1$ characteristic fields are associated with *convective* waves whereas the last two are associated with *pressure* waves. If the sound speed is very large with respect to the fluid speed (i.e. low Mach number), an explicit time integration is unable to correctly track the propagation of all signals. The pressure waves associated with the eigenvalues λ_2 and λ_3 propagate very fast across the domain. Even if the *hyperbolic character* of the equations is preserved, the influence of the pressure waves affects the entire domain very rapidly. The same effect is associated with the diffusive term $\nabla\cdot(\kappa\nabla W)$ in the scalar advection–diffusion equation, which provides the *parabolic character* of (49). The correct tracking of the convective signals would require an implicit time integration scheme to overcome the severe time step restriction due to the CFL condition for the fast sonic waves.

Many *semi-implicit* formulations consist of a time discretization which is implicit only for the part of the flux directly linked to the pressure waves. The aim is to retain accuracy for the convectively transported signals, treating them explicitly with a convective CFL number close to unity, whereas the sonic part of the flux can be treated with time steps larger than those imposed by the CFL sonic restriction. The reader is referred to Reference 24 for a survey.

The time discretization scheme adopted here[25] is based on the consideration of the conservation equations integrated between times $t = t_n$ and $t = t_n + \Delta t = t_{n+1}$ using the mid-step flux formulation

$$\mathbf{F}^{n+1/2}(W) \approx \begin{bmatrix} (\rho_\gamma)^n \mathbf{u}^n \\ (\rho\mathbf{u})^n \mathbf{u}^n \\ (\rho E)^n \mathbf{u}^{n+1} \end{bmatrix} + \begin{bmatrix} 0 \\ p^{n+1} \\ p^{n+1}\mathbf{u}^{n+1} \end{bmatrix}. \tag{62}$$

The conservation equation in control volume i can be approximated by

$$|\Omega_i| \frac{\delta W_i^n}{\Delta t} + \int_{\Omega_i} [\nabla\cdot\mathbf{F}]^{n+1/2} d\Omega = |\Omega_i| \frac{\delta W_i^n}{\Delta t} + \sum_{j \in T(i)} \phi_{ij}^{n+1/2} = 0. \tag{63}$$

The *numerical flux* ϕ_{ij} at the interface between control volumes Ω_i and Ω_j has to be computed as a projected flux in the direction η_{ij}, depending on both states W_i and W_j:

$$\phi_{ij} = f(F_{ij}(W_i), F_{ij}(W_j)). \tag{64}$$

The Roe scheme has been selected to compute the numerical flux. This approach uses a secant approximation of the flux homogeneity relation $F_{ij} = \underline{A}_{ij} W$. An averaged state $\hat{W} = \hat{W}(W_i, W_j)$ is constructed in such a manner that $F_{ij}(W_i) - F_{ij}(W_j) = \underline{A}_{ij}(\hat{W})(W_i - W_j)$, thus allowing for an upwind treatment of the different characteristic fields. The explicit version of the Roe numerical flux fluction reads[26]

$$\phi_{ij}^n = 1/2(F_{ij}(W_i^n) + F_{ij}(W_j^n)) + \tfrac{1}{2}|\underline{A}_{ij}(\hat{W}^n)|(W_i^n - W_j^n), \tag{65}$$

where $|A| = T^{-1}|\Lambda|T$, T and T^{-1} being the matrices that reduce the matrix A to its diagonal form, and $|\Lambda| = \text{diag}\{|\lambda_1|, \ldots, |\lambda_1|, |\lambda_2|, |\lambda_3|\}$. The semi-implicit numerical fluxes are obtained according to the following two-step time linearization.

(a) The state vector is decomposed into mass flux (Γ entries) and momentum and energy flux (three entries). The same is done with the global flux, i.e. $W = \{\rho_\gamma, \omega\}$ and $\phi = \{\Psi_\gamma, \varphi\}$.

(b) The explicit equations *for the partial densities* are first solved (possibly with a MUSCL-like second-order evaluation of the mass fluxes ψ_γ^n). A set of advanced densities $\{\rho_1^{n+1}, \ldots, \rho_\Gamma^{n+1}\}$ is obtained.

(c) The numerical flux *for the momentum and energy*, computed according to (62), is obtained by means of a time linearization (similar to (52)) around the fictitious state $W^0 = \{\rho_1^{n+1}, \ldots, \rho_1^{n+1}, (\rho u_x)^n, (\rho u_y)^n, (\rho E)^n\}$. Since the numerical flux has only three non-zero entries in this step, the corresponding Jacobian blocks are of constant size (3 × 3), independently of the number of components in the mixture:

$$\varphi_{ij}^{n+1/2} = \varphi_{ij}^n + \left[\frac{\partial \varphi}{\partial \omega_i}\right]_0 \delta\omega_i + \left[\frac{\partial \varphi}{\partial \omega_j}\right]_0 \delta\omega_j.$$

(d) The explicit flux term φ_{ij}^n is also computed according to a MUSCL-like second-order approximation.

(e) A Lax–Wendroff-like stabilization term of the form $\nabla\cdot(0{\cdot}5\Delta t[\mathbf{uu}]\cdot\nabla W)$ is added to the equations. This allows us to reach a *selectively second-order accuracy in time* for the convective modes. The reader is referred to Reference 25 for details.

To summarize, the described numerical technique requies at each time level, after the density update, the solution of the system of equations

$$|\Omega_i| \frac{\delta\omega_i^n}{\Delta t} + \sum_{j \in T(i)} \varphi_{ij}^n + \left[\frac{\partial \varphi}{\partial \omega_i}\right]_0 \delta\omega_i + \left[\frac{\partial \varphi}{\partial \omega_j}\right]_0 \delta\omega_j = 0, \tag{66}$$

whose size is $3N_g \times 3N_g$, N_g being the number of grid points. The linear system will be solved using the iterative schemes under analysis for some selected problems.

3.2.1. Test case 1: subsonic shock tube. The first test case concerns a subsonic shock tube calculation. The computational domain is a tube (1 unit-length long, 0·1 unit-length wide) divided into two parts (each 0·5 unit-length long) by a membrane. The grid is a regular triangulation of the domain with 100 divisions in the main direction and three divisions in the transversal direction, thus having 303 nodes. A perfect gas with $\gamma = 1{\cdot}4$ fills the tube, the left part at pressure $p = 1$ and temperature $T = 2{\cdot}5$, the right at $p = 0{\cdot}95$ and $T = 2{\cdot}5$. At time $t = 0$ the membrane is assumed to disappear. Three characteristic waves appear: a shock propagates to the right with velocity $u + c$, a contact discontinuity moves also to the right with velocity u and a leftwise-moving rarefaction wave appears travelling with velocity $u - c$. the density field at times $t = 0$, 0·05, 0·1, 0·15 and 0·2 is shown in Figure 9. Note that for the final time the shock wave has reached the position $x = 0{\cdot}75$, whereas the contact discontinuity is still very vlose to $x = 0{\cdot}5$ (the maximum Mach number reached by the flow is about 0·018).

The convergence curves for this problem are shown in Figure 10. In order to study the influence of the grid size and orientation on the results, the same problem has been tested on a unique ribbon of triangular elements (202 nodes) oriented in a different manner. The convergence curves corresponding to these computations are shown in Figure 11. The results are summarized in Table III.

The most interesting feature of Table III comes from the significant differences found between the problem treated either with three or with two rows of nodes. The matrices associated with these two cases are supposed to possess similar properties from the *physical* characteristics of the problem, but the *spatial discretization support* provides a noticeable difference. The two-row case yields a more ill-posed problem since it tends to reflect two parallel Riemann problems that actually are almost uncoupled. The more rows of nodal control volumes added, the stronger the coupling between these Riemann problems is through the normal between each row of nodes. The worst-performing methods are again the gradient and Newton methods combined with a diagonal prconditioning. The Gauss–

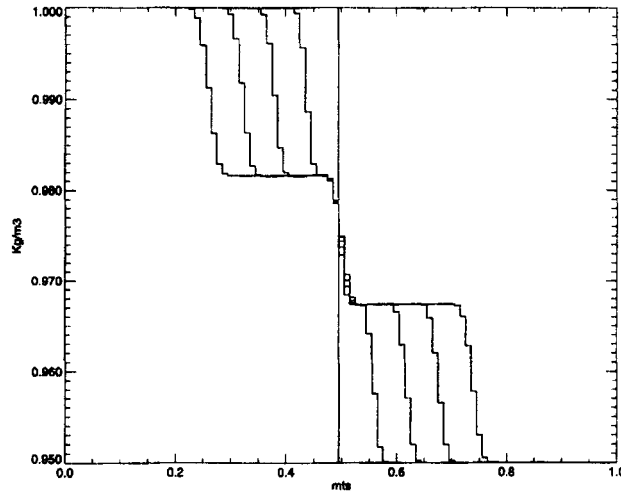

Figure 9. Density evolution for problem 3.2.1

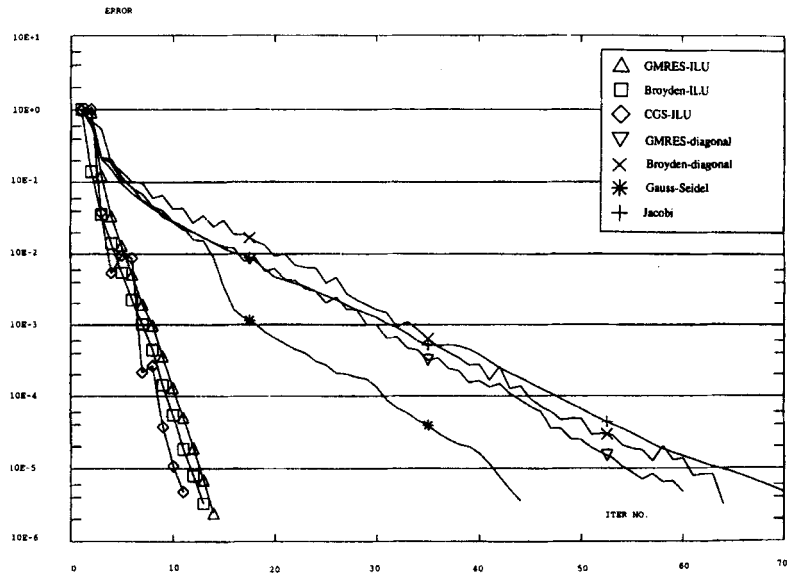

Figure 10. Convergence curves for problem 3.2.1 (303 nodes)

Seidel method is the best-performing method in both cases in terms of CPU time, thus providing a very competitive alternative to the ILU-preconditioned methods. However, one should also take into account that the CPU time might not be relevant for such small problems. The comparison among the GMRES–ILU, Broyden–ILU and CGS–ILU methods reveals that the second one is the best performing method also for these cases.

3.2.2. Test case 2: gravity-induced gas oscillation in an annular domain. An annular domain (inner radius 0·2, outer radius 0·5) is filled with a perfect gas with $\gamma = 1{\cdot}4$ and $C_v = 1$. The gas is initially at rest with $p = 1$ and $T = 0{\cdot}5$. At times $t = 0$ a gravitational acceleration begins to act, producing a

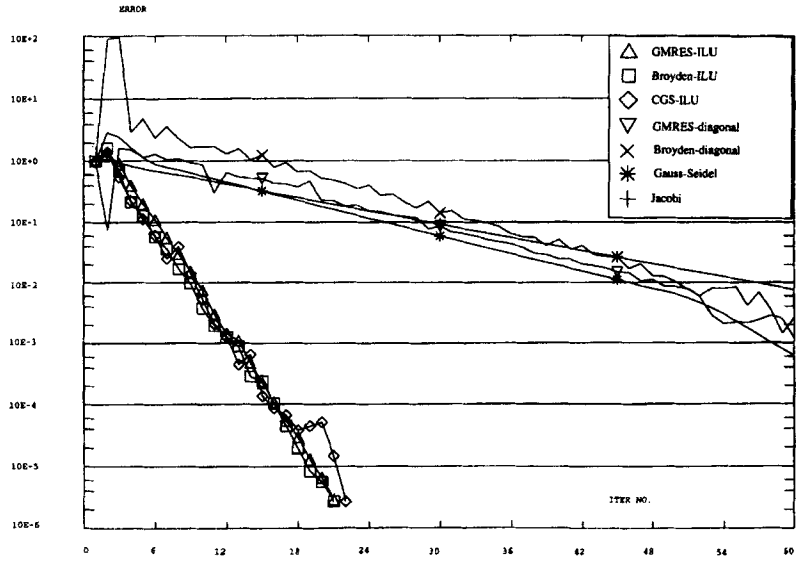

Figure 11. Convergence curves for problem 3.2.1 (202 nodes)

Table III. Summary of test 3.2.1

	GMRES–ILU	Broyden–ILU	CGS–ILU	GMRES–diagonal	Broyden–diagonal	Gauss–Seidel	Jacobi
303 nodes	14 iter.	13 iter.	11 iter.	60 iter.	64 iter.	44 iter.	70 iter.
	1·28 s	0·94 s	1·12 s	3·01 s	2·22 s	0·82 s	1·71 s
202 nodes	21 iter.	21 iter.	22 iter.	106 iter.	113 iter.	91 iter.	143 iter.
	1·31 s	0·98 s	1·58 s	3·54 s	2·23 s	0·93 s	1·79 s

stratification of the fluid. Pressure waves develop which reach the lower walls rapidly and are then reflected. After a series of reflections the hydrostatic pressure field should be reached, with the density accumulated in the lower part of the domain and the fluid at rest again. Depending on the intensity of the gravity, the flow regime can be characterized by a low Mach number (for weak gravity) or develop a transonic pattern (for large enough gravity). The oscillation pattern of sonic waves can be clearly appreciated in the numerical simulations if an adequate shock-capturing scheme is used for any value of the gravity.

Two explicit second-order calculations have been made on a 1D geometry to illustrate this. The results are shown in Figures 12 and 13, where the evolution of the pressure at various heights ($x = 0$, 0·25, 0·5, 0·75 and 1) is plotted. The initial conditions for both cases were as described. Figure 12 corresponds to a gravity value $g = 0{\cdot}01$ and Figure 13 corresponds to $g = 1$.

The necessary dissipation required to reach the hydrostatic pressure and density profile is provided in this case by the numerical viscosity of the semi-implicit numerical scheme, especially designed to smear out the sonic waves rapidly, retaining good accuracy for the convective signals.

For the 2D annular domain test problem (2100 nodes) a moderate value of the gravitational acceleration has been selected, $\mathbf{g} = (0, -0{\cdot}5)$. The Mach number field for the annular domain problem is plotted at time $t = 1{\cdot}46$ s in Figure 14.

The convergence curves for the seven iterative schemes under analysis for a typical time step of this test case are plotted in Figure 15. The results are summarized in Table IV.

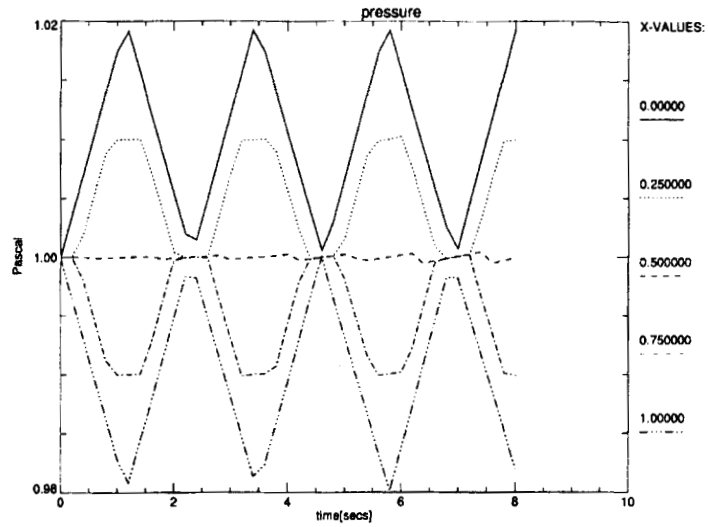

Figure 12. 1D stratification problem: $g = 0{\cdot}01$

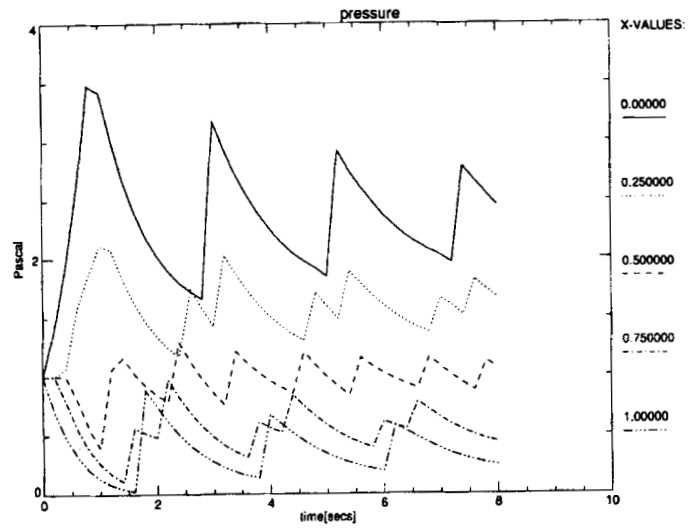

Figure 13. 1D stratification problem: $g = 1$

One can clearly distinguish in Figure 15 the two slopes in the convergence curves corresponding to the ILU-preconditioned algorithm family and the weakly conditioned or non-conditioned algorithms. In terms of CPU time the GMRES–diagonal and Broyden–diagonal methods are found to be too expensive. The Gauss–Seidel method is around two to three times more expensive than the ILU-preconditioned methods. Among the latter, the best-performing method in this case is the Broyden quasi-Newton method. Among the gradient methods, CGS behaves better than GMRES, which exhibits an even greater computational cost per iteration than CGS. This tends to show that the additional cost of the second back-substitution in the CGS method is compensated, for relatively large cases like this (6300 degrees of freedom), by the number of auxiliary operations (Gram–Schmidt orthogonalizations, Givens rotation processes, etc.) required in the GMRES method.

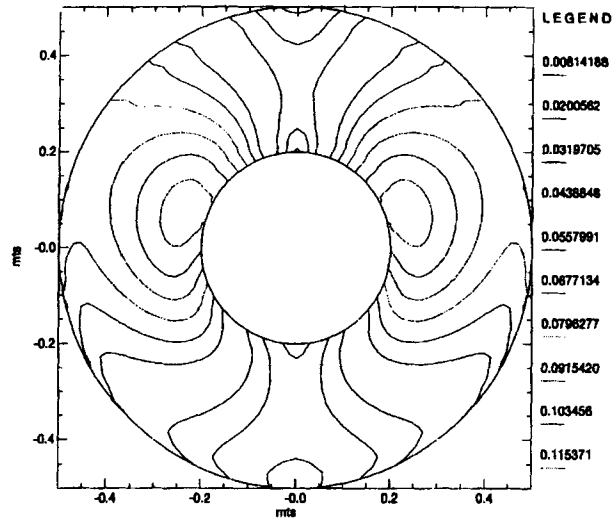

Figure 14. Problem 3.2.2: Mach number field at $t = 0{\cdot}146$ s

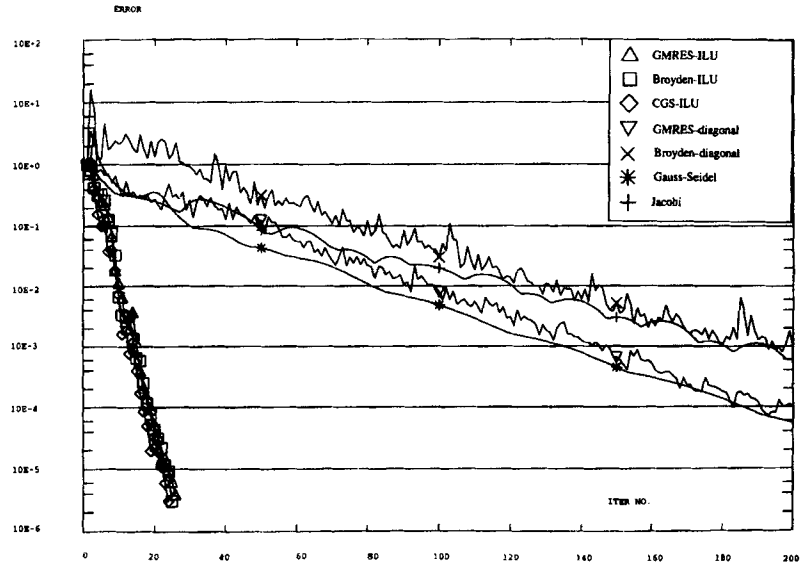

Figure 15. Convergence curves for problem 3.2.2

Table IV. Summary of test 3.2.2

	GMRES–ILU	Broyden–ILU	CGS–ILU	GMRES–diagonal	Broyden–diagonal	Gauss–Seidel	Jacobi
2100 nodes	24 iter. 27 s	25 iter. 14 s	24 itr. 19 s	254 iter. 160 s	349 iter. 124 s	258 iter. 63 s	350 iter. 95 s

3.2.3. Test case 3. convection of a density wave in a two-component mixture. The third test problem concerning the multicomponent Euler equations consists of the simple 2D convection of the density peak of a gas A in a background gas B within a squared computational domain by a constant, externally imposed velocity field. Both gases have $\gamma = 1{\cdot}4$. The problem is physically very similar to the test case 3.1.1 but is governed now by a *system* of partial differential equations. In this case the uniform velocity field results as the solution at each time step of the momentum conservation equation. The computational grid is a triangulation of the unit square by a 60 × 60 element pattern, yielding a total number of degrees of freedom equal to 11,163. The convergence curves for the seven analysed methods are shown in Figure 16, while the number of iterations and the required CPU time to reach convergence are given in Table V.

The aim of this test was to treat a relatively large case of a purely convective flow where the 'fast' characteristics associated with pressure waves were not relevant. Comparing the results obtained for this case with those corresponding to the preceding test problem, the most noticeable fact is that even when the system size is larger, the convergence of all the methods is much faster both in terms of iterations as well as in terms of CPU time.

These results show that the Gauss–Seidel method is the most efficient in this case, followed by the Broyden–ILU method. Note also that the CGS–ILU method is the most efficient solely in terms of iterations.

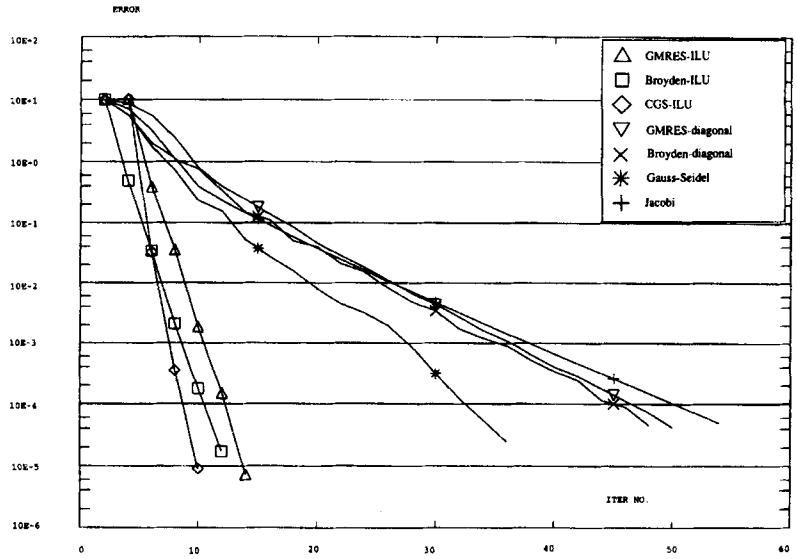

Figure 16. Convergence curves for problem 3.2.3

Table V. Summary of test 3.2.3

	GMRES–ILU	Broyden–ILU	CGS–ILU	GMRES–diagonal	Broyden–diagonal	Gauss–Seidel	Jacobi
3721 nodes	7 iter. 9·5 s	6 iter. 7·9 s	5 iter. 9·3 s	25 iter. 18·8 s	24 iter. 14·3 s	18 iter. 6·5 s	27 iter. 12·4 s

With respect to the preceding test case the Gauss–Seidel improvement ratio in CPU time is 63 s/6·5 s = 9·7, while those corresponding to the Broyden–ILU, CGS–ILU and GMRES–ILU methods are 14 s/7·9 s = 1·77, 19 s/9·3 s = 2·04 and 27 s/9·5 s = 2·84 respectively.

The conclusion is that for purely convective problems in which the matrix eigenvalues are not spread within a large range, the Gauss–Seidel method seems to become relatively more competitive with respect to the ILU-preconditioned gradient and Newton methods, whose performance is less sensitive (owing to the preconditioning process) to the spectral structure of the system matrix. These results are in agreement with what was observed in the test cases for the scalar advection–diffusion equation. The role played there by the viscosity in representing fast phenomena has been undertaken in the present problem by the sonic phenomena (pressure waves).

3.2.4. Test case 4: buoyancy-induced bubble motion. This last problem has been included as an example of an extremely badly conditioned system. It corresponds to a case with very low Mach number. The computational domain is a square with 10 m side, filled with a heavy gas ($\gamma = 1{\cdot}28$). In the lower left corner a bubble of a light gas ($\gamma = 1{\cdot}4$) is initially placed. All the boundaries are supposed closed. The initial pressure and temperature are 5×10^4 Pa and 273 K. From $t = 0$ the difference in densities induces motion of the bubble. The characeristic Mach number of this flow is around 10^{-4}. The velocity field at two instants is shown in Figure 17.

The main difficulty associated with this problem consists of the uncoupling between the energy field and the velocity field. For such a low Mach number the kinetic energy of the particles is almost negligible with respect to the internal energy of the fluid, which is linked to the pressure through the

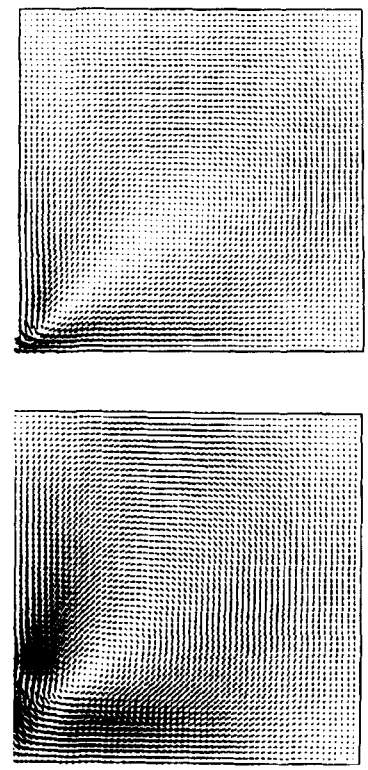

Figure 17. Problem 3.2.4: velocity fields at $t = 10$ (top) and $t = 30$ (bottom)

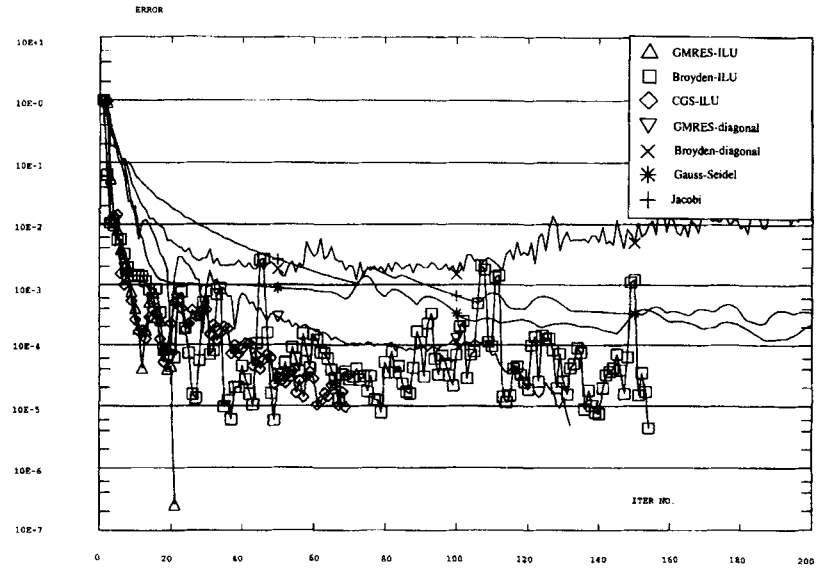

Figure 18. Convergence curves for problem 3.2.4

Table VI. Summary of test 3.2.4

	GMRES–ILU	Broyden–ILU	CGS–ILU	GMRES–diagonal	Broyden–diagonal	Gauss–Seidel	Jacobi
3721 nodes	21 iter. 29 s	154 iter. 139 s	68 iter. 557 s	132 iter. 99 s	Diverges	400-NC	400-NC

state equation. However, the pressure gradient (due mainly to the density gradient) induces fluid motion through the momentum equation. The velocity field has become almost a divergence-free field, coupled with the energy equation only indirectly though the density variations. For such a low-Mach number situation the compressible, multicomponent Euler equations are no longer adequate and a better representation to model this problem would be a multicomponent, incompressible approach where the closure relation provided by the equation of state $p = f(\rho_1, \ldots, \rho_\Gamma, e)$ is replaced by the solenoidal condition for the velocity, $\nabla \cdot \mathbf{u} = 0$.

The convergence curves are obtained for this problem are shown in Figure 18 and reflect the numerical stiffness of the problem. The results are summarized in Table VI.

These results put in evidence the stiffness of the problem under consideration. None of the methods is able to reduce the logarithmic error measure below about 10^{-5}, with the exception of the GMRES-ILU method which yields an error of about 5×10^{-5} in 21 iterations. For the other ILU-preconditioned techniques the convergence criterion has been relaxed to $\varepsilon = 10^{-5}$. The Broyden–ILU method exhibits an extremely irregular convergence path (in fact, from iteration 70 to 154 it does not yield any significant improvement in the solution). The CGS–ILU method converges more smoothly in 68 iterations, but the price to pay (two back-substitutions at each iteration level) makes its computational cost extremely high with respect to the other two methods. Another interesting fact is that the GMRES–diagonal method succeeds in reaching convergence within 132 interations and 99 s, with quite a smooth convergence path. The error reaches a plateau of 10^{-4} in about 80 iterations and remains more or less constant up to iteration 110, where the curve again takes a smooth negative slope to reach convergence at iteration 132. These results seem to show a relative advantage in terms of

algorithmic robustness for the GMRES methods with respect to the Broyden method in the presence of extremely badly conditioned matrices.

4. CONCLUSIONS

Let us recall that the aim of this work was to identify adequate solution schemes to treat the linear systems of equations arising from an implicit (or semi-implicit) time integration scheme (combined with an unstructured finite volume spatial discretization) applied to hyperbolic problems where a parabolic operator is present or where parts of the characteristics signals have fast propagating speeds that induce a certain degree of *numerical stiffness* in the problem. Some recent and sophisticated iterative methods (GMRES and CGS) from the gradient algorithm family have been compared with a quasi-Newton method (Broyden) as well as with standard relaxation methods (Gauss–Seidel and Jacobi). The main conclusions that can be drawn from the numerical tests presented in this paper can be summarized as follows.

The *most efficient* iterative schemes have been found in the combination of the gradient (GMRES and CGS) and Newton methods with a relatively good preconditioning technique (incomplete lower–upper decomposition). The use of these iterative methods in combination with a weak preconditioning technique is not competitive in terms of CPU time with the other tested techniques and should be avoided. However, it is underlined that standard relaxation methods (mainly the Gauss–Seidel method) perform *remarkably well* in most of the analysed tests.

The *relative CPU time* advantage observed for the ILU-preconditioned gradient and Newton methods with respect to the Gauss–Seidel method tends to disappear when the discretized physical problem loses its fast characteristic part (either its viscous or sonic character). Indeed, for pure convection problems the Gauss–Seidel method becomes an efficient alternative to the ILU-preconditioned gradient and Newton methods, but the performance of the former is much more dependent on the spectral pattern of the system matrix than that of the latter.

The comparison among the GMRES–ILU, CGS–ILU and Broyden–ILU methods reveals a slight advantage for the last one in terms of CPU time and number of iterations. Second in this ranking would be the GMRES–ILU method. The comparative advantage (in terms of CPU time) of the GMRES–ILU method with respect to the CGS–ILU method (due to the double back-substitution at each iteration level in the latter) seems to disappear as the problem size increases.

Concerning the *algorithmic complexity*, CGS–ILU and Broyden–ILU have an easier practical implementation than GMRES–ILU.

In addition, the convergence curves of CGS–ILU seem to be smoother, thus suggesting a greater *robustness* of the algorithm when compared with GMRES–ILU or Broyden–ILU.

These conclusions are somewhat disappointing, since it is impossible to establish a clear recommendation. It seems that a good preconditioner is mandatory to make the gradient and Newton schemes perform optimally. According to our experience, a computer programme aiming at treating a wide range of applications should be supplied with a variety of system solver techniques. Among these, one should select at least two well-preconditioned schemes (possibly Broyden and GMRES). On the other hand, for convection-dominated flows the Gauss–Seidel method, which is simple to implement and is present in many linear algebra packages, is very competitive and should also be retained.

Although not explored here, a large variety of ILU-like preconditioning methods are also offered to the code designer. A simple technique that could allow for significant CPU time savings would consist of freezing a given ILU decomposition for a number of time steps and updating it only when convergence difficulties are detected. This would permit one to avoid (at least partially) the computational cost of the incomplete factorization at each time step calculation.

The situation in which the matrix handling becomes impracticable (owing to the number of degrees of freedom, typically for 3D flows) would probably change this scenario. More sophisticated techniques, possibly based upon multigrid schemes, allowing a parallel treatment would then be required, but this goes beyond the scope of this work.

ACKNOWLEDGEMENTS

The authors wish to thank Messrs. Guillard and Nkonga from INRIA (Sophia-Antipolis) for some helpful discussions as well as for providing the tests corresponding to the advection–diffusion scalar equation.

REFERENCES

1. H. P. Langtangen, 'Conjugate gradient methods and *ILU* preconditioning of non-symmetric matrix systems with arbitrary sparsity patterns', *Int. j. numer. methods fluids*, **9**, 213–233 (1989).
2. G. Montero, R. Montenegro, G. Winter and L. Ferragut, 'Aplicación de esquemas EBE en procesos adaptativos', *Métodos Numér. Cálc. Diseño Ing.*, **6**, 311–332 (1990).
3. P. Joly and R. Eymard, 'Preconditioned biconjugate gradient methods for numerical reservoir simulation', *J. Comput. Phys.*, **91**, 298–309 (1990).
4. H. A. van der Vorst, 'The convergence behaviour of some iterative solution methods', in R. Gruber, J. Periaux and R. P. Shaw (eds), *Proc. Fifth Int. Symp. on Numerical Methods in Engineering*, Vol. 1, Springer, Berlin, 1989, pp. 61–72.
5. P. Sonneveld, 'CGS: a fast Lanczos-type solver for nonsymmetric linear systems', *SIAM J. Sci. Stat. Comput.*, **10**, 36–52 (1989).
6. H. A. van der Vorst, 'BI-CGSTAB: a fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems', *SIAM J. Sci. Stat. Comput.*, **13**, 631–644 (1992).
7. W. E. Arnoldi, 'The principle of minimized iteration in the solution of the matrix eigenvalue problem', *Q. Appl. Math.*, **9**, 17–29 (1951).
8. Y. Saad and M. H. Schultz, 'GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems', *SIAM J. Sci. Stat. Comput.*, **7**, 856–869 (1986).
9. F. Shakib, T. J. R. Hughes and Z. Johan, 'A multi-element group preconditioned GMRES algorithm for nonsymmetric problems arising in finite element methods', *Comput. Methods Appl. Mech. Eng.*, **75**, 415–456 (1989).
10. M. S. Engelman, G. Strang and K. J. Bathe, 'The application of quasi-Newton methods in fluid dynamics', *Int. j. numer. methods eng.*, **17**, 707–718 (1981).
11. C. G. Broyden, 'A class of methods for solving nonlinear simultaneous equations', *Math. Comput.*, **19**, 577–593 (1965)
12. J. A. Meijerink and H. A. van der Vorst, 'An iterative solution method for linear systems of which the coefficient matrix is symmetric M-matrix', *Math. Comput.*, **31**, 148–162 (1977).
13. P. G. Ciarlet, *Introduction à l'Analyse Numérique et à l'Optimisation*, Masson, Paris, 1986.
14. R. S. Varga, *Matrix Iterative Analysis*, Prentice-Hall, Englewood Cliffs, NJ, 1962.
15. M. R. Hestenes and E. Stiefel, 'Methods of conjugate gradients for solving linear systems', *Natl. Bur. Stand. J. Res.*, **49**, 409–436 (1952).
16. R. Fletcher, 'Conjugate gradient methods for indefinite systems', in *Lecture Notes on Mathematics*, Vol. 506, Springer, Berlin, 1976, pp. 73–89.
17. M. Buffat and L. Hallo, personal communication, École Centrale de Lyon, 1993.
18. A. H. Sherman, 'On the Newton-iterative methods for the solution of nonlinear equations', *SIAM J. Numer. Anal.*, **15**, 775–771 (1978).
19. P. N. Brown, 'A local convergence theory for combined inexact Newton/finite difference projection methods', *SIAM J. Numer. Anal.*, **24**, 407–424 (1987).
20. J. E. Dennis and J. J. Moré, 'Quasi-Newton methods: motivatin and theory', *SIAM Rev.*, **19**, 46–89 (1977).
21. B. Nkonga, 'Développement de méthodes numériques pour les écoulements tridimensionnels réactifs dans un domain déformable', *Thèse*, Université de Nice, 1992.
22. L. Fezoui and B. Stoufflet, 'A class of implicit upwind schemes for Euler simulations on unstructured grids,' *J. Comput. Phys.*, **84**, 174–206 (1989).
23. B. van Leer, 'Towards the ultimate conservative difference scheme. V: A second order sequel to Godunov's method', *J. Comput. Phys.*, **32**, 101–136 (1979).
24. G. Fernandez, 'Simulation numérique d'écoulements réactifs à petit nombre de Mach', *Thèse*, Université de Nice, 1989.
25. J. Donea, F. Ruel and A. Soria, 'On the numerical solution of hyperbolic problems', in F. Navarrina, M. Casteleiro and E. Oñate (eds), *Proc. II Congr. de Métodos Numéricos en Ingenieria*, 1993, pp. 19–30.
26. P. L. Roe, 'Approximate Riemann solvers, parameter vectors and difference schemes', *J. Comput. Phys.*, **43**, 357–372 (1981).