AN ADAPTIVE FINITE ELEMENT METHOD FOR THE
COMPRESSIBLE NAVIER-STOKES EQUATIONS

P. J. Capon\textsuperscript{1} and P. K. Jimack\textsuperscript{1}

ABSTRACT

This paper concerns the steady solution of the compressible Navier-Stokes equations, using a simple but stable finite element method on unstructured meshes. An adaptive approach is taken so that an initial mesh is refined during solution, using a residual based criterion to indicate where to refine. We show numerical results which demonstrate the application of this strategy.

1 INTRODUCTION

Numerical methods for solving the Euler and Navier-Stokes equations have undergone much development in the past few years, with a number of techniques in current use. In this paper the combination of finite elements with unstructured meshes, which enable an adaptive approach to be taken, is shown to provide a robust and efficient method for solving the compressible Navier-Stokes equations. We mainly consider steady solutions of laminar flow for both transonic and supersonic cases.

The Navier-Stokes equations and boundary conditions for compressible flow are given in the next section, and discretized via a finite element method described in section 3. The standard Galerkin approach is inadequate for the Navier-Stokes equations, introducing spurious numerical oscillations in the solutions, so in order to overcome this, the Galerkin/least squares modification has been developed by Hughes [1],[2].

This discretization of the problem leads to a system of nonlinear algebraic equations to be solved, and details of this procedure are given in section 4. Section 5 describes how the mesh is adaptively refined by the addition of extra nodes in regions where the error is greatest. The type of refinement adopted, which is based upon the work of Lohner [3], requires an adaptive algorithm which determines where to refine, without necessarily having an estimate of the overall error. One such strategy is to use the residual of the original problem, an approach taken by Hansbo and Johnson [4] for example.

In order to demonstrate the effectiveness of these methods, some numerical results for transonic flow around an aerofoil and supersonic flow over a flat plate are given in section 6. Finally section 7 outlines further work currently being undertaken and possible future areas of research.

2 THE NAVIER-STOKES EQUATIONS

The compressible Navier-Stokes equations may be written in terms of non-dimensionalized primitive variables as follows:

\[
\rho_t + \mathbf{u} \cdot \nabla \rho + \rho \nabla \mathbf{u} = 0
\]

\textsuperscript{1}School of Computer Studies, University of Leeds, Leeds LS2 9JT, United Kingdom.
\[ u_t + (u \cdot \nabla) u + (\gamma - 1) \left( \frac{T}{\rho} \nabla \rho + \nabla T \right) = \frac{1}{Re \rho} [\Delta u + \frac{1}{3} \nabla (\nabla \cdot u)] \quad (2) \]
\[ T_t + u \cdot \nabla T + (\gamma - 1) T \nabla \cdot u = \frac{1}{Re \rho} \left( \frac{\gamma}{Pr} \Delta T + F(\nabla u) \right) \quad (3) \]

where \( \rho, u \) and \( T \) are the density, velocity and temperature variables respectively. \( Re \) is the Reynolds number, \( Pr \) is the Prandtl number and \( \gamma \) is the ratio of specific heats.

For two dimensional flows, \( u = (u, v) \) and \( F(\nabla u) \) has the form

\[
F(\nabla u) = \frac{4}{3} \left[ (\frac{\partial u}{\partial x})^2 + (\frac{\partial v}{\partial y})^2 - \frac{\partial u}{\partial x} \frac{\partial v}{\partial y} \right] + \left( \frac{\partial v}{\partial x} + \frac{\partial u}{\partial y} \right)^2 . \quad (4)
\]

This formulation follows the approach of Bristeau et al [5]. Initially we consider steady state problems, and in this case the terms \( \rho_t, u_t \) and \( T_t \) will not be present. The equations are to be solved in the region \( \Omega \subset \mathbb{R}^2 \), with a boundary \( \Gamma \). In the case of external flows, we define \( \Gamma_{\infty} \) to be the farfield boundary, which is split up into the inflow boundary,

\[ \Gamma_{\infty}^- = \{ x | x \in \Gamma_{\infty}, u_{\infty} \cdot n < 0 \} , \quad (5) \]

and outflow boundary

\[ \Gamma_{\infty}^+ = \Gamma_{\infty} \setminus \Gamma_{\infty}^-, \quad (6) \]

where \( n \) is the unit vector of the outward normal to the farfield boundary. For a given angle of attack \( \alpha \),

\[ u_{\infty} = \begin{pmatrix} \cos \alpha \\ \sin \alpha \end{pmatrix} \quad (7) \]

is the freestream velocity. As boundary conditions, on the inflow boundary \( \Gamma_{\infty}^- \), we have

\[ u = u_{\infty} \quad (8) \]
\[ \rho = 1 \quad (9) \]
\[ T = T_{\infty} = \frac{1}{\gamma (\gamma - 1) M_{\infty}^2} \quad (10) \]

where \( M_{\infty} \) is the freestream Mach number. On the outflow boundary, \( \Gamma_{\infty}^+ \),

\[ \frac{\partial u}{\partial n} = 0 \quad (11) \]
\[ \frac{\partial T}{\partial n} = 0. \quad (12) \]

If \( M_{\infty} < 1 \) then we also have \( \rho = 1 \). On the internal rigid boundary, \( \Gamma_B \), the no-slip condition of

\[ u = 0 \quad (13) \]

and

\[ T = T_B = T_{\infty} [1 + (\gamma - 1) M_{\infty}^2 / 2] \quad (14) \]

26
are enforced.

Other formulations of the Navier-Stokes equations are often used, for example taking \((p, u, v, T)^T\) as the primitive variables or the conservative form with variables \((\rho, \rho u, \rho v, \rho e)^T\). Hauke et al [6] show that any such formulation can be written in quasi-linear form as

\[
A_0 U_t + A_i U_{i,t} - (K_{ij} U_{j}),_i = F, \quad i, j = 1, 2. \tag{15}
\]

In the case of (1)-(3),

\[
U = \left( \begin{array}{c} \rho \\ u \\ v \\ T \end{array} \right), \text{ and } F = \left( \begin{array}{c} 0 \\ 0 \\ 0 \\ F(\nabla u) \end{array} \right).
\]

\(A_0, A_i, \text{ and } K_{ij}\) are 4x4 matrices which may depend upon \(U\). By working with (15), rather than (1)-(3), the approach taken in the next section may be applied more generally to formulations other than the primitive form used here.

3 DISCRETIZATION

The system of equations (15) are a set of partial differential equations which can be solved with a suitable finite element method. For the time being we shall consider the time independent equations only, and deal with the time derivatives in the next section.

It is well known that the Galerkin method alone lacks stability and is insufficient to generate accurate solutions for convection dominated flows. In order to overcome this problem, Hughes and Mallet initially established the SUPG method [7] for both the Euler and Navier-Stokes equations. Johnson has introduced the similar concept of streamline diffusion [8], and has proved some theoretical results, for example a consistency result for a system of conservation laws using entropy variables [9]. Hughes et al [1] developed the Galerkin/least squares operator, identical to SUPG for hyperbolic equations, but modified to incorporate diffusion [1]. The operator is added to the usual Galerkin formulation, and improves the stability whilst maintaining consistency.

Here we differ from the work of both Hughes and Johnson in that we use a simplified form of the operator, and employ the finite element method for spatial discretization only, rather than use combined space-time finite elements, thus allowing a choice of explicit or implicit solvers of different orders in time (see section 4). We have also omitted a shock-capturing operator, which may be necessary for flows of higher Reynolds number.

We first define our finite element spaces. Consider a triangulation of \(\Omega, \Omega^e\), consisting of non-overlapping triangles. Both the trial and the test functions are approximated by continuous piecewise linear polynomials, the precise spaces in which they lie being defined below.

Trial space:

\[
U^h = \{ U^h | U^h \in (C^0(\Omega))^4, U^h|_{\Omega^e} \in (P_1)^4, U^h \in S_T \}, \tag{16}
\]
where $\mathcal{S}_T = \{ U^h | U^h = (1, \cos \alpha, \sin \alpha, T_\infty)^T \text{on } \Gamma_{\infty}, u, v = 0, T = T_B \text{ on } \Gamma_B \}$ ($\rho = 1$ on $\Gamma_+^+$ is also required when $M_\infty < 1$) and $P_1$ is the space of linear polynomials.

Test space:

$$V^h = \{ V^h | V^h \in (C^0(\Omega))^4, V^h|_{\Omega^t} \in (P_1)^4, V^h \in T_T \},$$

where $T_T = \{ U^h | U^h = 0 \text{ on } \Gamma_{\infty}, u, v, T = 0 \text{ on } \Gamma_B, \rho = 0 \text{ on } \Gamma_+^+ \text{ if } M_\infty < 1 \}$. (17)

A variational equation can now be stated, based upon (15): find $U^h \in U^h$, such that for all $V^h \in V^h$, the following is satisfied

$$\int_\Omega (A_0 U^h_t + A_i U^h_i) \cdot V^h + V^h_i.K_{ij} U^h_j - F.V^h) d\Omega + \sum_e \int_{\Omega^e} \mathcal{L} V^h.r \mathcal{L} U^h d\Omega$$

$$- \int_{\partial\Omega} V^h.(K_{ij} U^h_j)n_i d\Omega = 0.$$ (18)

The first and last terms in (18) constitute the usual Galerkin formulation. The second integral contains the least-squares operator where we define

$$\mathcal{L} = A_0 \partial/\partial t + A_i \partial/\partial x_i - (\partial/\partial x_i)(K_{ij} \partial/\partial x_j) - \mathcal{F}.$$ (19)

It is important to note that this method is consistent (the operator vanishes at an exact solution). In Shakib et al [2], where entropy variables are used to derive a symmetric formulation of the Navier-Stokes equations, a stability proof is given along with a means of constructing $\tau$, a $4 \times 4$ matrix. From this, Hauke and Hughes [6] show how this $\tau$ can be transformed into a form suitable for other given formulations.

In this work however, we simply define

$$\tau = \frac{h}{2} I$$ (20)

where $h$ is a mesh size parameter and $I$ is the $4 \times 4$ identity matrix. This has the significant computational advantage of being very cheap to evaluate in comparison to the more complicated definition of $\tau$, where an eigenvalue problem needs to be solved for each element. The disadvantage of this simple choice of $\tau$ is that it does not provide the optimal value required for each component of the system, however practical experience of the authors suggests that for the types of flow under consideration here, the solutions obtained are not adversely affected by the use of (20).

4 SOLUTION ALGORITHM

The discretization (18) leads to a system of nonlinear equations to be solved, the size of which will depend upon the number of nodes in the mesh. We use unstructured meshes, as they are ideally suited for local refinement. They also allow the generation of meshes for complex geometries more easily than structured meshes, although they require a more complicated data structure. The algebraic equations obtained from (18) can be written as

$$G(W) = 0,$$ (21)
where \( \mathbf{W} \) is the vector of unknown coefficient values. The system is assembled element by element, using a three point gaussian quadrature rule to perform the numerical integration. Numerical experiments with higher order quadrature rules \cite{10} suggest that applying this rule on each element does provide sufficient accuracy. A Newton algorithm is employed to solve \((21)\), so with an initial guess \( \mathbf{W}^0 \), at each iteration, we have

\[
\mathbf{J}(\mathbf{W}^k)\delta \mathbf{W} = \mathbf{G}(\mathbf{W}^k) \tag{22}
\]

where \( \mathbf{J}(\mathbf{W}) \) is the Jacobian of \( \mathbf{G}(\mathbf{W}) \). The current solution is then updated,

\[
\mathbf{W}^{k+1} = \mathbf{W}^k + \delta \mathbf{W}, \tag{23}
\]

until convergence to within a given tolerance is obtained. The linear system \((22)\) is nonsymmetric, sparse and contains no band structure, due to the use of an unstructured mesh, so is solved using GMRES, an iterative solver. Performance of the Newton method is made more robust by the use of a linesearch backtracking algorithm \cite{11}, which is designed to improve it's global convergence.

GMRES is a Krylov Subspace based method, and requires storage of the subspace vectors generated at all previous iterations in order to generate an update vector. Without preconditioning the algorithm requires many iterations and hence excessive memory and expense, so it is important to precondition the system in some way. The preconditioner used here is an incomplete LDU factorization of \( \mathbf{J}(\mathbf{U}) \), where decomposition uses no fill-in, ie the sparsity pattern of \( \mathbf{J}(\mathbf{U}) \) is preserved in the lower and upper decomposed matrices.

The effect of preconditioning is to significantly reduce the number of iterations carried out, however, for anything other than low Reynolds number flows, there is still insufficient memory for to guarantee convergence of the linear system, even using restarts of the iteration. An alternative is to return to the full time-dependent equations and use time-stepping to reach steady state. The large storage requirement for subspace vectors is no longer a difficulty as only a small number are required for the linear systems at each time-step.

The discretization in time is implemented via an implicit, backward Euler scheme. For a time-step \( \Delta t \), at time \( t_m \),

\[
\frac{\partial \mathbf{U}}{\partial t} = \frac{\mathbf{U}_m - \mathbf{U}_{m-1}}{\Delta t}. \tag{24}
\]

By substitution of \((24)\) in \((18)\), the variational form becomes: given \( \mathbf{U}_0^h \in \mathcal{U}^h \), for \( m = 1,2,\ldots \), find \( \mathbf{U}_m^h \in \mathcal{U}^h \), such that for all \( \mathbf{V}^h \in \mathcal{V}^h \), the following is satisfied

\[
\int_{\Omega} \left( \frac{1}{\Delta t} A_0(\mathbf{U}_m^h - \mathbf{U}_{m-1}^h) + A_i \mathbf{U}_{m,i}^h \right) \mathbf{V}^h + \mathbf{V}_{i,j}^h K_{ij} \mathbf{U}_{m,j}^h - \mathbf{F} \cdot \mathbf{V}^h d\Omega \\
+ \sum_c \int_{\Omega_c} L \mathbf{V}^h \cdot \tau L \mathbf{U}_m^h d\Omega - \int_{\partial \Omega} \mathbf{V}^h \cdot (K_{ij} \mathbf{U}_{m,j}^h) n_i d\Omega = 0. \tag{25}
\]
Writing this as a algebraic system, we obtain
\[ \frac{M(W^m - W^{m-1})}{\Delta t} + G(W^m) = 0, \] (26)
where M is similar to the usual finite element mass matrix. Solution of (26) requires an initial condition \( W^0 \) and a time-step size \( \Delta t \). To obtain a steady solution, the current solution \( W^m \) is calculated at each time-step \( m \), via the same Newton algorithm that is used to solve (22), until
\[ ||W^{m+1} - W^m||_2 < \text{TOL}, \] (27)
where TOL is a specified tolerance.

We wish to reach this steady state as quickly as possible, and are allowed to take large time-steps without any loss of stability due to the implicit Euler scheme being used. In addition, convergence to steady state may be accelerated by the use of local time-stepping where the time-step size is allowed to vary over the flow region. The different element sizes means that information about the flow is being propagated through the domain at different rates, so by careful choice of \( \Delta t \) for each element, steady state is achieved more quickly. Shakib et al [2] give details of how local time-step sizes may be calculated. It should be emphasised that both local and global approaches to time-stepping lead to the exactly the same equilibrium solution as \( t \to \infty \) (although the solutions for small \( t \) may be very different indeed).

5 ADAPTIVE REFINEMENT

The methods discussed so far lead to a successful way of generating steady solutions of the Navier-Stokes equations on a given unstructured mesh. The accuracy of the solutions will be determined to some extent by the quality of this mesh, so in order to obtain accurate solutions one approach would be to have a fine mesh throughout the domain, requiring a very large number of elements.

A more efficient strategy is to start with a coarse mesh, and refine this mesh only in regions where the error is large. This has the benefit of achieving the same level of accuracy as the first approach, but with fewer elements. However, the local error in the solution cannot be known in advance and some other means of determining where to refine is needed, so that the size of the error can be controlled in some sense. The use of unstructured meshes makes the implementation of \( h \)-refinement as outlined here a relatively straightforward task. The algorithm for refining a particular subset of the elements is described by Jimack in [12] and is based on the work of Lohner [3]. If an element is to be refined, extra nodes are added at the midpoints of the three sides, and element is divided into four while neighbouring triangles are split into two, unless they too are to be refined, see figure 1.

Before this refinement algorithm may be used, the elements to be refined need to be identified. There are many criteria which can be used for this, and we have contrasted three possibilities. Two of these are based on physical properties of the flow; the gradient of the density and the vorticity. These will
be large near shocks and boundary layers and hence should allow the mesh to be refined in these regions. Experiments indicate that this is the case, but they often fail to resolve more subtle features such as the wake behind an aerofoil.

The third criterion considered is based on the residual of the original equations, rather than physical aspects of the flow. In [4], Hansbo and Johnson outline an adaptive algorithm for a one-dimensional linear advection-diffusion equation which is shown to be both reliable and efficient. The algorithm refines the mesh where the residual obtained by inserting the finite element solution into the original partial differential equation is above a specified tolerance. This can be generalized to the Navier-Stokes problem, where the residual on each element (ignoring diffusive terms) is defined to be

$$R(U) = |A_i U_{i} - F|.$$  \hspace{1cm} \hspace{1cm} (28)

When time-stepping is used, rather than reach a steady solution on one mesh before refining it where necessary, and starting again on the new mesh, the mesh is refined as the solution proceeds. This is achieved by introducing a refinement step each time a fixed number of time-steps have been completed. An element will be refined here, if

$$R(U) > \text{TOL},$$  \hspace{1cm} \hspace{1cm} (29)

for a specified tolerance TOL and the depth of the refinement has not exceeded a given level. The current solution is interpolated onto the new mesh, and the time-stepping procedure resumes, until a steady solution has been obtained.

6 NUMERICAL RESULTS

In this section we present some numerical results obtained using the methods outlined above. Both of the examples considered have been used extensively as test problems elsewhere. The solution procedure in both cases is the same, starting with a coarse mesh and the freestream values as initial conditions, except where dirichlet conditions have been specified. After every four local time-steps, the mesh is adaptively refined with a tolerance of 0.025 and a maximum refinement level of five, until the two-norm of the steady state finite element residual reaches 1.0e-6. The maximum size of the Krylov subspace in the GMRES routine is 25.

Transonic flow around a NACA0012 aerofoil

As a first example, we consider flow around a NACA0012 aerofoil, with a Mach number of 0.85, a Reynolds number of 2000 and no angle of attack (which
Figure 2: Subsection of initial mesh for Flow around NACA0012

Figure 3: Subsection of final mesh for Flow around NACA0012
was a test case of the compressible Navier-Stokes workshop held in Nice in 1985 [13]). The coarse mesh on which the problem was initially solved consists of 1617 elements, and a subsection of the mesh is shown in figure 2. Figure 3 shows this same subsection of the final mesh on which the solution was obtained, which consists of 8442 elements. The Mach contours of the flow are shown in figure 4. These results compare well with results from [13]. Flow features such as the wake are resolved, and coefficients of lift and drag fall within the range of the results obtained for this case in [13].

**Supersonic Flow over a Flat Plate**

Carter's problem consists of flow travelling over an infinitesimally thin plate with a Mach number of 3, and a Reynolds number of 1000. The computational domain covers the area $-0.2 \leq x \leq 1.2, 0 \leq y \leq 0.8$, with the leading edge of the plate at $x = 0$, the inflow boundary along $x = -0.2$ and and the outflow boundary on $y = 0.8$ and $x = 1.2$. Figure 5 shows the initial coarse mesh of 392 elements and figure 6 shows the final mesh consisting of 3497 elements. The pressure contours of the solution are shown in figure 7, and show that both the boundary layer and the shock are resolved successfully. For comparison, the plots of pressure and friction coefficients are similar to results obtained by Shakib et al in [14].

**7 CONCLUSIONS**

In this paper we have described a method which numerically solves the Navier-Stokes equations for compressible fluid flow. The method uses a stable finite element scheme, which avoids the main stability problems caused by the
Figure 5: Initial mesh for Flow over flat plate

Figure 6: Final mesh for Flow around flat plate
standard Galerkin method, even for very coarse meshes. This allows adaptive refinement to be used extensively and, when combined with a suitable choice of adaptive criterion, accurate solutions can be generated efficiently.

There are several areas of further work currently under investigation. When the Reynolds number is increased, the laminar flow rapidly becomes unsteady, so the use of global time-stepping is required to integrate consistently. In addition, for unsteady problems it becomes necessary to integrate accurately in time, and so this accuracy criterion, rather than the stability criterion used to ensure convergence to steady state, will determine the choice of time-step size. Work is in progress in which we aim to optimize this choice of time-step in terms of getting an approximate balance between temporal and spatial errors. Other improvements which we wish to incorporate soon are the introduction of a turbulence model to allow more complex flows to be simulated, including steady flows at high Reynolds numbers, and modifications to the mesh refinement algorithm. One particular aspect that is being investigated is to permit the movement of nodes within the mesh, thus enabling mesh elements to be aligned with properties of the flow, such as shocks or boundary layers, with the additional effect of smoothing the mesh.

References


[2] SHAKIB F., HUGHES T. J. R and JOHAN Z. - A New Finite Element For-


